Expert Systems With Applications 117 (2019) 29–41

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Assessing the performances of different neural network architectures for the detection of screams and shouts in public transportation

Pierre Laffitte[a,*], Yun Wang[b], David Sodoyer[a], Laurent Girin[c]

[a] Univ Lille Nord de France, F-59000 Lille, IFSTTAR, COSYS, LEOST, Villeneuve Ascq, F-59650, France
[b] Carnegie Mellon Department of Computer Science, Language Technologies Institute, Pittsburgh, PA, USA
[c] Grenoble-INP, Gipsa-lab, Grenoble, France

ARTICLE INFO

Article history:
Received 28 March 2018
Revised 31 July 2018
Accepted 29 August 2018
Available online 20 September 2018

Keywords:
Audio surveillance
Acoustic event detection
Transportation
Classification
Neural networks

ABSTRACT

As intelligent transportation systems are becoming more and more prevalent, the relevance of automatic surveillance systems grows larger. While such systems rely heavily on video signals, other types of signals can be used as well to monitor the security of passengers. The present article proposes an audio-based intelligent system for surveillance in public transportation, investigating the use of some state-of-the-art artificial intelligence methods for the automatic detection of screams and shouts. We present test results produced on a database of sounds occurring in subway trains in real working conditions, by classifying sounds into screams, shouts and other categories using different Neural Network architectures. The relevance of these architectures in the analysis of audio signals is analyzed. We report encouraging results, given the difficulty of the task, especially when a high level of surrounding noise is present.

1. Introduction

The proposed study is an attempt to build an intelligent surveillance system capable of automatically detecting abnormal situations in public transportation environments, such as underground subways or metros, based on the analysis of audio signals. Recently, Neural Networks have rose to prominence in most intelligent and expert systems, in applications as varied as image classification, customer behavior prediction, or medical diagnosis (Affonso, Rossi, Vieira, & de Leon Ferreira de Carvalho, 2017; Eshtay, Faris, & Obeid, 2018; Vanneschi, Horn, Castelli, & Popovič, 2018). In the literature, intelligent systems for automatic surveillance generally use video signals (Baran, Rusc, & Fornalski, 2016; Foggia, Petkov, Saggese, Strisciuglio, & Vento, 2016; Hata, Kuwahara, Nozawa, Schwenke, & Vetro, 2005; Orhan Bulan, 2013; Velastin, Boghossian, & Vicencio-Silva, 2006), but the use of audio can be an interesting complement as it helps circumvent issues inherent to video signals such as vision field obstruction or lighting changes. In this paper, we focus on a particular context for surveillance systems; namely, that of public transportation. While most research in this specific application also use video signals (He, Chen, Jiang, Lu, & Yuan, 2017; Orhan Bulan, 2013; Velastin et al., 2006), we tackle the issue from a different angle by using audio signals instead. This approach has been introduced within the framework of public transportation surveillance; in (Rouas, Louradour, & Ambellouis, 2006) to detect screams, in (Ganansia et al., 2011) to detect and localize shouts and graffiti sprays, in (Zouaoui et al., 2015) to detect abnormal sounds, and within the framework of general surveillance systems in (Valenzise, Gerosa, Tagliasacchi, Antonacci, & Sarti, 2007) to detect screams and gunshot sounds. The contribution of this paper is mostly experimental and applicative, not methodological. It shows that using state of the art neural networks can serve the purpose of audio surveillance in public transportation, and reveals some interesting characteristics of those methods, as well as how a good understanding of them can lead to an improvement in performances.

From a more general perspective, the task we address is referred to as acoustic event detection (AED), which is a research topic of growing interest in the audio signal processing community (Chu, Narayanan, & Kuo, 2009; Crocco, Cristani, Trucco, & Murino, 2016; Dennis, Tran, & Chang, 2013; Diment, Cakir, Heittola, & Virtanen, 2015; Fernández-Delgado, Cernadas, Barro, & Amorim, 2014; McLoughlin, Zhang, Xie, Song, & Xiao, 2015). The DCASE challenge (Virtanen et al., 2016) attests to the popularity of this task, which has many applications ranging from smart houses involving automatic systems for domestic events detection using audio and video data streams (Wang, Lin, Chen, & Tsai, 2014; Wu, Gong, Chen, Zhong, & Xu, 2009), to humanoid robotics where an

* Principal Corresponding Author.
E-mail addresses: pl.laffitte@gmail.com (P. Laffitte), yun.wang@cs-cmu.edu (Y. Wang), david.sodoyer@ifsttar.fr (D. Sodoyer), laurent.girin@gipsa-lab.grenoble-inp.fr (L. Girin).

https://doi.org/10.1016/j.eswa.2018.08.052
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

audition model is a prerequisite for natural human-robot interaction (Janvier, Alameda-Pineda, Girin, & Horaud, 2012; Nakadai, Matsuura, Okuno, & Tsujino, 2004; Noda, Yamaguchi, Nakadai, Okuno, & Ogata, 2015; Wu et al., 2009), including automatic surveillance applications (Foggia et al., 2016; Velastin et al., 2006). AED can be thought of as a combination of automatic audio segmentation and audio event classification (Janvier, Alameda-Pineda, Girin, & Horaud, 2014), hence adding to the classification task the difficulty of identifying the temporal location of the audio events (Phan, Maas, Mazur, & Mertins, 2015), as it does not rely on prior segmentation of the data. Audio event classification techniques in the state of the art are diverse, with many different combinations of features and classifiers: Mel-Frequency Cepstral Coefficients (MFCCs) classified with Gaussian Mixture Models (GMMs) (Pohjalainen, Raitio, & Alku, 2011), with Support Vector Machines (SVMs) (Huang, Chiew, Li, Kok, & Biswas, 2010; Lei & Mak, 2014; Wu et al., 2009), with Hidden Markov Models (HMMs) (Ntalampiras, Potamitis, & Fakotakis, 2009); MFCCs and other spectral features classified with GMMs (Chu et al., 2009; Gerosa, Valenzise, Tagliasacchi, Antonacci, & Sarti, 2007), with the *k*-nearest-neighbors algorithm (kNNs) (Chu et al., 2009), and more recently with random forests (RFs) (Phan et al., 2015); Gabor features classified with GMMs (Geiger & Helwani, 2015; Gerosa et al., 2007) and with SVMs (Wang et al., 2014); all-pole group delay features classified with Deep Neural Networks (DNNs) (Diment et al., 2015); Gammatone-Wavelet features classified with SVMs (Valero & AlÃas, 2012); spectrogram image features classified with kNNs (Dennis et al., 2013), SVMs and DNNs (Diment et al., 2015; McLoughlin et al., 2015; Wei, Li, Pham, Das, & Qu, 2016); MFCCs and deep scattering features classified with RFs and the k-means algorithm (Salamon & Bello, 2015). However, the use of Neural Networks is becoming more and more prominent, be it Deep (Sharan & Moir, 2017), Convolutional (Hershey et al., 2017; Lee, Kim, Park, & Nam, 2017) or Recurrent Neural Networks (RNNs) (Wang & Metze, 2017). The task defined in this study is to detect violent events in the subway via automatic detection of screams and shouts emitted by the people involved in the events. Such events span different cases, such as people in physical difficulty, people quarreling, panic situations, calls for help, etc. Shouts and screams are here defined as loud vocal sounds with and without explicit semantic content, respectively. Since it turns out that scream occurrences are outnumbered by shout occurrences in our database, in the following we employ the general term “shout” to characterize the overall set of abnormally loud sounds generated by people subject to or witnesses of violent events. Although such alert signals are quite specific, this task remains challenging since there generally exists a large variability between different realizations of screams and shouts, depending on the causing events, a large variability of “speakers”, number of persons involved, their emotional state, etc. The first specific aspect of the present work is the rarity of the data: in order to design a realistic AED system, a dedicated database was recorded, consisting of real signals recorded in the Paris subway (called ’Metropolitain’, or simply Metro). A whole Metro train was booked for the recording sessions, thanks to the Paris public transportation authority (the RATP) being a partner of the research project which frames this study. Abnormal situations were enacted by actors, including many extra participants representing the crowd of passengers. As a consequence all recordings used in the present study are real and not derived from synthetic signals or simulated acoustic mixes, and the size of the corpus cannot match that of handcrafted synthetic data such as in (Lafay, Lagrange, Rossignol, Benetos, & Roebel, 2016) and (Wang & Metze, 2017). The second specific aspect concerns the characteristics of this environment which is very noisy and variable. It contains many objects that can act as sound sources and filters, shaping the acoustic scene; noise from the vehicle itself (e.g., motor noise, boogie-rails frictions), noise coming from the surrounding environment (e.g., railway traffic, station noise, loud-speaker announcements), and noise produced by passengers. Within such an environment, the choice of target classes used to define the acoustic landscape is crucial, especially within the framework of audio event detection. The classification techniques used here are different architectures of neural networks applied on acoustic MFCC features, namely (feedforward) deep neural networks (DNNs), convolutional neural networks (CNNs) and recurrent neural networks (RNNs; in particular we used Long Short Term Memory (LSTM) cells). We set the main task as a 3-way classification task, with target classes defined as shouts (comprising shouts and screams), conversational speech and background noise. Besides this main task, we created another set of 14 classes (by dividing each of the previous 3 classes into 5 smaller sub-classes) on which we performed a 14-way classification. We report results of an extensive benchmark made using the 3 types of neural networks, for both the 3-class problem and the 14-class problem. The remaining of this paper is organized as follows: Section 2.1 gives a detailed description of the database used for the experiments. Section 3 presents the methodological background of neural networks. Section 4 presents the parameters and settings of the experiments we carried out, while Section 5 reports the results. Finally, Section 6 draws some conclusions and perspectives.

