Computer Networks 196 (2021) 108199

Contents lists available at ScienceDirect

Computer Networks

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

FIWARE based low-cost wireless acoustic sensor network for monitoring and classification of urban soundscape

Pau Arce [a], David Salvo [b], Gema Piñero [a,*], Alberto Gonzalez [a]

[a] Institute of Telecommunications and Multimedia Applications (iTEAM) - Universitat Politècnica de València, Camino de Vera s/n, Valencia, 46022, Spain
[b] Robotnik Automation S.L.L., Calle Ciudad de Barcelona, 3-A, Paterna, Valencia, 46988, Spain

ARTICLE INFO

Keywords:
Acoustic sensor networks
Urban sound classification
FIWARE
Edge computing

ABSTRACT

This work presents a wireless acoustic sensor network (WASN) that monitors urban environments by recognizing a given set of sound events or classes. The nodes of the WASN are Raspberry Pi devices that not only record the ambient sound, but also detect and recognize different sound events. All the signal processing tasks, from the recording to the classification carried out by a convolutional neural network (CNN), are run on Raspberry Pi devices. Due to the low cost of the proposed acoustic nodes, the system exhibits a very high potential scalability. Regarding the underlying WASN, it has been designed according to the open standard FIWARE, thus the whole system can be deployed without the need of proprietary software. Regarding the performance of the sound classifier, the proposed WASN achieves similar accuracy compared to other WASNs that make use of cloud computing. However, the proposed WASN significantly minimizes the network traffic since it does not exchange audio signals, but only contextual information in form of labels. On the other hand, most of the time the class reported by the WASN nodes is the "background" soundscape, which usually contains no event of interest. This is the case when monitoring the soundscape of big avenues, where four events have been identified: "traffic", "siren", "horn" and "noisy vehicles", being the "traffic" class associated to the background soundscape. In this paper, the use of a simple pre-detection stage prior to the CNN classification is proposed, with the aim of saving computation and power consumption at the nodes. The pre-detection stage is able to differentiate the other three relevant sounds from the "traffic" and activates the classifier only when some of these three events is likely occurring. The proposed pre-detection stage has been validated through data recorded in the city of Valencia (Spain), achieving a reduction of the Raspberry Pi CPU's usage by a factor of six.

1. Introduction

According to the United Nations, 55% of the world's population lives in urban areas, but this proportion is expected to rise to 68% by 2050 [1]. Cities are going to face major changes in the near future, thus the developing of smart city solutions will be a key factor in order to carry out successful strategies. In this sense, understanding their environment at any time of day and night, and interpreting the data obtained from sensors deployed over the city will be essential.

A significant source of information to monitor what is happening in a city are the sounds, whose recognition is the goal of the so called *environmental sound classification* (ESC) techniques. ESC is a very active area of research and its applications are numerous, most of them focused on urban, nature and domestic environments [2–4]. The systematic approach to ESC can be considered to have started with the challenge on *Detection and Classification of Acoustic Scenes and Events* (DCASE) organized in 2012-2013 by Stowell *et al.* [4]. The first survey on environmental sound recognition techniques was presented by Chachada and Kuo in 2014 [2], while the ESC dataset released by Piczak in 2015 [3] certainly contributed to accelerate the research in the field.

Acoustic sensor networks, mostly wireless (WASN), are usually deployed to capture the urban sounds and classify them [5–7]. Generally speaking, part of the processing carried out by the ASNs is done *in situ* at the node, but the decision and intelligence tasks are usually carried out by the network by means of cloud or edge computing [6,8,9]. In this paper, a WASN formed by smart sensors is presented, which is able to pour contextualized data into the network [10–12], giving a different perspective from the Internet-of-Things (IoT) vision where the nodes only sense the environment and the intelligence is, in general, provided by the network. The fact that the network nodes perform certain calculations themselves, thus offloading the cloud servers, brings this approach closer to an edge computing solution.

* Corresponding author.
E-mail address: gpinyero@iteam.upv.es (G. Piñero).

https://doi.org/10.1016/j.comnet.2021.108199
Received 11 April 2021; Accepted 25 May 2021
Available online 6 June 2021

The acoustic nodes of the proposed WASN are based on Raspberry Pi devices, whose cost is very low, but whose processing unit is powerful enough to recognize urban sounds by means of a deep convolutional neural network (CNN) classifier [13,14]. Once the classification output is obtained, the node transmits this result together with the required context information through a network based on the FIWARE open standard [15].

FIWARE is an European set of specifications designed to enable the development of smart applications in multiple vertical sectors, especially in smart cities, where infrastructures were initially deployed with proprietary solutions. In order to tackle this, FIWARE was developed within the Seventh Framework Programme (FP7) of the European Commission as part of the Future Internet Public–Private Partnership Programme (FI-PPP) with the aim of creating a smart and open solution for sensor networks whose main characteristics were being open, public and royalty-free. Due to the chosen characteristics of the nodes and the network, the proposed WASN can be easily replicated and deployed, and it exhibits very high scalability. Furthermore, the nodes can be continuously updated with the latest advances on classification and computing since they have not been developed on any specific hardware.

The outline of the paper is as follows: Section 2 describes the different components of the proposed low-cost WASN, including the audio signal processing, the CNN that carries out the sound classification, and the network and server architectures. A brief study on the power consumption of the Raspberry Pi due to the constant execution of the sound classification task is also given. In order to alleviate the high computational burden required by the CNN, an efficient pre-detection stage is proposed in Section 3. This stage activates the classifier only when some event of interest is likely to occur. Experimental results are also shown for the detection of events related to the soundscape of three big avenues of Valencia, within the framework of Valencia as a smart city [16,17], obtaining a reduction by a factor of six compared to the original computational load. Finally, Section 4 outlines the main conclusions of the paper.

2. WASN for urban sounds

Fig. 1 illustrates the different tasks carried out by the proposed WASN, from sound recording till data visualization through a web application. They will be described in detail throughout this section. The "Acoustic sensor node" box of Fig. 1 represents all the tasks performed by the Raspberry Pi in the WASN. In Section 2.1 the feature extraction and the sound event classification tasks are described, whereas details on the Raspberry Pi's hardware and its performance will be provided in Section 2.4. The WASN components related to the network and the server are explained in Section 2.2 and Section 2.3 respectively.