2. Database and analysis of environment impact

As stated in the introduction, the present task is to detect dangerous situations occurring in the metro by analyzing the audio environment. In an effort to account for the likelihood that the situation exhibits a potentially dangerous/violent aspect, we devised the following three classes;

- **Shouts** (includes screams and all overlapping background sound),
- **Speech** (includes all overlapping background sound),
- **Background sounds** (all sounds not pertaining to the previous two classes; no speech nor shout sounds are assumed to be present).

Additionally we define some more classes to describe the acoustic environment related to the metro’s trajectory during its course:

- **Stand-by** (acoustic scene when the train is idle, in the station),
- **Compressor** (noise from compressor, very specific),
- **Departure** (acoustic scene during departure, when speed increases from zero to full-speed),
- **Cruise** (acoustic scene during the period of time when the train is at full speed),
- **Arrival** (acoustic scene during arrival in station, when speed decreases to zero).

Those classes will be used in order to help classify the main three classes.

2.1. Data acquisition

Within the framework of research project DéGIV (Zouaoui et al., 2015), a subway train from the automatic line 14 of the Paris Metropolitain was reserved for the recording sessions. An Omnidirectional microphone (C224 6v cell from ELNO brand) was placed on the ceiling of the metro car, and a low-latency JACK server audio interface, made specifically for this project, was used to record the signal, producing 16-bits/48-kHz PCM signals. Several sessions took place between 10 am and 4 pm while the train was running its usual course, among other trains from the same subway line, running between different stations and stopping at all of them.

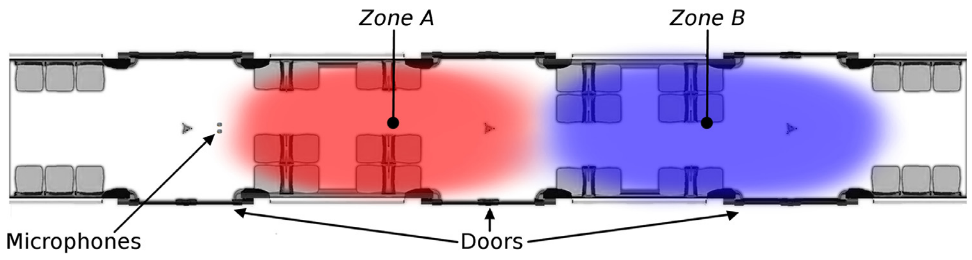


Fig. 1. Graphical representation of one metro car, where the microphones are represented by the two dots to the left of the red square. The red zone is the near-zone and the blue zone is the far-zone. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

For security matters, the train did not allow regular passengers in, and three sets of actors played several pre-defined alert scenarios displaying a situation of security matter (robbery, assault, fight, person falling, etc). Numerous extras were present to simulate regular passengers. For data recording, two microphones were placed on the ceiling about 10 cm from one another (along with a video camera, although the latter was not used in this study). Different settings were defined, in which each scenario was played; two different zones, depending on the distance with the microphones, as shown in Fig. 1[1]: close-distance ($\sim$1 m to 2.5 m) in red and far-distance ($\sim$3 m to 4.5 m) in blue; two crowd densities: heavy density (between 12 and 17 people involved in the scene) and low density (between 5 and 7 people). Every alert scenario was repeated for all possible combinations of settings. Violent scenes were played and captured while the train was either accelerating, moving at stationary high speed, or braking, but not when it was stopped at a station for security reasons. For the same reasons the doors never opened at the stations to make sure no regular passengers would get on the train. To compensate, violent scenes and sequences of doors opening and closing were also captured while the train was in the workshop. Additionally a large amount of sequences of chatter among passengers was captured. Finally, our database contains all the noises induced by the train activity (doors opening and closing, brake compressors draining air, doors closing signal, etc.). We believe these data convey a realistic diversity of signal occurrences for the considered application (different scenarios, different source-to-microphone positions, different noises, etc.). A preliminary study using this database has been presented in (Laffitte, Sodoyer, Tatkeu, & Girin, 2016).

2.2. A short analysis of data variability

It is important to note that in this transport environment, speech or shout signals can contain a huge amount of noise, making the distinction between speech, shouts and background noise often very difficult. In this subsection, we illustrate this notable characteristic of our database. Fig. 2 displays the spectrograms of segments of speech (Fig. 2-(a,b)) and shout (Fig. 2-(c,d)), in a situation where the train is idle (stopped at the station, in a static/immobile state) (Fig. 2-(a,c)) and in a situation where the train is at full speed (Fig. 2-(b,d)). By comparing Fig. 2(a) and Fig. 2(b), we can see that full speed implies a much higher level of noise than idle, and consequently the speech signal is much severely masked at full-speed. Because shouts have more energy than speech, the full speed has a lower impact on shouts, as seen by comparing Fig. 2(c) and Fig. 2(d). Still, in Fig. 2(d), the shout signal is somewhat "blurred" by the full-speed noise.

To quantify the difference between the full speed and the idle conditions in terms of signal energy, we calculated the energy of individual 10ms-frames extracted from typical portions of our dataset (33.7s in total) and manually labeled (see Section 2.3). Fig. 3(a) displays the histograms of frame energy (in dB) for the two conditions (idle and full speed), for frames that contain no speech or shout signal (thus pure background/vehicle noise). We can see that the two histograms (therefore the two conditions) are clearly separated, with an average difference of about 60 dB, which is a large value. Fig. 3(b) also displays the histograms of frame energy for the two conditions, but this time for frames containing shouts. We can see that the energy of shouts clearly exceeds the energy of the background noise when the train is idle, and that the amount of additional energy from shouts spreads over a large range of values. When the train is at full speed, the (spread of) additional energy from shouts is more moderate. There is a significant overlap between the energy distributions of the full-speed noise-only frames and the full-speech noise + shouts frames. For this kind of frames, we can expect the presence of full-speed noise to make the classification into the shout category particularly difficult. For frames above a certain threshold, −20 dB for instance, the classification into shouts may be easier. Note that this confirms the example spectrogram plot in Fig. 2(d) where some shout regions are masked by the full-speed background noise, and some other regions are more visible.

2.3. Data labeling

The data were manually cross-labeled by two different audio experts (two of the authors). The first expert labeled the entire dataset and the second expert ran a second pass to validate or correct the labeling of the first expert. Our first approach consists in decomposing the entire dataset in the three main target classes described above. The first class is obviously driven by the targeted application. Recall that, due to the low proportion of scream occurrences, this class contains both screams and shouts. The distinction between speech (+ background sounds) and background sounds (only) corresponds to the most obvious distinction appearing in the data in the absence of shouts. We believe that the definition of those three categories can help the detection methods in detecting shouts, as opposed to a two-category classification (shout (+ background sounds) vs. "everything else"). Indeed, in the latter case, the mixing of speech with background sounds in a unique class may disturb the classifier, since the "proximity" between those two types of signals is questionable compared to the proximity between speech and shouts. In the following, for simplicity, we refer to the shouts (+ screams) / speech / background sounds classification problem as "the 3-class problem".

In light of the characteristics of our dataset exposed in the previous section, we attempted to better account for the noisiness and variability of the physical environment by explicitly classifying the background sounds (i.e. the environment/vehicle sounds) into subclasses, superimposed on shouts and speech. To this aim, based on an extensive listening of the database, we defined the following 5 subclasses, illustrated on the spectrogram in Fig. 4:

[1] Image from research project DÃ©GIV (Détection et Gestion d'Incidents dans une Voiture ferroviaire) co-funded by"FUI-BPI France" and "Conseil Général de l'Essonne"

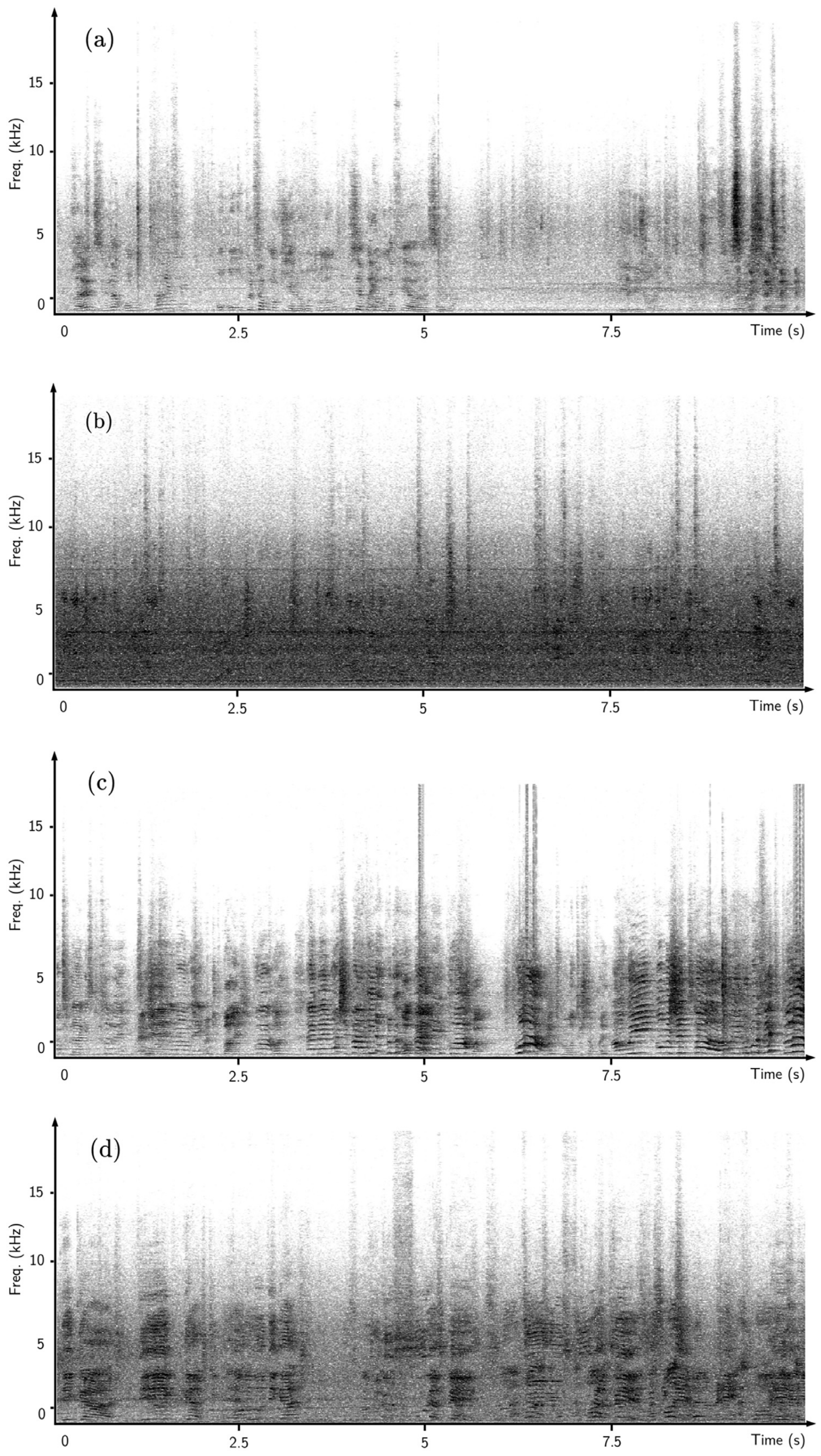


Fig. 2. Example spectrograms of speech during portion on standby (a) and at high speed (b). Example spectrogram of shouts during portion on standby (c) and at high speed (d).

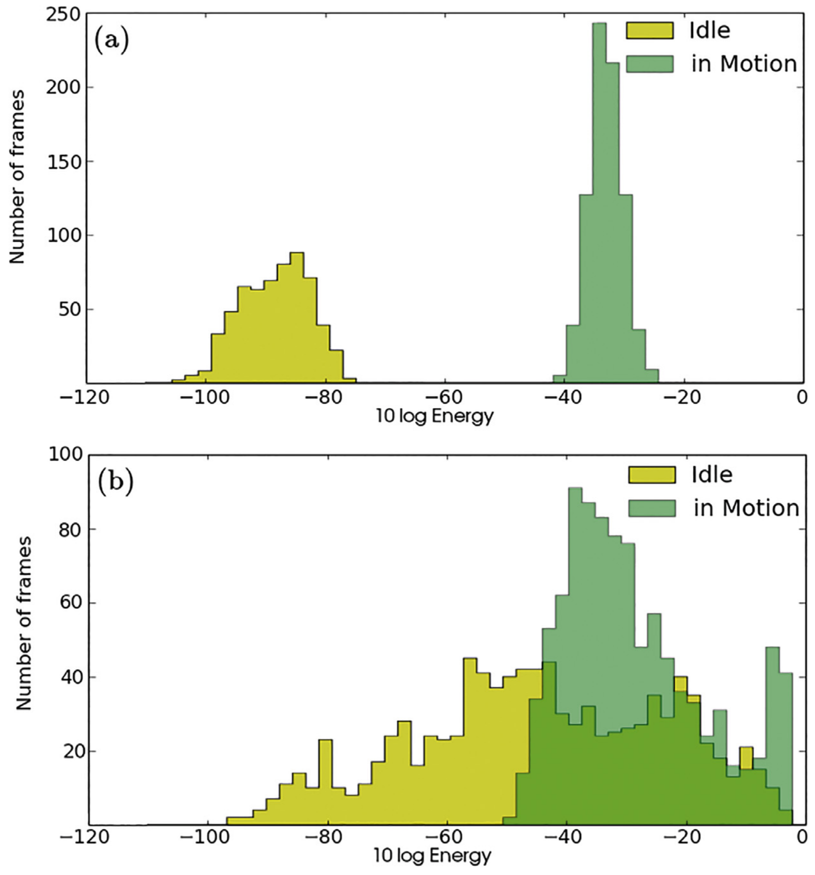


Fig. 3. Distribution of short-term (10ms) frame energy when the train is idle vs. when it is at high speed. (a): background noise only; (b): background noise + shouts.

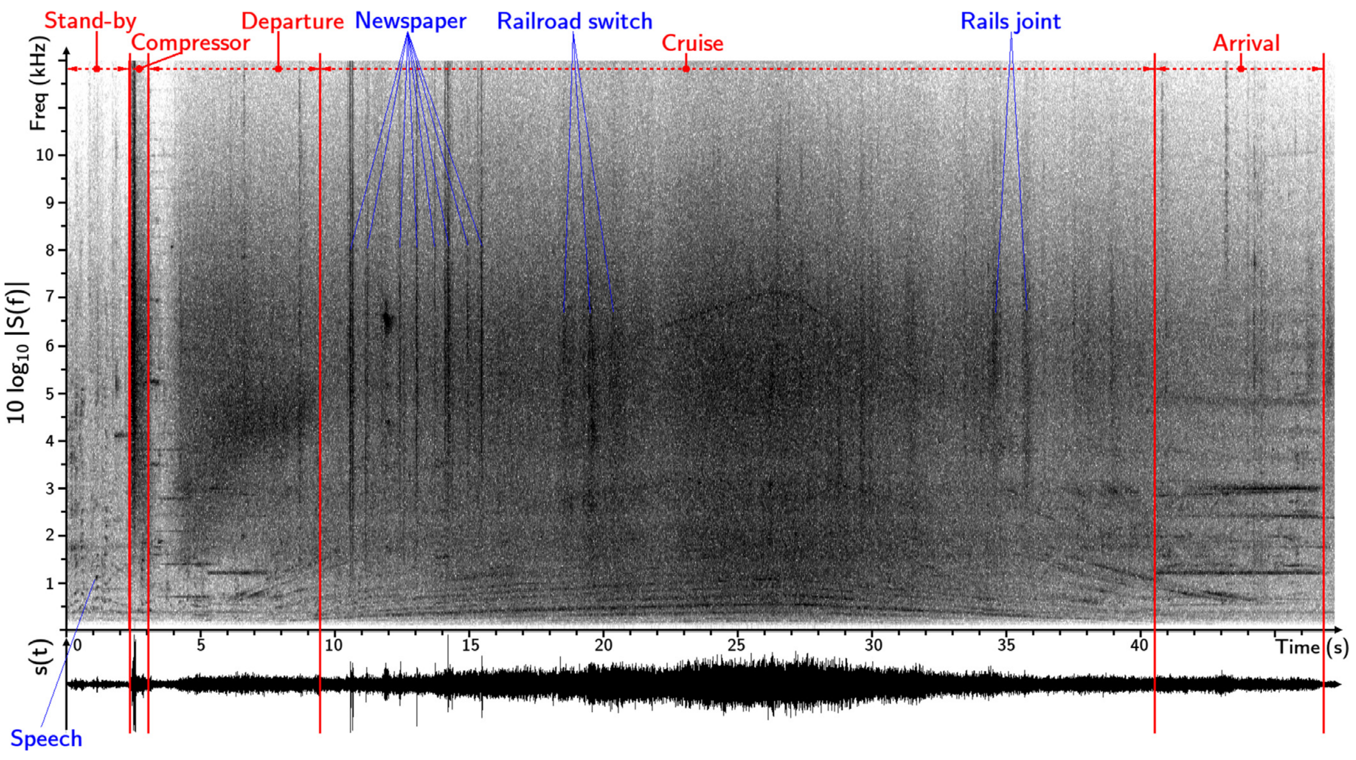


Fig. 4. Example spectrogram of a recording between two stations of line 14. The red lines show the boundary of the 5 environment-related classes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Duration (in seconds) of the 15 fine classes and the 3 broad classes in the train/validation/test datasets.

	Stand-by	Departure	Start compressor	Cruise	Arrival	Total
Background noise	290/36/57	100/28/22	19/4/4	663/115/88	69/11/9	1292/190/194
Speech	650/68/187	124/8/40	19/3/6	472/57/105	91/24/25	1554/152/356
Shout	65/1/13	6/0/2	-	228/41/14	7/1/4	358/43/30

- **Stand-by**: This class is defined as all situations where the train is not moving. When the train is operating on the line, the duration of this class usually stands between 15–20 s. When the train is parked in the workshop or at the end of the line, the duration ranges from 2 min (end of the line) to 10 min (workshop).
- **Compressor**: Specific noise made by the vehicle's compressors when about to start moving. Its duration is around 1 s. It is composed of noise ranging from 2000 Hz to 22050 Hz with diminishing energy after 600 ms.
- **Departure**: The train engages in the process of starting its engine and moving forward, between the compressor noise and the moment it enters the tunnel. An energy stain specific to this event can be observed, beginning around 2000 Hz and progressing towards 5000 Hz at the end. This sound is very distinct to the ear.
- **Cruise**: Time span between two stations, when the train is inside the tunnel. The duration of this portion varies from 40 s to 90 s, depending on the distance between stations. Its spectral content varies according to the speed. In terms of spectrum representation, the speed leads to harmonics that can be seen around 7000 Hz in Fig. 4. The stages of acceleration and deceleration can be identified, with a transition around 26 s. Moreover this event is composed of very energetic noise along the entire frequency axis, stemming from the mechanical rolling.
- **Arrival**: This event occurs between the moment the train comes out of the tunnel and when it stops. As with the Departure event, it is easily identifiable upon listening. Its duration depends mainly on the length of the station.

In the example shown in Fig. 4, some additional sounds can be identified such as; speech signal (between 0 s and 5 s), sound of crumpled newspaper right underneath the microphones (between 10 s and 16 s), internal and external mechanical noises due to railroad switch (between 18 s and 20 s) or due to the joint between two rails (between 34 s and 36 s).

Each of the 5 background sound classes are then combined with the three main (shout/speech-related) classes, to perform classification on $3 \times 5 = 15$ composite classes (e.g. speech + cruising, shouts + compressor, etc). The result of this classification can then be marginalized over the 5 background sound classes, so as to end up with a 3-way classification result, over the three main classes. The goal of this approach is to see if it alleviates the burden of the classifier, compared to the raw 3-class problem, by reducing the variability of each class despite a higher number of classes.