2.1. Acoustic sensor node

After recording a sound segment of 3 s, the first task performed by the node is the feature extraction, as it is shown in Fig. 1. In the context of urban sounds, the Mel-spectrogram has been widely used to characterize acoustic events and create a meaningful image to represent them [18,3]. The proposed WASN node follows this approach and characterizes the recorded sound by the Mel frequency cepstral coefficients (MFCCs) [19] computed for each time frame, together with their difference between consecutive frames, known as delta coefficients. The computation is carried out using time frames of 512 samples (23.22 ms given a sampling frequency of 22050 Hz), weighted by a Hanning window of the same duration. Therefore, 128 frames from each 3 s audio recording are obtained. The MFCCs are computed from log-scale Mel-spectrograms covering the frequency range of 0–11.025 Hz divided into 128 bands. Therefore, the input of the CNN is a $128 \times 128 \times 2$ matrix.

The CNN structure is shown in Fig. 2. This CNN structure has been previously proposed in [20,21] as the neural network architecture of the SONYC project. However in this paper, 3 convolutional layers and 2 dense layers are used instead. All convolutional layers have a small kernel of size 5×5 and are followed by a max-pooling layer of size 4×2. A dropout layer is used for the first two convolutional layers to prevent overfitting [21,22]. The activation function of the convolution layers and first dense layer is ReLU, while Softmax is applied for the last dense layer [22].

Regarding the training of the neural network, a stochastic gradient descent method has been used, which is based on adaptive estimation of first and second order moments, also known as Adam optimization [23]. In addition, a crossentropy loss function has been used between labels and predictions. The CNN has been trained with the UrbanSound8K database, freely available at [24], which contains the ten following classes of urban sounds [18]: *air conditioner*, *car horn*, *children playing*, *dog bark*, *drilling*, *gun shot*, *jackhammer*, *siren*, *street music* and *traffic* (equivalent to the *engine idling* class in [18]). The whole dataset has been divided into a training dataset (9 folders of the UrbanSound8K dataset) and a validation dataset (1 folder of the UrbanSound8K dataset), following the methodology described in [18].

In an initial stage, the CNN was trained in a personal computer using TensorFlow [25], and afterwards, the CNN was copied in the Raspberry Pi through the TensorFlow Lite Converter. As it can be seen in Fig. 2, the output of the CNN is a vector formed by 10 float numbers that indicate the probability of each class. The detected class is chosen as the one whose probability is higher. The average classification accuracy was 71%, which is close to the results reported in [18]. Other works have reported better accuracy figures by adding features to the CNN input [14], or by using smaller datasets different from the UrbanSound8K in their experiments [13].

2.2. Network architecture

The information provided at the output of the CNN is modeled and sent to a cloud server for storage and further processing. The acoustic sensors developed on a Raspberry Pi can be connected to the server through different communication technologies. These communication technologies are usually wireless in order to facilitate the installation of nodes in the most suitable places for the sound acquisition, regardless of the availability of wired connection. In addition, these nodes are generally powered using batteries, which makes them much more portable and versatile. The main disadvantage is that the batteries need to be recharged on a regular basis, or, alternatively, the installation of solar energy is required to power the nodes. This is why the optimization of urban sound detection and classification mechanisms is so important, and it is something to which is paid special effort within this work (Section 3). A small consumption saving can mean extending the autonomy of the node during hours or days.

In order to guarantee the connectivity with the server, three type of wireless scenarios can be considered. First of all, the Raspberry Pi can connect via built-in WiFi if it is near to an access point with Internet connectivity. This scenario is recommended in situations where the environment is controlled and WiFi is available. A second scenario can occur when there is no WiFi nearby and long distance communication is necessary. In this case, a 3G/4G dongle can be installed in the Raspberry Pi, allowing the node to communicate directly with the server in the cloud. Finally, in situations where only a set of the WASN nodes are under WiFi or 3G/4G coverage, an ad hoc network of sensors can be used, where at least one of the nodes must have Internet connectivity. In this scenario, the nodes use an ad hoc routing protocol to know the route to the rest of the nodes and, particularly, to the node that acts as a gateway to the Internet. In the proposed WASN, WiFi connectivity has been used, but all three alternatives have been successfully developed and tested. In particular, our ad-hoc network uses the OLSR protocol [26], but different alternatives exist for ad hoc networks and mesh networks [27]. This network architecture can guarantee scalability, although other advanced techniques could be used [28].

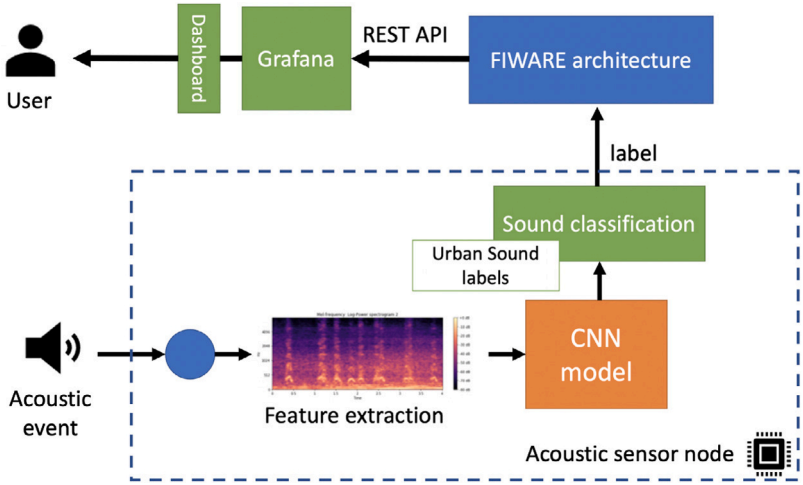

Fig. 1. Block diagram of the tasks performed by the WASN.

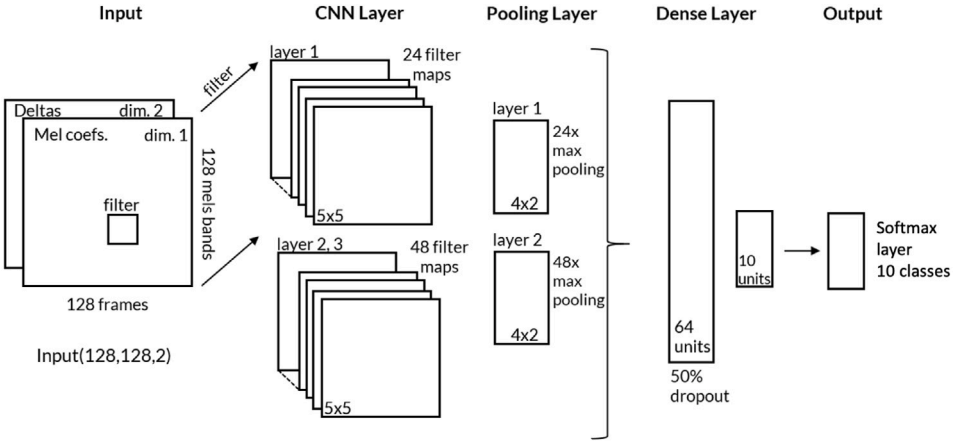

Fig. 2. Structure of the CNN used for urban sound classification.

2.3. Server architecture

On the server side, several components have been deployed to manage the data sent by the acoustic sensor nodes, which essentially means receiving, storing and processing these data, and subsequently representing them through a dashboard. To deploy this infrastructure, the server software is based on FIWARE's IoT middleware [15].

FIWARE is an open platform that provides a set of tools, components and libraries called Generic Enablers (GEs) that offer reusable functionalities for the creation of new application and Internet services. One of the key aspects of FIWARE is the context management. FIWARE provides mechanisms to generate, collect, publish or query massive context information and use it for applications to react to their context through restful (REST) APIs (Application Programming Interface), which are based mainly on the Next-Generation Service Interface (NGSI) 9/10 data model over HTTP. Although complex scenarios may require off-the-shelf models, FIWARE provides some standard data models for common smart city elements and applications, such as streetlights, noise detection and air quality observations, among others. In this case, the generic *Device* model has been used for the data structure of each Raspberry Pi, following the data model standard recommendation proposed by the Schema community [29].

The Context Broker GE is the FIWARE component that is in charge of creating, deleting, retrieving, and updating entities in the platform. This module acts as a broker, by allowing external systems (consumers) to make data subscriptions with specific rules to the entities and attributes, which improves the architectural distribution and scalability. The Context Broker sends notifications to these consumers when subscription rules are observed. The last values of information sent by the entities are stored in the attached MongoDB database as context information.

However, the Context Broker is not envisaged to be used for storing historical data. For this purpose, another FIWARE GE has been used additionally, with the ability to store historical data in a time series database (TSDB), the QuantumLeap GE, which uses CrateDB as the underlying database. The QuantumLeap component is subscribed to updates on entities in the Context Broker, so any new incoming data is stored for later access, analysis and representation.

The whole server architecture is depicted in Fig. 3. Once the CNN provides the classification of a sound event in the acoustic node, this information is sent to the Context Broker using any of the wireless communication mechanisms explained in Section 2.2 and stored in the TSDB, especially designed for historical data and time range queries and operations.

Therefore, the data flow in the proposed architecture involves three essential components: context producer, context broker and context consumer. As aforementioned, the WASN acts as context producer, recording audio samples and performing classification operations. FIWARE provides the context broker and all the components required to store historical data. Finally, both context and historical data are consumed by a visualization component, a dashboard.

In order to represent this historical data in an intuitive and visual way, Grafana dashboards have been deployed. Fig. 4 illustrates the

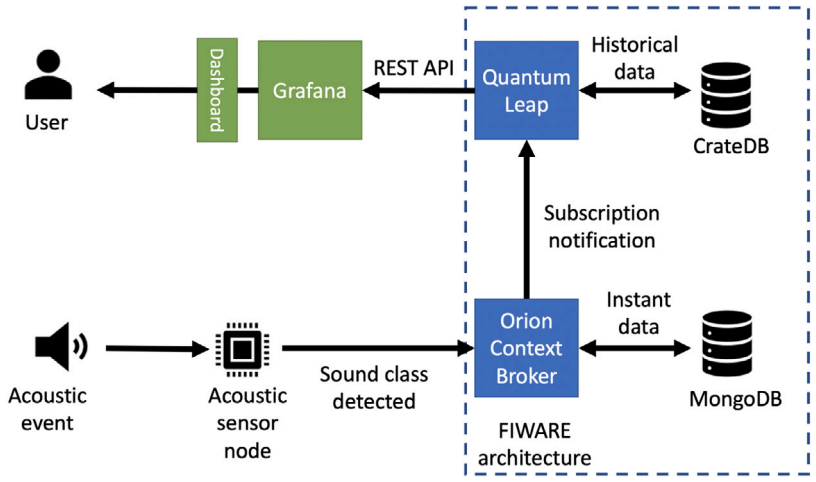

Fig. 3. Block diagram of the proposed server architecture.

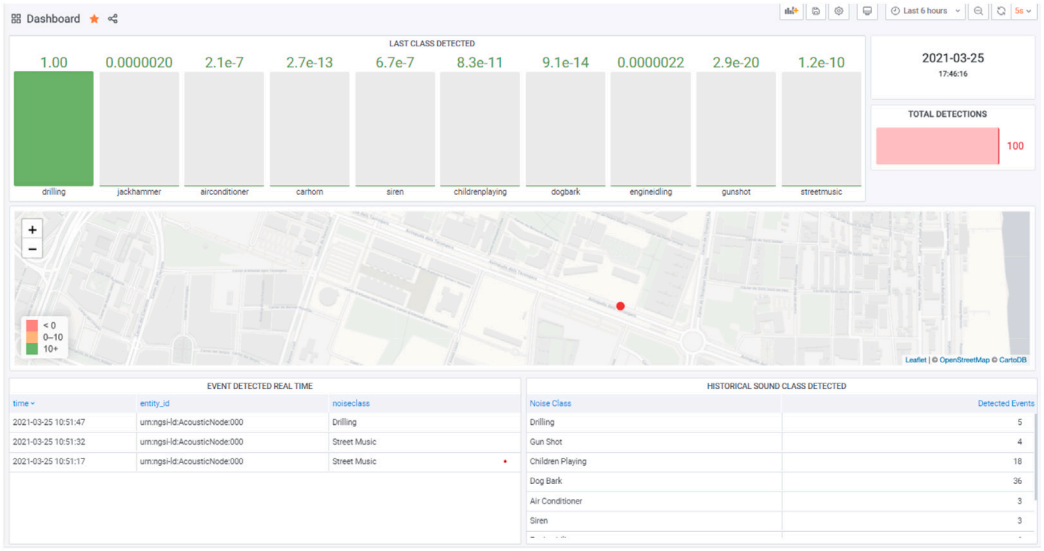

Fig. 4. Dashboard showing event labels and location visualization.

dashboard layout. On the upper side, the probability of the last class detected is depicted among all the available classes, in addition to the total amount of events detected in a given time period. The desired time period can be selected using the dropdown component in the upper-right corner. At the bottom, real time information about the last detected events is presented on the left, and the number of historically detected events per class is shown on the right. Finally, at the center of the dashboard, a map shows the location of the nodes present in the network, using a color-coded scale to depict the sound power currently detected.