We split our database in three different sets: a training set, a validation set and a test set. All the results presented further in this paper were calculated on the test set. The total amount of data was approximately 3800 seconds, distributed across training, validation and testing datasets as shown in Table 1. This table gives the amount of data in seconds for each sub-class, contained in the training dataset, the validation dataset, and the test dataset. For example, data pertaining to sub-class Background noise + Stand-by is scattered across the three datasets in the following proportion: 290s for training, 36s for validation and 57s for test. Because the long-term temporal structure of the environment seemed crucial to us, we did not want to artificially cut any sequences containing a natural continuity. Thus, our dataset was processed so as to keep the chronological order of appearance of sounds. As a result, a precise control over the data distribution for each class across train, valid and test dataset was beyond our reach.

As can be seen in Table 1 that the sub-class "Shout + Start compressor" contains no data at all. This is because the occurrences from class "Start compressor" were too short and those from class "Shout" were too sparse to get an overlap between occurrences from these two classes. As a result we obtain a total of 14 subclasses. In the following, for simplicity, this second approach is referred to as "the 14-class problem". Classes "Shout+Departure" and "Shout+Arrival" have respectively 2s and 4s of training occurrences, which will obviously limit the complexity of our model.

Table 1 shows that our dataset suffers from imbalance, i.e. the different classes have quite a different amount of occurrences (and thus training, validation and testing data). In particular, we already mentioned that scream occurrences are quite rare. Even screams + shouts together, i.e. the shout class, is sparse compared to the speech class and the background noise class. Class imbalance is a known issue in the machine learning literature (He & Garcia, 2009; Japkowicz & Stephen, 2002) and it is not perfectly clear how to take this problem into account within a discriminative model framework such as the DNN framework (as opposed to generative models where it can be tackled by inserting prior class distribution in the model). However our situation is a very specific case scenario, wherein the classes are naturally imbalanced. In fact, scream/shout occurrences are even more imbalanced (under-represented) in real life than in our dataset. We have not had the chance to conduct experiments to address this problem extensively in our study, however, we decided to simply create a dataset with all available data, thereby displaying a higher proportion of occurrences of the rare class than would be found in fully real conditions, because the absolute number of occurrence is already low. Indeed, the more data the more accurate the model. In short, we tried to find a trade-off between (limiting the) over-representation of screams and shouts in our dataset w.r.t. reality, and (limiting) their under-representation w.r.t. other classes.

3. Methods

The machine learning systems used to perform the classification tasks introduced above, as well as their theoretical background, are presented in this section.

3.1. Artificial neural networks

3.1.1. Generalities

In the present study we used artificial neural networks (ANNs) to achieve the classification task. More precisely we used classical feed-forward deep neural networks (DNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTM) networks. ANNs have been extensively studied and used as a powerful tool for regression and classification problems, for many applications. Because they are extensively described in the literature (Bengio, Lamblin, Popovici, & Larochelle, 2007; Bishop, 1995; Funahashi, 1998; Gish, 1990; Lippmann, 1994; Zhang, 2000) and because the contribution of this paper is mostly experimental and applicative, and not methodological, we only give a brief overview of the general principles of ANN-based classification, and of the different neural architectures that we have tested. The reader is referred to the above-mentioned references for a detailed description of ANN architectures and methodology.

In a classification paradigm the output of an ANN is an estimate $\mathbf{y}_n$ of the posterior probability $p(c|\mathbf{x}_n)$ of each class $c \in [1, C]$, where C is the number of classes, given an input data vector $\mathbf{x}_n$, where n is here the time index. The selected class is the most likely, i.e. the one with the highest posterior probability.

ANNs are associated with supervised learning: they are trained on labeled data because they need some ground-truth information to learn from so as to adjust their weights accordingly. The training, i.e. the optimization of the parameter values (layer weights, see below) for optimal data discrimination, is done via maximization of a likelihood function. The latter encodes the link between input and output data (i.e. the associated labels) from the training dataset. Maximization of this likelihood function is usually achieved by means of a gradient descent algorithm which sequentially updates the parameters. Again, details on learning (deep) ANNs can be found in (Bengio et al., 2007; Bishop, 1995; Gish, 1990).

3.1.2. Deep neural networks

A classical feed-forward neural network is composed of several layers of elementary units (also called cells). Each unit performs a non-linear transformation on a weighted sum of its inputs, which are the outputs of the previous layer. The input layer is simply the input vector, and the output layer computes the posterior probability of each class (one class per output unit), deciding which class the input belongs to. Intermediary layers between the input and output layers are called hidden layers, and when the number of hidden layers is larger than one, an ANN becomes “deep”. In that case The units operate in chain, building more complicated transformations of the input. The idea is that the resulting non linear functions can be arbitrarily complex. Conceptually, the units represent unknown factors which “explain” the data. The more layers, the more conceptual these factors become, eventually representing some abstractions of the data that enable the output layer to discriminate it more efficiently.

3.1.3. Convolutional neural networks

CNNs were introduced in (Fukushima, 1980) as a visual pattern recognition technique unaffected by shifts in position. As such, their power lies in their ability to consider 2-dimensional data. The first layers filters the input, analyzing it across both dimensions . Each layer consists in a certain number of “plies” or “feature maps” (the term varies in the literature) each corresponding to a filter. Concretely, each hidden unit in a given convolution ply takes a small subset of the previous layer’s output as input, and performs a linear combination with a weight matrix. Since the weight matrix is the same for all hidden units within a ply, the operation amounts to performing a convolution. Several such plies are used to create multiple (hidden) filtered versions of the input. A pooling ply is generally used after a convolution ply, where a mathematical max operation is performed to reduce the dimensionality. (Abdel-Hamid et al., 2014; LeCun & Bengio, 1998) present some applications of CNNs for automatic speech recognition (ASR).

3.1.4. Recurrent neural networks

RNNs were made to circumvent DNNs’ inherent flaw in processing sequences. Their general architecture is the same as classical feed-forward neural network ANN , but they are capable of considering a series of input vectors as a temporal sequence, by means of a memory. Two forms of RNN exist: unidirectional (used in this study) where the flow of information entering the memory goes forward in time, and bidirectional, (which will not be discussed here) where the information travels both ways (forward and backward in time). The memory is contained in the hidden cells, who are able to retain and accumulate passed information and subsequently use it when necessary to output classification decisions. A more advanced type of cell called Long Short Term Memory (LSTM), introduced in (Hochreiter & Schmidhuber, 1997), actually uses a separate container for its memory: the state, denoted c_n, where all information are accumulated. Three different operations are performed on that state at each time-step: erase pieces of information, add some new elements, and output what is relevant. In order to do this the LSTM cell analyses the current input x_n and the previous output h_{n-1}; it filters them through three different sigmoid gates corresponding to the three operations. Each gate outputs values between 0 and 1, deciding what to let through and how much of it. These outputs are then combined together to discard some information, add some new into the memory and release some which is then used to update the state c_n and compute the current output h_n. LSTMs have recently shown excellent performances in ASR (Graves, r. Mohamed, & Hinton, 2013; Graves & Schmidhuber, 2005; Miao, 2014) and AED (Parascandolo, Huttunen, & Virtanen, 2016), and become the new state-of-the-art in these domains and other domains of pattern recognition and mapping.

3.2. Processing temporal sequences with ANNs

Within the framework of audio event detection, the target sounds are naturally characterized by the temporal evolution of their spectral content. The spectral content on a short duration (a few tens of ms) is usually represented by a vector of spectral coefficients such as Mel-frequency cepstral coefficients (MFCCs). Successive vectors are obtained by sliding the analysis window, generally by a portion of its size, to ensure some overlap between successive frames. Different realizations of target sounds usually have a different duration, which implies that the total number of spectral coefficient vectors required to fully represent a given sound event also varies, in an unpredictable manner (for instance, a shout expressing fear can be quite short or quite long depending on the speaker).

Unfortunately, in their “default configuration”, DNNs only take fixed-size inputs and consider each successive input vector x_n independent from one another, failing to capture the sound’s temporal structure. To remedy this, it is common to feed a DNN with a supervector obtained by concatenating K consecutive vectors of spectral parameters. For example, if some delay is tolerated, at time n (at the frame sampling level), the DNN can be fed with a supervector encompassing all input vectors from frame $n - K/2$ to $n + K/2 - 1$. Such DNNs are sometimes called “contextual” DNNs. One advantage is that they can learn the temporal structure of sounds on the scale of K successive frames, but are limited to that length. Yet, K successive frames may very well represent only a portion of a (long) sound, and conversely may also be longer than a given sound and include several sounds that we want to discriminate (which happens when the block of K frames overlaps the boundary between two sounds. This situation is detailed below).

As for CNNs, they consider 2D inputs (i.e. images). Therefore the temporal dimension can be taken into account in a simple manner by feeding the CNNs with (portions of) 2D spectrograms formed with K consecutive vectors. The convolution operation then allows an efficient joint exploitation of the spectral correlations in both time and frequency dimensions. However, the problem remains of fixed-sized processed blocks that contrast with variable-size sounds.

The intrinsic capacity of RNNs to process sequences of input vectors makes them perfectly suitable for audio sequences. Their main advantage over DNNs and CNNs is the use of a memory linking consecutive inputs together over a possibly very long temporal horizon, set automatically. This memory allows the network to capture the time structure of the audio patterns without the constraint of an arbitrary, fixed-size segmentation, even when the RNNs is fed one vector/frame at a time.