Finally, it is worth mentioning that data traffic over the network is minimized since only the vector containing the probability of each class is transmitted by the nodes, instead of the whole audio recording. On the other hand, most of the computation load is carried out at the acoustic node, which is in line with the emerging concept of *edge computing*.

2.4. Raspberry Pi as the acoustic sensor node

This subsection focuses on the tasks regarding the acoustic node depicted in Fig. 1 when implemented on a Raspberry Pi. Specifically, the Raspberry Pi Model B has been used, whose hardware specifications are shown in Table 1. Fig. 5 shows a Raspberry Pi (inside a protective case) connected to a battery (below) and to a miniDSP UMIK-1 microphone covered by a windproof case. The picture on the right was taken during one of the recording sessions described in Section 3.

Table 1
Hardware specifications and total processing time.

Device	Processor	RAM	Time
Dell XPS 15 (2015)	I7-6700HQ 2.60 GHz	16 GB	0.195 s
Raspberry Pi 3 Model B	Quad Core 1.2 GHz	1 GB	2.147 s

The processes shown in Fig. 1 (recording, feature extraction and classification) have been programmed using Python libraries for the sound recording and feature extraction, and Tensorflow and Keras for the implementation of the neural network. The same environment has been implemented on a personal computer (PC) whose hardware specifications are shown in Table 1. The neural network has been trained in the PC and tested on the PC and on the Raspberry Pi.

The time required to perform the whole recording and classification processing is also shown in Table 1 for both devices. Notice that this processing time is the time to record a 3 s audio file and classify it. The processing time shown in Table 1 does not include the 3.089 s needed to record and save the sound in a *wav* file, since it takes the same time

Fig. 5. On the left side: Acoustic node formed by a Raspberry Pi encapsulated in a protective case and connected to a battery (below) and a microphone. On the right side: The acoustic node placed at one of the sites described in Section 3.

Table 2
Processing time per specific task.

Device	*wav file* loading	Feature extraction	CNN classification
Dell XPS 15 (2015)	0.124 s	0.012 s	0.059 s
Raspberry Pi 3 Model B	2 s	0.052 s	0.095 s

for both devices. On the other hand, Table 2 shows the processing time devoted to each specific task of the classification stage, including the loading of the audio segment from the *wav* file. It can be observed that the feature extraction task is four times longer on the Raspberry Pi than on the PC, whereas the classification task is almost twice.

As explained in Section 2.3, once the audio recording is classified, the data will be sent to the FIWARE context broker and any subscribed client, including the time series database, will receive the notification of a new sound event. Given the processing times of Table 2, the WASN node is able to provide a sound event classification approximately every 5 s for an observation time of 3 s, which can be considered real-time processing for smart city applications. However, the main drawback of the proposed low-cost acoustic network is its high power consumption due to its constant activity in this operation mode, which means that the WASN could only be deployed at sites with access to the power grid. In order to cope with this problem, Section 3 proposes a sound detection scheme that reduces the power consumption of the Raspberry Pi by reducing the time that the classification stage must be active.

3. Pre-detection stage for traffic noise

As described above, the WASN is constantly recording sound events, classifying them and sending its output to the network, making the Raspberry Pi consume a significant amount of power [30]. On the other hand, an urban environment is usually defined by a certain background sound ("children playing" for an urban park, "traffic" noise for a downtown avenue, etc.), but an event of interest is something that occurs sparsely and is clearly different from the background sound. For instance, a gunshot in a park or an ambulance driving along an avenue. Thus, a detection stage prior to the classifier is proposed in this section in order to detect whether an event of interest is occurring. The goal is to minimize the computation demanded by the Raspberry Pi's CPU, but without losing key information from the environment. For this purpose, a preliminary study has been carried out on the traffic noise recorded at three big avenues of the city of Valencia. From the data observed, an efficient solution have been proposed, which reduces computation while preserving the goal of the WASN, that is, monitoring the environmental sound and reporting an acoustic event.

3.1. Experiment setup and dataset

Acoustic nodes were placed at three big avenues of Valencia as it can be seen in Fig. 6 (also available as a Google map at https://tinyurl.com/yecxvuk9). Location #1 was very close to one of the biggest hospitals of the city in order to obtain enough samples of siren sounds, whereas Location #2 usually suffers from traffic jams in the afternoon. Location #3 is a big avenue where the lights are synchronized and most of the vehicles can circulate at the maximum permitted speed of 50 km/h. At that speed, some cars and motorbikes generate annoying loud noises for the pedestrians. Around two hours of urban sounds were recording at each location in three different days. The same acoustic node and equipment described in Section 2.4 and shown in Fig. 5 were used along the three sessions.

Once all the recorded sounds were analyzed, three main sound events apart from the *traffic* event were identified: vehicles that generate a loud noise when passing, which were labeled as *noisy vehicles*, ambulances and police vehicles with the siren switched on, which were labeled as *siren*, and the sound of car horns, labeled as *horn*. Note that traffic noise is used to refer to any type of noise or sound produced by a vehicle, but the *traffic* class is assigned to the background noise recorded when no horn, siren or noisy vehicle is present. Noisy vehicles are usually motorbikes and cars with damaged mufflers, and vehicles circulating at high speed. Other sounds like birds, people talking or children yelling were also recorded by the nodes, but the study in this paper is focused on these four sound events related to the vehicles passing along an avenue.

After segmenting, selecting and labeling the whole recorded material, a dataset was built, whose number of *wav* files and total time duration for every class are shown in Table 3. A *wav* file is a continuous recording of one or more continuous events of the same class, no matter their duration.

As it can be appreciated from Table 3, most of the time the system of Fig. 1 would detect the *traffic* event, which characterizes the background noise at the locations of Fig. 6. In this sense, 56% of the time the Raspberry Pi would be busy computing the classification of signals that have no interest for the observer. Therefore, a pre-detection stage prior to the classifier is proposed in order to activate the classification process only if one of the noise events other than *traffic* is likely occurring.

Fig. 6. Valencia map with the three locations used for the recording of the new dataset.

Table 3
Sound Event Dataset.

Class	*wav* Files	Observation time
Traffic	56	903.34 s
Noisy vehicles	40	595.04 s
Siren	10	80.16 s
Horn	32	37.91 s

3.2. Design of the pre-detection stage

For the design of the pre-detection stage, firstly the equivalent continuous sound pressure level (SPL) [31], L, in linear units has been calculated for the four classes of Table 3 as

$$L(m) = \sqrt{\frac{\sum_{n\in T_m} x^2(n)}{p_0}}\ , \tag{1}$$

where m denotes the time frame index, T_m is the time interval of the mth frame, and $x(n)$ is the recorded sound $x(t)$ at time $t = nT_s$, being T_s the sampling time. The parameter p_0 is the reference pressure level and is equal to 20 μPa. The duration of T_m has been chosen as 250 ms, and contiguous time intervals overlap 50% of the samples. Fig. 7 shows the values of L (1) for each event along all the recording time. It can be noted that the other three classes present a lower energy compared to the *noisy* class. However, some recorded horn signals also present high levels of energy.

Considering the SPL $L(m)$ as a random variable, their cumulative density function (CDF) has been estimated and a best fit approximation has been stated for each class. The best fit is obtained for a Generalized Extreme Value (GEV) distribution for all the classes. The GEV is often seen as an approximation to model the maximum values of long (finite) sequences of random variables [32]. Moreover, the GEV combines three simpler distributions into a single form, allowing a continuous range of possible shapes. The estimated CDFs of the four events together with their real cumulative sums obtained from the data are shown in Fig. 8. It can be observed that the GEV distribution provides a good fit for every class, although it suffers a slight deviation of 0.2% when fitting the highest values of the *horn* class. For the rest of estimated CDFs, their deviation with respect to the real curve is under 0.07% (mean squared value).

Since the purpose of the pre-detection stage is to activate the classifier when an event different from *traffic* is likely occurring, firstly a threshold on L is proposed in order to detect *noisy* or loud *horn* events. The threshold level has been chosen such that only 10 % of the *traffic* energy is above that level, that is, $L_\gamma = 33.3$ dB. The probability of presenting a higher SPL than L_γ, according to the CDFs of Fig. 8, are 0.6, 0.2 and 0.05 for the *noisy*, *horn* and *siren* events, respectively.

Moreover, in order to avoid annoying fluctuations around the chosen threshold L_γ, an additional condition to activate the classifier has been added: it is required that $L(m) \geq L_\gamma$ during 1 s (equivalent to 7 consecutive time frames). This condition is coherent with the common scene of a *noisy* event, where a vehicle driving at 50 km/h travels a distance of 13.88 m in 1 s. Therefore, it is plausible that the recorded sound level remains high during that period of time when a noisy vehicle is passing by. Fig. 9 shows the SPL of the *noisy* and *traffic* events where the blue line represents the corresponding L values, and the red line overlaps the blue line when both conditions are fulfilled, but takes a constant value of L_γ for those frames that do not. Therefore, the periods of time depicted only in blue would not activate the classifier. The percentage of activation time with respect the whole sequence of the *traffic* event is 4% (36.1 s).

Regarding the pre-detection of the *siren* and *horn* events from the background traffic noise, other parameters have been investigated. Since the sounds produced by sirens and horns usually presents a significant amount of energy in high frequencies, the recorded sound $x(n)$ has been filtered through different high-pass (HP) filters obtaining a new signal denoted by $x_{\mathrm{HP}}(n)$. The resulting $x_{\mathrm{HP}}(n)$ have been analyzed for the three classes, *siren*, *horn* and *traffic*, looking for some discriminating parameter. Finally, the following ratio has been found:

$$R_L(m) = \frac{L_{\mathrm{HP}}(m)}{L(m)}\ , \tag{2}$$

where $L(m)$ was previously computed (1) to discriminate the *noisy* events and $L_{\mathrm{HP}}(m)$ is the equivalent continuous SPL of $x_{\mathrm{HP}}(n)$ expressed as:

$$L_{\mathrm{HP}}(m) = \sqrt{\frac{\sum_{n\in T_m} x^2_{\mathrm{HP}}(n)}{p_0}}\ , \tag{3}$$

where a high-pass infinite-impulse-response (IIR) filter of order two and cutoff frequency 1 kHz has been used to filter $x_{\mathrm{HP}}(n)$. Therefore, the computation of $R_L(m)$ only requires a filtering of order two (involving 3 products and sums per sample), and the computation of $L_{\mathrm{HP}}(m)$ and their division by $L(m)$.

The estimated CDFs of R_L for the *traffic*, *siren* and *horn* classes are shown in Fig. 10. All of them fit a Generalized Extreme Value (GEV) distribution as was the case for $L(m)$. It can be appreciated that the *siren* CDF falls far apart from the CDF of the *traffic* class, while the *horn* CDF is quite close for low values of the ratio $R_L(m)$. At this point we have used the same methodology as for the discrimination of the *noisy* event. A threshold has been set on the SPL ratio (2) such that the value for the *traffic* event is below the threshold with a probability of 0.8. According to Fig. 10 this value is $R_{L,\gamma} = 0.41$.