In the present study we use the “contextual” configuration for DNNs and CNNs, i.e. we process blocks of K consecutive input vectors. In the literature, in such a configuration the classification task is often performed and evaluated offline with pre-segmented audio patterns. This means that at training and testing time, only sequences with K vectors belonging to the same sound are considered. However, in the present framework of alert signals detection, we have to perform an online analysis on an incoming “continuous” audio stream. The sound patterns that we must detect are thus no longer pre-segmented. In this case, concatenating the K “current” consecutive input frames (for instance the current input frame and the $K-1$ past frames) does not necessarily match the onset and offset of the sound pattern. In the neighborhood of class boundaries (when we go from one type of sound to another), a supervector can contain vectors from the first sound concatenated with vectors from the second sound. This naturally brings some ambiguity in the classification process (test time) and/or the estimation of the models (training time). In order to evaluate the influence of this problem in the classifier’s performances we considered two configurations: pre-segmented data and streaming data. In the latter, the supervector is formed by concatenating consecutive input frames regardless of their labels. The label of a supervector containing segments pertaining to two different classes is chosen by taking the class with the highest number of frames within that supervector. More precisely, the results presented in this paper were obtained in 3 different configurations: i) using pre-segmented data for both training and testing, ii) using pre-segmented data for training and streaming data for testing, and iii) using streaming data for both training and testing. Pre-segmentation was done manually. Configuration i is unrealistic in a practical application (in the absence of automatic robust pre-segmentation which is a problem on its own). We used it mainly as a baseline for the other two configurations. Configurations ii and iii are realistic since online processing on unsegmented data is required only at test time. Concretely, tests in Configuration iii will indicate if including information about the surrounding sounds at training time is a good thing for the classification system.

3.3. Post-processing for final decision

For DNNs and CNNs, the block of K consecutive vectors slides by $K/2$ to proceed to the next detection. Therefore, an estimate of the input target class is provided every $K/2$ frames, hence at every frame index $n = pK/2$. With RNN, the estimation is made for every input frame x_n, hence at every frame index $n = 1$ to N.

To further improve the results, we tested two different post-processing algorithms:

- A smoothing algorithm that prevents a decision at time n from diverging if the two directly adjacent decision, $n-1$ and $n+1$ are the same. In that case, decision at time n is forced to be the same that at time $n-1$ and $n+1$.
- A majority voting algorithm that looks at $N_{maj.vot.}$ consecutive output decisions, and finally outputs the class with most occurrences for the whole set of $N_{maj.vot.}$ consecutive decisions.

4. Experimental set-up

4.1. Feature extraction

The signal extracted from the microphones was 16 bits PCM samples at 48-kHz. For DNNs and RNNs, the audio feature vectors consisted of 40 Mel-frequency cepstral coefficients (MFCCs) calculated every $T_i = 10$ ms, over a $T_w = 20$ ms time window (hence a 10-ms overlap), from the 48-kHz audio recordings, using the SoX software tool (SoX, 2015). Each audio sequence was thus turned into a series of such audio feature vectors. According to the aforementioned parameters, 1s of audio produced 100 feature vectors, which were then concatenated (in the DNN and CNN case) to form input examples. For CNNs, 40 Mel-frequency spectral coefficients (MFSCs) instead of MFCCs were used as defined in (Abdel-Hamid et al., 2014), in order to keep the locality property of the spectrum.

Table 2
Data-to-parameter ratios for different window sizes and DNN architectures.

number of frames K	10	20	50
1*128	473	228	79
2*128	349	-	74
1*512	115	57	20

Table 3
Data-to-parameter ratios for different numbers of context frames.

	1*128	1*256	1*512	2*128
RNN/LSTM	152	34	9	48

4.2. Configurations

As seen in Section 2.3, we tested two strategies to classify the 3 main target classes (Shouts, Speech and Background sounds): First we considered them without any information about the (vehicle) environment, and second we combined them with the 5 background sound classes, resulting in 15 composite classes.

In the 3-class case, we tested all three types of considered networks: DNNs, CNNs and RNNs, using classical sigmoid units. For DNNs, the following parameter settings were tested and compared:

- Number of hidden layers: 1 or 2,
- Number of units per layer: 128, 256 and 512,
- Number K of consecutive frames used to form the input supervector: 10, 20 and 50 frames.

The corresponding ratios between number of scalar training data and number of network parameters (called data-to-parameter ratios; DPRs) are given in Table 2.[2] For CNNs, the following parameter values were tested:

- Number of convolution plies in each CNN layer: 150, 100,
- Size of CNN convolution filters: 5×5, 5×10, 8×10, 8×10,
- Number of hidden (convolution) layers: 1 and 2,
- Size of the top NN layer: 1×128, 1×512, 2×128.

Due to a high number of tested configurations for CNNs, we do not report DPRs for all of them. We will give the DPRs corresponding to the selected results in Section 5.

For RNNs, we tried the following configurations:

- Number of hidden layers: 1 or 2,
- Number of units per layer: 128, 256 and 512,

The corresponding DPRs are given in Table 3. Here again, the RNN’s complexity is limited by the DPR to avoid the over-fitting problem.

[2] As seen in Table 2 the DPR did not allow us to test more complex networks architecture, to avoid training data over-fitting, which is why our deepest network only had 2 hidden layers.

For each network type, we compared the results obtained for all proposed configurations, and we selected the best performing configuration, whose results will be presented in Section 5. In those preliminary tests, the 2 post-processing methods mentioned in Section 3.3 were tested and compared with the results obtained without any post-processing. We also compared the use of pre-segmented data and the use of streaming data, for both training and testing, in line with the discussion in Section 3.2. Altogether, we ran a large amount of experiments, corresponding to the parameter settings above. For clarity, in Section 5, we present only the best results obtained in the main configurations (e.g. for each type of ANN). Along the same lines, for the 14-class problem, we only tested one architecture, corresponding to the best results obtained in the 3-class case. This case is detailed in Section 5.3.

4.3. Training algorithms settings

All tested DNNs were trained using the momentum method of the Stochastic Gradient Descent introduced by (Rumelhart, Hinton, & Williams, 1986), applied on batches of size 128 input frames with a momentum of 0.5, without cost regularization and without dropout (preliminary tests showed that no performance improvement was obtained with cost regularization and dropout). Cross-entropy was used as the cost function. The learning rate was set to 0.1 for the first 100 epochs, then decaying by a factor of 0.9 if the error delta was less than 0.1, and the process was completely stopped when the error delta fell below 0.01.

For CNNs, cross-entropy was also used as the cost function with square-norm regularization of 0.001. Weight optimization was done via Stochastic Gradient Descent on batches of 128 input matrices. The stopping criterion was the same as with DNNs.

RNNs used LSTM units and were trained on a minimum of 50 epochs and up to 100 epochs, with a fixed learning rate of 0.05. Two stopping criteria were used; pike in the validation error of more than 5% or in the training error of more than 3%. Cross-entropy was used as the cost function, with both norm-one and square-norm regularization of 0.001. No momentum was used in the gradient descent, and batches were composed of ten 500 frames-long sequences.

Finally, note that no pre-training (e.g. on conventional audio data) was used in this study for all networks. In other words, all our ANNs were trained from scratch on the presented Metro database.

5. Experimental results

5.1. Metrics

The results are reported in the form of confusion matrix, where entry $e_{i,j}$ is the percentage of occurrences from class i classified as class j. The highest numbers in each line of the confusion matrix are displayed in bold, representing the largest proportion of data from class i and showing how it was classified. For the 15-way classification we provide either the "raw" 14-class confusion matrix, or the corresponding 3-class matrix averaged over the 5 background classes, or both. To take into account class imbalance in the averaging process, we weigh the classification score for the composite classes by the actual proportion they represent within the main classes, as provided by Table 1. Formally, let $N_{i(k)}$ denote the total number of tested data frames of shout/speech-related class i in background sound class k. Let $e_{i(k),j(l)}$ denote the percentage of occurrences of shout/speech-related class i in background sound class k classified as shout/speech-related class j in background sound class l. Then, the entries of the average 3-class confusion matrix are given by:

$$\hat{e}_{i,j} = \frac{\sum_k \sum_l N_{i(k)} e_{i(k),j(l)}}{\sum_k N_{i(k)}}. \quad (1)$$

Table 4
Confusion matrix of a DNN with 1 layer of 128 on pre-segmented events after smoothing.

	1	2	3	Total
1: Shout	**78.8**	17	4.2	
2: Speech	28.6	**69.9**	1.5	**74.9**
3: Background	1	14.6	**84.4**	

Table 5
Confusion matrix of a CNN with 1 layer of 128 units, on pre-segmented segments of 50 frames.

	1	2	3	Total
1: Shout	**79.6**	18.4	2.04	
2: Speech	24.4	**75**	0.63	**77.7**
3: Background	0.6	16.6	**82.8**	

5.2. Results for the 3-class problem

Before commenting the scores presented in the different tables, it is important to mention that these represent classification results over one single experiment and should not be considered as averages. On the other hand, in order to attest to the meaningfulness of the scores, we made sure that the variance between multiple experiments did not exceed a few percent.

In this section, we present the results obtained for the 3-class problem. First, we present the best results obtained with DNN architectures, reported in Table 4. The DNN architecture that provided those results is composed of 1 hidden layer of 128 units. It was trained and tested on pre-segmented data, with blocks of $K = 10$ frames. The smoothing algorithm was applied to improve the results. We can see in Table 4 that the correct classification scores are 78.8%, 69.9% and 84.4%, for the shout class, the speech class and the background sound class respectively, with a total accuracy of 74.9%. These numbers are quite satisfactory given the difficulty of the task due mainly to the variability and noisiness of the signals (see Section 2.2). The main confusions come from speech classified as background sound (28.6%) and vice-versa (14.6%), which can be explained by the large amount of noise present in a lot of speech sequences, and shouts (or screams) recognized as speech (17%), which was somehow expected given that the boundary between loud speech and moderate shouts can be difficult to define sometimes, even for human listeners. Interestingly, shouts are pretty rarely confused with background sound (4.2%) and the reciprocal result is even better (only 1%). This may indicate that the DNN is able to capture the specificity of shout/scream sounds. Yet, it is interesting to note that those results were obtained with a single-layer DNN, i.e. adding a second layer with as many units as the first layer did not improve the results, even though the DPR remains acceptable (see Table 2), which gives an idea of the inherent difficulty of discriminating the considered data.