For this threshold value, it has been tested that the 51.4% of *horn* and the 87.3% of *siren* time frames will activate the classifier,

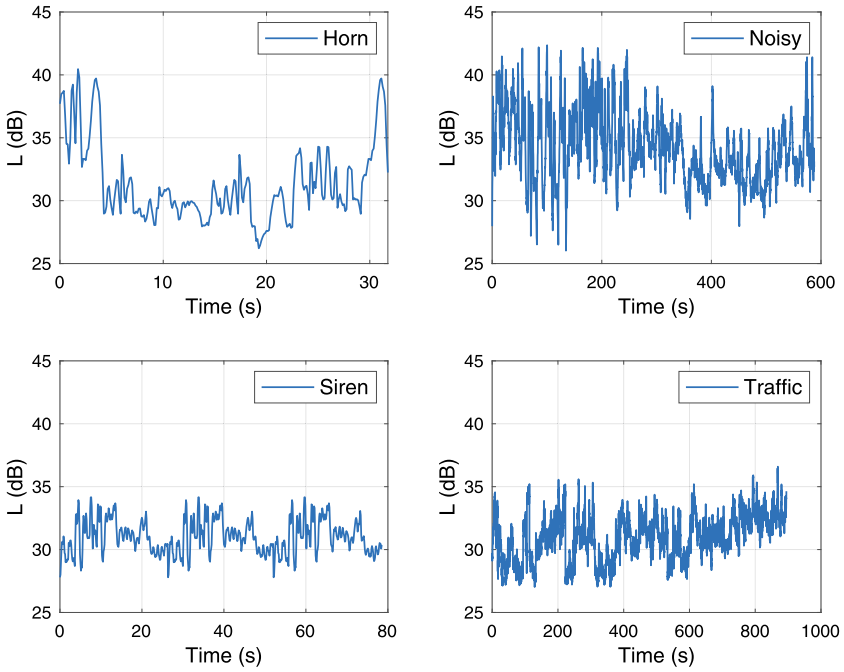

Fig. 7. Equivalent continuous sound pressure level $L(m)$ in dB for the whole recording time of each class.

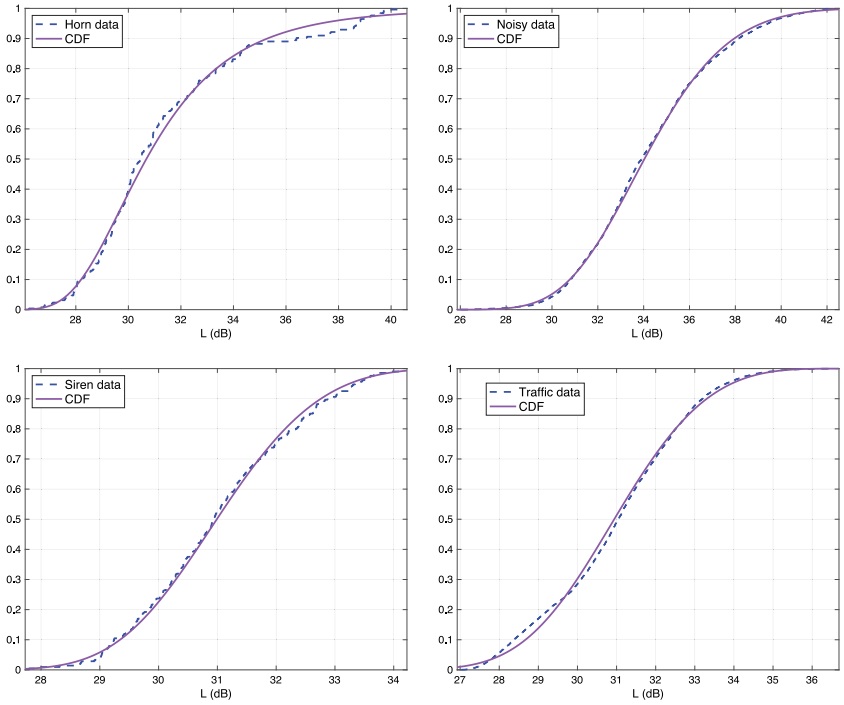

Fig. 8. Estimated CDF (solid line) and real cumulative sum (dashed line) for every class.

in contrast to a 21.2% of *traffic* frames that would trigger it. In this case, no requirement has been set on the number of frames above the threshold due to the impulsive nature of the horn sounds. Fig. 11 shows the SPL of the *siren*, *horn* and *traffic* events where the blue line represents the corresponding L values. The red line overlaps the blue line when the condition $R_L(m) \geq R_{L,\gamma}$ is fulfilled. Therefore, the periods of time depicted only in blue would not activate the classifier.

In order to summarize the computation required by the pre-detection stage, its flow chart is shown in Fig. 12, where the rhomboid blocks are implemented as threshold comparisons. This process depends only on two parameters, thresholds L_γ and $R_{L,\gamma}$, which can be set depending on the interest of the observer. For instance, if it is crucial to detect any ambulance or police car, a value of $R_{L,\gamma} = 0.35$ would trigger the classifier for the 94.6% of the *siren* time frames, although it would also trigger the classifier for the 38.9% of the *traffic* time frames.

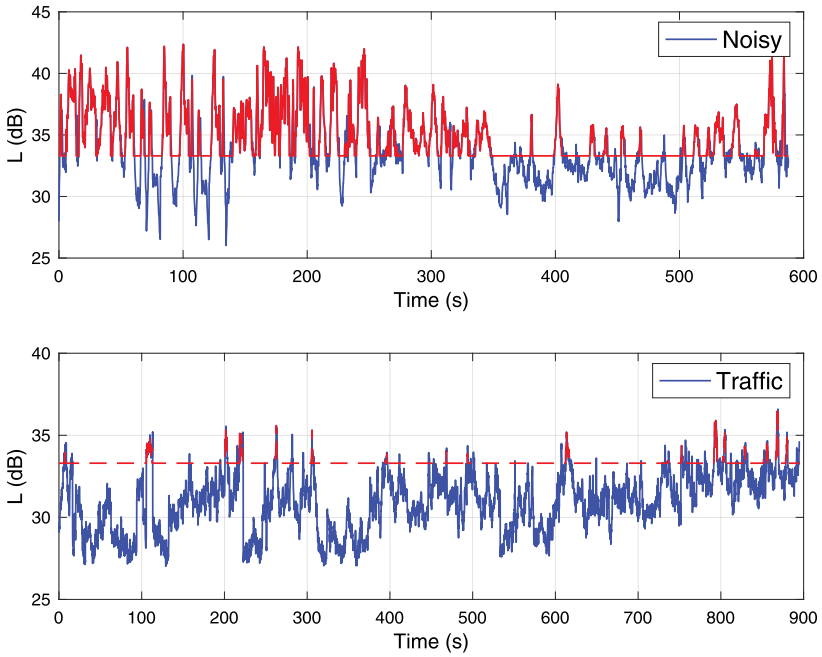

Fig. 9. SPL of the *noisy* and *traffic* events depicted by blue lines. The red lines represent those frames such that $L(m) \geq L_\gamma$ during 7 consecutive time frames. The threshold level of $L_\gamma = 33.3$ dB is represented with a red dashed line as well. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