As for CNNs, the best results, presented in Table 5, were obtained with the following architecture: 150 plies using 2D filters of size 8×10, with a layer of 128 hidden units on top of it. It was trained and tested on pre-segmented matrices of $K = 50$ consecutive frames of MFSCs, and the corresponding DPR was 15. Overall, the performance is a little better than that of the best DNN with an improvement in total accuracy of 2.8%: the shout score is slightly better (79.6% vs 78.8%), whereas the background sound score is slightly below (82.8% vs 84.4%). However, the speech score has significantly improved (75% vs 69.9%).

Table 6
Confusion matrix of an RNN with 1 layer of 256 LSTM units on pre-segmented data.

	config i			
	1	2	3	Total
1: Shout	**78.1**	15.9	5.98	
2: Speech	16.2	**82**	1.73	**82.1**
3: Background	0.11	16.8	**83.1**	

Table 7
Confusion matrices of an RNN with 1 layer of 256 trained on pre-segmented or streaming data, and tested with streaming data.

training data:	pre-segmented (config ii)			streaming (config iii)		
	1	2	3	1	2	3
1: Shout	**69.1**	26.7	4.21	**67.7**	24.9	7.4
2: Speech	1.04	**86.5**	12.5	0.6	**85.8**	13.5
3: Background	1.72	27.7	**70.6**	0.4	18.0	**81.5**
Total		**80.5**			**83.4**	

Finally, the best results obtained with RNNs are presented in Table 6, with a network composed of one layer of 256 LSTM units. Those results were obtained with pre-segmented training data and test data, and with no post-processing. The overall accuracy significantly improved, now reaching over 82%. The score for the background sound class (83.1%) is in-between the scores of the best DNN and CNN. However, the scores for shouts and for speech vary differently: the shout score (78.1%) is slightly below those of the best DNN and CNN, but this mild decrease is largely compensated for by a better score for the speech class (82%), with much less confusions with the background sound class (16.2%) compared to DNN and CNN (respectively 28.6% and 24.4%). Overall, the RNN provides the best performances when averaged across the three classes.

All those results were obtained in Configuration i (i.e. with pre-segmented training data and pre-segmented testing data), however, as mentioned in Section 3.2, this configuration is hardly realistic. To remedy this we now assess the performances of the classifiers in two other, more realistic, configurations: using streaming data for testing and pre-segmented data (Configuration ii) or streaming data (Configuration iii) for training. The results are presented for these two new configurations using the best architecture (one layer of 256 LSTM units) in Table 7. As expected, the overall performance decreases compared with Configuration i, from 82.1% to 80.5%. In Configuration ii the scores drop significantly for the background sound class (70.6% vs 83.1%) and the shout class (69.1% vs 78.1%). Surprisingly, the score for speech increases from 82% to 86.5%. In Configuration iii however, the overall performance is slightly better with an improvement from 82.1% to 83.4% in total accuracy. The drop for the background sound class is a lot less obvious than with Configuration ii. (81.5% vs 70.6%), but a little more important for the shout class (67.7% vs 69.1%). Conversely, the score for the speech class is also higher than in Configuration i (85.8% vs 82%). Altogether, those latter results are mitigated: They show a decrease of the score for the main target class of our task (the Shout class), but they also show some robustness of the two other classes when tackling the tricky case of streaming data, leaving room for hope for future improvement of the system.

5.3. Results for the 14-class problem

This section presents the results for the classification of composite classes. Again, based on our previous results, those results are given for a RNN with 1 layer of 256 LSTM units. The results with the RNN being trained and tested with pre-segmented data (Configuration i) are given in Table 8. The results for pre-segmented training data and streaming test data (Configuration ii) are presented in Table 9.

The main takeaway from Table 8 is that 7 sub-classes out of 14 get an accuracy rate over 80%, and 3 get an accuracy rate over 59%. The sub-classes which are not recognized correctly (less than 50% of accuracy) are “Shout+Stand-by” which is misclassified into “Shout+Cruise” 57.4% of the time, “Shout+Arrival” which is misclassified into “Shout+Cruise” 47.3% of the time, “Speech+Compressor” which gets misclassified into “Background+Compressor” and “Speech+Arrival” which gets misclassified into “Background+Arrival” 34.9% of the time. In the first two cases, this means that the Shout-related subclasses produce errors in other Shout-related classes, resulting in a correct classification of the Shout occurrences. However in the other two cases, it appears the environment overshadowed the Speech content of the sound and the Speech occurrences were misclassified as Background.

The averaged results (averaged across background sound classes, see Section 5.1) are shown in Table 10(left) for Configuration i, and in Table 10(right) for Configuration ii. They show that in Configuration i, splitting the environment into several subclasses was beneficial in terms of classification accuracy, both on average (with an increase in total accuracy from 80.5% to 83%) but also for two classes out of three (shout and background), while the negative impact on the third class (speech) was limited to an accuracy decrease of 1.5%. However, in Configuration ii, directly addressing the 3-class problem seemed to be significantly more efficient, since the scores for all classes in Table 10(right) are lower than in Table 7(left), with a decrease in total accuracy from 83.4% to 74.24%. This can be explained by the fact that our double-labeling scheme (into 15 classes) resulted in a finer partition of the audio dataset and therefore in smaller individual events. Thus, when the test dataset was processed in streaming mode, the number of resulting segments comprising cross-event boundaries turned out to be significantly higher than with the 3-class experiment, leading to more difficulty for the classifier. This speaks to the difficulty of the streaming mode for AED in general. More efforts will be put in the future to improve the system in this difficult configuration.

5.4. Limitations

For security reasons we could not record instances of shouts when the train was in a station, which prevented a good collection of data for classes “Shout+Arrival” and “Shout+Departure” (events for class “Shout+Stand-by” were recorder in the warehouse. This resulted in a dataset presenting a high level of imbalance, which mirrors the natural distribution of the classes. Since we are dealing with discriminative systems without a prior distribution of classes, this could be a way for the classifier to deal with the natural imbalance between the classes considered. However it is true that it could be an issue in learning a representation of each class, the risk being that the model ends up not being able to model under-represented classes and ignoring them as a result. It appears to bar the classifier from modeling classes ”Shout+Arrival” in Table 8 and Table 9, resulting in an ’empty output class’, where no input examples were associated to it. For class “Shout+Departure” suggests that the classifier did not suffer from this imbalance (85.7% of accuracy on this class). The important thing is that this environment labeling can improve the performance of the system on the overall 3-way classification, when marginalizing the results from the 14-way classification over the background classes. Now, it appears this issue did not affect results in Configuration i, which improved with the addition of background environment information, as shown in Table 10(left), decreasing performance for Configuration ii only in Table 10(right) when compared to Table 7. This emphasizes the

Table 8
Confusion matrix of an RNN with 1*256 LSTM units on pre-segmented data (i.e Configuration i).

	1(1)	1(2)	1(3)	1(4)	1(5)	2(1)	2(2)	2(3)	2(4)	2(5)	3(1)	3(2)	3(3)	3(4)	3(5)
1(1): Shout + Stand-by	27.8	0	0	**57.4**	0	6.73	5.73	0.62	0.08	0.31	0.54	0	0.85	0	0
1(2): Shout + Departure	0	**85.7**	0	0	0	0	0	0	0	14.3	0	0	0	0	0
1(3): Shout + Comp.	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1(4): Shout + Cruise	0	0	0	**83**	0	1.55	0.14	0	6.78	0	0	0.14	0	8.33	0
1(5): Shout + Arrival	10.1	0.94	0	**47.3**	0	0.47	0.47	1.41	8.67	9.6	0	0	0	0	21.1
2(1): Speech + Stand-by	0.62	0.04	0	0.47	0	**85.8**	1.96	0.16	4.7	0.69	4.36	0.85	0.03	0.13	0.21
2(2): Speech + Departure	0.89	0.18	0.33	0	0	6.04	**59.4**	0.51	0.56	0.58	1.07	23.9	0	2.54	3.97
2(3): Speech + Comp.	0	0	0	0	0	0.49	0	23.6	0	0	2.55	0	**73.4**	0	0
2(4): Speech + Cruise	0	0.27	0	1.8	0	3.12	1.35	0	**66**	0	0.02	3.27	0	23.6	0.55
2(5): Speech + Arrival	0.23	0	0	0.45	0	4.24	2.43	0	6.14	**49.7**	1.21	0.11	0	0.57	34.9
3(1): Background + Stand-by	0	0	0	0	0	10	1.17	0.2	0.85	0	**68.5**	3.3	0.09	11.3	4.5
3(2): Background + Departure	0.38	0	0	0	0	1.12	1.3	0.06	0	0.09	2.39	**85.2**	0	1.51	7.94
3(3): Background + Comp.	0	0	0	0	0	0.28	0	2.49	0	0	11.9	0	**85.3**	0	0
3(4): Background + Cruise	0	0.32	0	0.98	0	0.35	0	0	12.7	0	0.23	2.64	0	**82.5**	0.37
3(5): Background + Arrival	0	0	0	0	0	1.62	1.26	0	0	0.45	0.72	0.72	0	0.45	**94.8**

Table 9
Confusion matrix of an RNN with 1*256 LSTM units trained on pre-segmented data and tested on streaming data (i.e Configuration ii).

	1(1)	1(2)	1(3)	1(4)	1(5)	2(1)	2(2)	2(3)	2(4)	2(5)	3(1)	3(2)	3(3)	3(4)	3(5)
1(1): Shout + Stand-by	9.88	0.27	0	27.2	0	**57.3**	2.91	0	0.03	0.98	0.3	0	0	0.07	1.02
1(2): Shout + Departure	0	5.84	0	**25.6**	0	16.6	15.4	0	1.57	7.3	0	24.6	0	0	3.14
1(3): Shout + Comp.	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1(4): Shout + Cruise	5.93	0.07	0	**68**	0	0	0	0	9.12	0.52	0	1.33	0	14.5	0.52
1(5): Shout + Arrival	6.17	18.2	0	11.6	0	0	5.66	0	0.26	**33.2**	0	0.51	0	0	24.4
2(1): Speech + Stand-by	1.65	0.52	0	3.64	0	**81.4**	0.79	0.04	3.99	1.96	2.89	0.5	0.03	2.42	0.15
2(2): Speech + Departure	0	0.05	0	0	0	10.3	**39.2**	2.24	2.69	0.25	1.57	29.6	1.82	6.26	5.91
2(3): Speech + Comp.	0	0	0	0	0	**43.1**	0	15.4	0	0	8.95	0	32.6	0	0
2(4): Speech + Cruise	0	0.09	0	4.61	0	1.09	1.02	0	**61.2**	0.44	0	0.74	0	30.4	0.36
2(5): Speech + Arrival	0	0	0	0	0	4.92	0.35	0	19.7	**55.5**	0.1	0.1	0	4.09	15.2
3(1): Background + Stand-by	0.43	0	0	0	0	25.7	0.46	0.07	2.39	0.72	**60**	0.87	0.46	3.91	4.97
3(2): Background + Departure	0	0	0	0.18	0	1.93	11.3	0.4	0	0.31	12.1	**58.6**	1.7	4.76	8.75
3(3): Background + Comp.	0	0	0	0	0	26.7	5.44	0	1.04	0	**38.6**	0	28.2	0	0
3(4): Background + Cruise	0	0.1	0	5.06	0	0.29	0.13	0	21.7	0.01	0.01	0.84	0	**71.5**	0.32
3(5): Background + Arrival	0	0	0	0	0	0.99	3.85	0	8.92	7.49	1.76	0.33	0	22.2	**54.4**

Table 10
Merged detection results using pre-segmented events and streaming.

context	pre-segmented (config i)			streaming (config ii)		
	1	2	3	1	2	3
1: Shout	**81.0**	12.3	6.7	**52.4**	35.4	12.2
2: Speech	1.4	**80.5**	18.1	4.3	**76.5**	19.2
3: Background	0.7	10.8	**88.5**	2.7	23.6	**73.7**
Total		**83.0**			**74.24**	

difficulty of the real-life streaming conditions, which brings about even more issues than the classical pre-segmented tasks.

Additionally, we would like to point out the fact that this study was conducted in a single recording context (recording gear, place, environment) and therefore the performance of the final system might change if tested in a different context. In such a recognition/detection task, if the training dataset can not exhibit such variability then the model needs to be adapted to new conditions. To do so, it can be re-trained with a new set of data. Several strategies can be used, either re-training the whole model or only part of it, or leveraging transfer learning methods as suggested in (Pan & Yang, 2010).

6. Conclusion

This paper reported the results obtained by applying three different types of neural networks (DNNs, CNNs and RNNs) to a specific task of scream/shout detection in a real, embedded, (and difficult) public transportation environment. The entire database consisted in a little over an hour of live sound, recorded inside a subway train running its usual course on the Paris Metro network. It was manually labeled with three labels, one of which corresponds to the target abnormal sounds.

A classifier was then trained to recognize those labels and used to classify an incoming stream, in order to detect abnormal situations. Although all models performed virtually equally well on the Background class with around $83 - 84.5\%$ accuracy, and for the Shout class with around $78 - 80\%$, the Speech class is what set them apart with accuracy ranging from 70% for DNNs to 82% for RNN, with CNN performing at 75% (experiments with pre-segmented training and test data). This attests to the temporal structure of speech, and shows that recurrent neural networks are better adapted to recognizing temporally structured sounds.

Other results show that explicitly taking into account the different categories of background sounds (with a two-level data labeling into composite classes) can have a positive impact. However, this was observed on pre-segmented classification performance. In the case of streaming test data, the complexity of the simultaneous "segmentation and classification task" led to more deceiving results and a need to further investigate this issue. The influence of the average length of each sound class (depending on the way the labeling is made) and the way the neural networks react to the transition between sounds, especially for LSTMs, also requires further investigation. Lastly, although it was not addressed in this study, RNNs have different ways of handling sequences regarding the memory reset throughout the entire learning process. This issue is currently being investigated and could affect the way RNNs identify temporal structure.

Other steps towards improvement would be to gather more data or use some data augmentation technique so as to allow for more complexity in the models, as in (Takahashi, Gygli, Pfister, & Gool, 2016). Transfer Learning in the context of audio events can also be beneficial as shown in (Diment & Virtanen, 2017). A hierarchical model might also provide more deftness in dealing with such a complex environment. Additionally, a more acute description of the environment through more specific labels (mechanical noises, sounds related to passengers activity, etc.) would help understand the instantaneous context.

Acknowledgment

The authors would like to thank Sebastien Ambellouis from IFSTTAR who greatly participated in setting up and managing the recording of the database. The recording sessions were supported by research project DéGIV co-funded by“FUI-BPI France” and “Conseil Général de l'Essonne” and by Secur-ED research project funded by European Commission's FP7 program (GA no.261605).

References

Abdel-Hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22*(10), 1533–1545. doi:10.1109/TASLP.2014.2339736.

Affonso, C., Rossi, A. L. D., Vieira, F. H. A., & de Leon Ferreira de Carvalho, A. C. P. (2017). Deep learning for biological image classification. *Expert Systems with Applications, 85*, 114–122. doi:10.1016/j.eswa.2017.05.039.

Baran, R., Rusc, T., & Fornalski, P. (2016). A smart camera for the surveillance of vehicles in intelligent transportation systems. *Multimedia Tools and Applications, 75*(17), 10471–10493. doi:10.1007/s11042-015-3151-y.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Nips*. Vancouver, B.C., Canada: MIT Press.

Bishop, C. M. (1995). *Neural networks for pattern recognition* (pp. 116–161)). Oxford University Press Inc., New York.

Chu, S., Narayanan, S., & Kuo, C. J. (2009). Environmental sound recognition with time-frequency audio features. *IEEE Trans. on Audio, Speech, and Language Processing, 17*(6), 1142–1158. doi:10.1109/TASL.2009.2017438.

Crocco, M., Cristani, M., Trucco, A., & Murino, V. (2016). Audio surveillance: A systematic review. *ACM Comput. Surv., 48*(4). 52:1–52:46 10.1145/2871183

Dennis, J., Tran, H., & Chang, E. (2013). Image feature representation of the subband power distribution for robust sound event classification. *IEEE Transactions on Audio, Speech, and Language Processing, 21*(2), 367–377. doi:10.1109/TASL.2012.2226160.

Diment, A., Cakir, E., Heittola, T., & Virtanen, T. (2015). Automatic recognition of environmental sound events using all-pole group delay features. In *European signal processing conference, Nice, France* (pp. 734–738).

Diment, A., & Virtanen, T. (2017). Transfer learning of weakly labelled audio. In *2017 ieee workshop on applications of signal processing to audio and acoustics (waspaa)* (pp. 6–10).

Eshtay, M., Faris, H., & Obeid, N. (2018). Improving extreme learning machine by competitive swarm optimization and its application for medical diagnosis problems. *Expert Systems with Applications, 104*, 134–152. doi:10.1016/j.eswa.2018.03.024.

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification ? *J. of Machine Learning Research, 15*, 3133–3181.

Foggia, P., Petkov, N., Saggese, A., Strisciuglio, N., & Vento, M. (2016). Audio surveillance of roads: A system for detecting anomalous sounds. *IEEE Transactions on Intelligent Transportation Systems, 17*(1), 279–288. doi:10.1109/TITS.2015.2470216.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics, 36*(4), 193–202. doi:10.1007/BF00344251.

Funahashi, K. (1998). Multilayer neural networks and bayes decision theory. *Neural Networks, 11*(2), 209–213. doi:10.1016/S0893-6080(97)00120-2.

Ganansia, F., Delcourt, V., Pham, Q., Lapeyronnie, A., Baudry, C., Lucat, L., et al. (2011). Audio-video surveillance system for public transportation. *World congress on railway research. Lille, 2011, France*.

Geiger, J. T., & Helwani, K. (2015). Improving event detection for audio surveillance using gabor filterbank features. In *European signal processing conference. Nice, France* (pp. 719–723).

Gerosa, L., Valenzise, G., Tagliasacchi, M., Antonacci, F., & Sarti, A. (2007). Scream and Gunshot detection in noisy environments. In *European signal processing conference. Poznan, Poland* (pp. 1216–1220).

Gish, H. (1990). A probabilistic approach to the understanding and training of neural network classifiers. In *International conference on acoustics, speech, and signal processing,Albuquerque, NM, USA: 3* (pp. 1361–1364). doi:10.1109/ICASSP.1990.115636.

Graves, A., r. Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 ieee international conference on acoustics, speech and signal processing. Vancouver, B.C., Canada* (pp. 6645–6649). doi:10.1109/ICASSP.2013.6638947.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*. (pp. 5-6).

Hata, T., Kuwahara, N., Nozawa, T., Schwenke, D. L., & Vetro, A. (2005). Surveillance system with object-aware video transcoder. In *2005 ieee 7th workshop on multimedia signal processing* (pp. 1–4). doi:10.1109/MMSP.2005.248636.

He, G., Chen, Q., Jiang, D., Lu, X., & Yuan, Y. (2017). A double-region learning algorithm for counting the number of pedestrians in subway surveillance videos. *Engineering Applications of Artificial Intelligence, 64*, 302–314. doi:10.1016/j.engappai.2017.06.017.

He, H., & Garcia, E. (2009). Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng., 21*(9), 1263–1284. doi:10.1109/TKDE.2008.239.

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., et al. (2017). Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 131–135). doi:10.1109/ICASSP.2017.7952132.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput., 9*(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735.