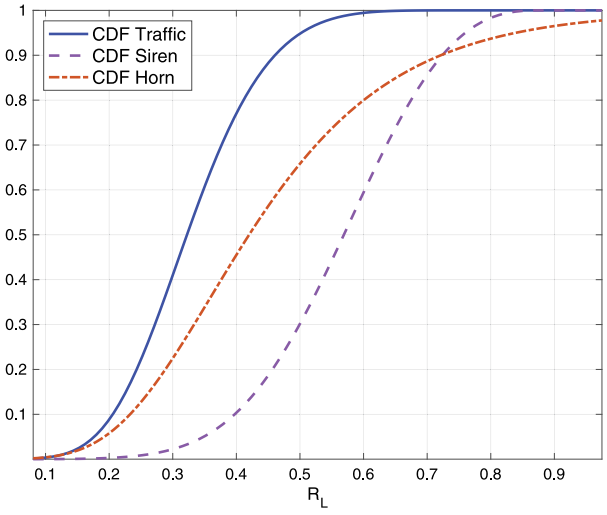

Fig. 10. Estimated CDFs of the ratio $R_L(m)$ for the *traffic*, *siren* and *horn* classes.

3.3. Time processing and power consumption

Once the pre-detection stage has been designed according to Fig. 12, the power consumption has been monitored and compared with the power consumption of the Raspberry Pi's CPU for the whole ESC task (feature extraction plus CNN classification). For this purpose, the Python library *Glances* [33] has been used to obtain the CPU and memory usage information of the Raspberry Pi for the two processes. The refresh rate used in *Glances* is 1 s. Table 4 shows the percentage of use of the CPU and the RAM for the pre-detection and the ESC tasks. It can be seen that the pre-detection process reduces 6 times the use of the CPU compared to the effort required by the ESC.

From the CPU usage data given in Table 4, an approximate power consumption can be calculated by means of previous studies [30]. The CPU usage of 97% corresponds to a 400% CPU load for the Raspberry Pi Quad Core. This means that the ESC task consumes above 3.7 W (730 mA), resulting in an autonomy of approximately 13 h for a battery supplying 10.000 mAh.[1] Given a CPU load of 15% for the pre-detection task, and taking into account that 1.4 W (260 mA) is the

Table 4
Resource Consumption.

	Pre-detection task	ESC task
CPU	15.42%	97.4%
RAM	29.67%	29.84%

[1] The power consumption needed for transmitting the information to the network has not been considered here.

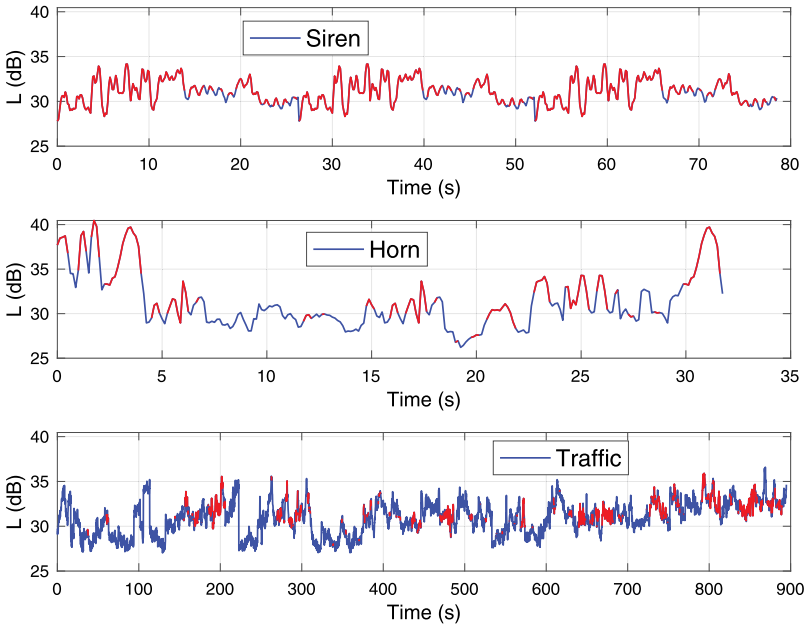

Fig. 11. SPL of the *siren*, *horn* and *traffic* events depicted by blue lines. The red lines represent those frames such that $R_L(m) \geq R_{L,\gamma}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

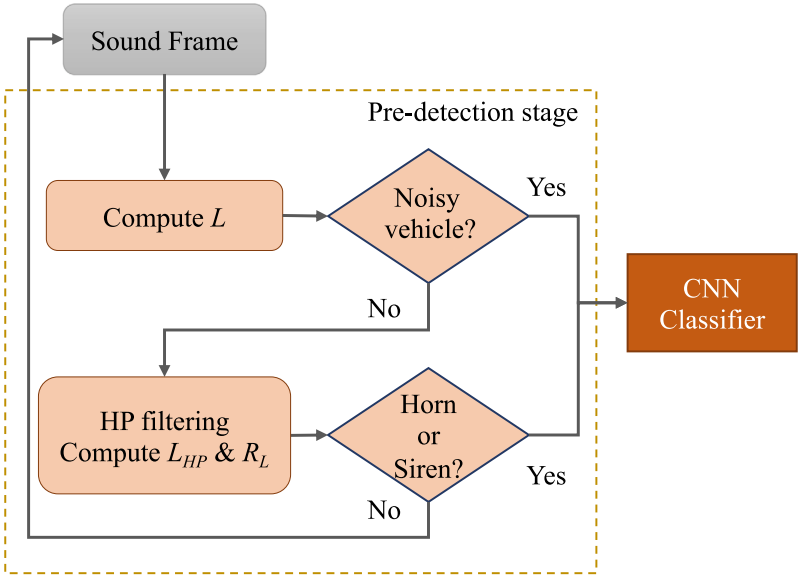

Fig. 12. Flow chart of the computation required at the pre-detection stage.

power consumption at idle state [30], a simple interpolation results in an power consumption close to 1.75 W (330 mA), which means an autonomy of 30 h. This power saving is even higher considering that the node transmits information to the network only when a sound event classification has been carried out.