Huang, W., Chiew, T.-K., Li, H., Kok, T. S., & Biswas, J. (2010). Scream detection for home applications. In *Ieee conference on industrial electronics and applications. Taichung, Taiwan* (pp. 2115–2120). doi:10.1109/ICIEA.2010.5515397.

Janvier, M., Alameda-Pineda, X., Girin, L., & Horaud, R. (2012). Sound-event recognition with a companion humanoid. In *Ieee-ras international conference on humanoid robots (humanoids). Osaka, Japan* (pp. 104–111).

Janvier, M., Alameda-Pineda, X., Girin, L., & Horaud, R. (2014). Sound Representation and Classification Benchmark for Domestic Robots. In *2014 IEEE International Conference on Robotics and Automation* (pp. 6285–6292). Hong-Kong, China: IEEE. doi:10.1109/ICRA.2014.6907786.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intell. Data Anal., 6*(5), 429–449.

Lafay, G., Lagrange, M., Rossignol, M., Benetos, E., & Roebel, A. (2016). A morphological model for simulating acoustic scenes and its application to sound event detection. *IEEE/ACM Transactions on Audio, Speech and Language Processing, 24*(10), 1854–1864. doi:10.1109/TASLP.2016.2587218.

Laffitte, P., Sodoyer, D., Tatkeu, C., & Girin, L. (2016). Deep neural networks for automatic detection of screams and shouted speech in subway trains. In *2016 ieee international conference on acoustics, speech and signal processing. Shanghai, China* (pp. 6460–6464). doi:10.1109/ICASSP.2016.7472921.

LeCun, Y., & Bengio, Y. (1998). The handbook of brain theory and neural networks. *Chapter Convolutional Networks for Images, Speech, and Time Series set as per msp* (pp. 255–258). Cambridge, MA, USA: MIT Press.

Lee, J., Kim, T., Park, J., & Nam, J. (2017). *Raw waveform-based audio classification using sample-level CNN architectures*.

Lei, B., & Mak, M. (2014). Sound-event partitioning and feature normalization for robust sound-event detection. In *Int. conf. on digital signal. processing. Hong Kong* (pp. 389–394).

Lippmann, R. (1994). In V. Cherkassky, J. H. Friedman, & H. Wechsler (Eds.), *Neural networks, bayesian a posteriori probabilities, and pattern classification. From Statistics to Neural Networks: Theory and Pattern Recognition Applications* (pp. 83–104)). Berlin, Heidelberg: Springer Berlin Heidelberg.

McLoughlin, I., Zhang, H., Xie, Z., Song, Y., & Xiao, W. (2015). Robust sound event classification using deep neural networks. *IEEE/ACM Trans. on Audio, Speech, and Language Processing, 23*(3), 540–552. doi:10.1109/TASLP.2015.2389618.

Miao, Y. (2014). Kaldi+pdnn: Building dnn-based ASR systems with kaldi and PDNN.

Nakadai, K., Matsuura, D., Okuno, H. G., & Tsujino, H. (2004). Improvement of recognition of simultaneous speech signals using av integration and scattering theory for humanoid robots. *Speech Communication, 44*(1), 97–112. Special Issue on Audio Visual speech processing doi: 10.1016/j.specom.2004.10.010.

Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H., & Ogata, T. (2015). Audio-visual speech recognition using deep learning. *Applied Intelligence, 42*(4), 722–737. doi:10.1007/s10489-014-0629-7.

Ntalampiras, S., Potamitis, I., & Fakotakis, N. (2009). On acoustic surveillance of hazardous situations. In *Ieee int. conf. on acoustics, speech and signal processing. Taipei, Taiwan* (pp. 165–168).

Orhan Bulan, E. A. B., R. P. L. (2013). Efficient processing of transportation surveillance videos in the compressed domain. *Journal of Electronic Imaging, 22*. 22 – 22 – 15 doi: 10.1117/1.JEI.22.4.041116.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345–1359.

Parascandolo, G., Huttunen, H., & Virtanen, T. (2016). Recurrent neural networks for polyphonic sound event detection in real life recordings. In *2016 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 6440–6444). doi:10.1109/ICASSP.2016.7472917.

Phan, H., Maas, M., Mazur, R., & Mertins, A. (2015). Random regression forests for acoustic event detection and classification. *IEEE/ACM Trans. on Audio, Speech, and Language Processing, 23*(1), 20–31. doi:10.1109/TASLP.2014.2367814.

Pohjalainen, J., Raitio, T., & Alku, P. (2011). Detection of shouted speech in the presence of ambient noise. In *Interspeech. Florence, Italy* (pp. 2621–2624).

Rouas, J.-L., Louradour, J., & Ambellouis, S. (2006). Audio events detection in public transport vehicle. In *Intelligent transportation systems conference. Toronto, Ont., Canada* (pp. 733–738). doi:10.1109/ITSC.2006.1706829.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536.

Salamon, J., & Bello, J. P. (2015). Feature learning with deep scattering for urban sound analysis. In *European signal processing conference. Nice, France* (pp. 729–733).

Sharan, R. V., & Moir, T. J. (2017). Robust acoustic event classification using deep neural networks. *Information Sciences, 396*, 24–32. doi:10.1016/j.ins.2017.02.013. http://www.sciencedirect.com/science/article/pii/S0020025517304553.

SoX (2015). the swiss army knife of sound processing programs. http://sox.sourceforge.net/Main/HomePage.

Takahashi, N., Gygli, M., Pfister, B., & Gool, L. V. (2016). Deep convolutional neural networks and data augmentation for acoustic event recognition. In *Proc. interspeech*.

Valenzise, G., Gerosa, L., Tagliasacchi, M., Antonacci, E., & Sarti, A. (2007). Scream and gunshot detection and localization for audio-surveillance systems. In *Ieee conference on advanced video and signal based surveillance. London,UK* (pp. 21–26). doi:10.1109/AVSS.2007.4425280.

Valero, X., & AlÃas, F. (2012). Gammatone wavelet features for sound classification in surveillance applications. In *European signal processing conference. Bucharest, Romania* (pp. 1658–1662).

Vanneschi, L., Horn, D. M., Castelli, M., & Popovič, A. (2018). An artificial intelligence system for predicting customer default in e-commerce. *Expert Systems with Applications, 104*, 1–21. doi:10.1016/j.eswa.2018.03.025.

Velastin, S. A., Boghossian, B. A., & Vicencio-Silva, M. A. (2006). A motion-based image processing system for detecting potentially dangerous situations in underground railway stations. *Transportation Research Part C: Emerging Technologies, 14*(2), 96–113. doi:10.1016/j.trc.2006.05.006.

Virtanen, T., Mesaros, A., Heittola, T., Plumbley, M., Foster, P., Benetos, E., & Lagrange, M. (2016). *Proceedings of the detection and classification of acoustic scenes and events 2016 workshop (DCASE2016)*. Tampere University of Technology. Department of Signal Processing.

Wang, J., Lin, C., Chen, B., & Tsai, M. (2014). Gabor-based nonuniform scale-frequency map for environmental sound classification in home automation. *IEEE Trans. on Automation Science and Engineering, 11*(2), 607–613.

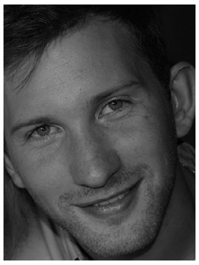
Wang, Y., & Metze, F. (2017). A first attempt at polyphonic sound event detection using connectionnist temporal classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 2986–2990). doi:10.1109/ICASSP.2017.7952704.

Wei, D., Li, J., Pham, P., Das, S., & Qu, S. (2016). Acoustic Scene Recognition with Deep Neural Networks (DCASE Challenge 2016). *Technical Report*. DCASE2016 Challenge.

Wu, X., Gong, H., Chen, P., Zhong, Z., & Xu, Y. (2009). Surveillance robot utilizing video and audio information. *J. of Intelligent and Robotic Systems, 55*(4), 403–421.

Zhang, G. P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 30*(4), 451–462. doi:10.1109/5326.897072.

Zouaoui, R., Audigier, R., Ambellouis, S., Capman, F., Benhadda, H., Joudrier, S., et al. (2015). Embedded security system for multi-modal surveillance in a railway carriage. *Spie security and defence. Toulouse, France*: 9652. (pp. 96520C–96520C–16)(vol. 9652).



Pierre Laffitte is a post-doctoral researcher at IRCAM in Paris. Research interests span audio signal processing, music signal analysis, deep neural networks and machine learning. He graduated with a PhD from Lille University in 2017 where he worked at IFSTTAR-COSYS-LEOST.

Yun Wang is pursuing a Ph.D. degree in Language Technologies from the Language Technologies Institute (LTI) in the School of Science at Carnegie Mellon University (CMU), USA, with sound event detection as his thesis topic. His research interests also include speech recognition and machine learning. He received a M.Sc. from Carnegie Mellon University in 2012.

David Sodoyer is a Permanent Researcher since 2010 with the LEOST laboratory (Villeneuve d'ascq) at IFSTTAR (the French Institute of Science and Technology for Transport, Development and Networks) which he joined in 2008. His research activity focuses on speech/audio signal processing and machine learning for transport application. He received the M.Sc. and Ph.D. degrees in signal processing from the Institut National Polytechnique de Grenoble (INPG), Grenoble, France, in 2001 and 2004, respectively.

Laurent Girin is a Professor at Phelma (Physics, Electronics, and Materials Department of Grenoble-INP), where he lectures signal processing theory and applications to audio. His research activity is carried out at GIPSA-Lab (Grenoble Laboratory of Image, Speech, Signal, and Automation). It deals with different aspects of speech and audio processing (analysis, modeling, coding, transformation, synthesis), with a special interest in joint audio/visual speech processing and source separation. Prof. Girin is also a regular collaborator at INRIA (French Computer Science Research Institute), Grenoble, as an associate member of the Perception Team. He received the M.Sc. and Ph.D. degrees in signal processing from the Institut National Polytechnique de Grenoble (INPG), Grenoble, France, in 1994 and 1997, respectively.