4. Conclusions

In this paper, a WASN formed by Raspberry Pi devices to monitor and classify urban sounds have been described. The WASN carries out the whole process required to classify urban sound events over the Raspberry Pi, from the feature extraction to the sound classification by means of a convolutional neural network. The WASN transmits the output of the CNN classifier to the cloud server through the FIWARE open standard. Therefore, the whole monitoring system is developed as an open system, from the sound classifier to the network, and it can be replicated in low-cost devices without the need of proprietary software. Although the performance of the ESC is comparable to previous works in terms of accuracy, its main drawback is the high computation load required by the CNN. A constant running of the classifier at the Raspberry Pi makes its CPU to reach an average of 97% load, which results in a huge power consumption of the battery. Therefore, a pre-detection stage has been proposed that activates the classifier only when an event different from the "background sound" event is likely to occur. For this purpose, multiple recordings have been carried out in three big avenues of the city of Valencia, and a pre-detection stage has been designed to detect other sounds different than the *traffic* noise event, as horns, sirens or very loud sounds. In this way, the recorded sound only would enter the classifier if the pre-detection stage decides that it is likely to contain an event. With the use of the pre-detection stage, the CPU usage of the Raspberry Pi is reduced by a factor of six and the battery duration is increased more than twice.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by European Commission (EC) through GA 774477 — MAtchUP, EC together with Spanish Government through RTI2018-098085-B-C41 (MINECO/FEDER) and Generalitat Valenciana through PROMETEO/2019/109. Authors would like to thank Daniel Sanz Montrull for aid in data collection.

References

[1] ONU, The world's cities in 2018, 2018, URL: https://www.un.org/en/events/citiesday/assets/pdf/the_worlds_cities_in_2018_data_booklet.pdf. (Accessed 12 March 2021).
[2] S. Chachada, C.C.J. Kuo, Environmental sound recognition: A survey, APSIPA Trans. Signal Inf. Process. 3 (2014) (2014) 1–15.
[3] K.J. Piczak, ESC: Dataset for environmental sound classification, in: Proceedings of the 23rd ACM International Conference on Multimedia - MM '15, 2015, pp. 1015–1018.
[4] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, M.D. Plumbley, Detection and classification of acoustic scenes and events, IEEE Trans. Multimed. 17 (10) (2015) 1733–1746.
[5] F. Alías, R.M. Alsina-Pagès, Review of wireless acoustic sensor networks for environmental noise monitoring in smart cities, J. Sens. 2019 (2019) 1–13.
[6] M.S. Hossain, G. Muhammad, Environment classification for urban big data using deep learning, IEEE Commun. Mag. 56 (11) (2018) 44–50.
[7] B. Malhotra, I. Nikolaidis, J. Harms, Distributed classification of acoustic targets in wireless audio-sensor networks, Comput. Netw. 52 (13) (2008) 2582–2593.
[8] Z. Sheng, S. Pfersich, A. Eldridge, J. Zhou, D. Tian, V.C.M. Leung, Wireless acoustic sensor networks and edge computing for rapid acoustic monitoring, IEEE/CAA J. Autom. Sin. 6 (1) (2019) 64–74.
[9] L. Luo, H. Qin, X. Song, M. Wang, H. Qiu, Z. Zhou, Wireless sensor networks for noise measurement and acoustic event recognitions in urban environments, Sensors 20 (7) (2020) 2093.
[10] C. Mydlarz, M. Sharma, Y. Lockerman, B. Steers, C. Silva, J. Bello, The life of a New York City noise sensor network, Sensors 19 (6) (2019) 1415.
[11] S. García, D.F. Larios, J. Barbancho, E. Personal, J.M. Mora-Merchán, C. León, Heterogeneous LoRa-based wireless multimedia sensor network multiprocessor platform for environmental monitoring, Sensors 19 (16) (2019) 3446.
[12] J. Segura-Garcia, S. Felici-Castell, J.J. Perez-Solano, M. Cobos, J.M. Navarro, Low-cost alternatives for urban noise nuisance monitoring using wireless sensor networks, IEEE Sens. J. 15 (2) (2015) 836–844.
[13] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, M.D. Plumbley, Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge, IEEE/ACM Trans. Audio Speech Lang. Process. 26 (2) (2018) 379–393.
[14] Y. Su, K. Zhang, J. Wang, K. Madani, Environment sound classification using a two-stream CNN based on decision-level fusion, Sensors 19 (7) (2019) 1733.
[15] FIWARE, FIWARE developer website, 2016, URL: https://www.fiware.org/developers/. (Accessed 12 March 2021).
[16] VLCSmartCity, Valencia smart city website, 2019, URL: http://smartcity.valencia.es/en/. (Accessed 12 March 2021).
[17] MatchUP, MAtchUP - MAximizing the UPscalin and replication potential of high level urban transformation strategies, 2017, URL: https://www.matchup-project.eu/cities/valencia/. (Accessed 12 March 2021).
[18] J. Salamon, C. Jacoby, J.P. Bello, A dataset and taxonomy for urban sound research, in: Proc. ACM Internat. Conf. on Multimedia - MM '14, 2014, pp. 1041–1044.
[19] S. Davis, P. Mermelstein, Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, IEEE Trans. Acoust. Speech Signal Process. 28 (4) (1980) 357–366, http://dx.doi.org/10.1109/TASSP.1980.1163420.
[20] J. Salamon, J.P. Bello, Deep convolutional neural networks and data augmentation for environmental sound classification, IEEE Signal Process. Lett. 24 (3) (2017) 279–283.
[21] K.J. Piczak, Environmental sound classification with convolutional neural networks, in: 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing, MLSP, 2015, pp. 1–6.
[22] T. Lidy, A. Schindler, CQT-based convolutional neural networks for audio scene classification, in: Proc. of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop, DCASE2016, 2016, pp. 60–64.
[23] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980 (2015).
[24] UrbanSound8K, Urbansound data sets, 2014, URL: https://urbansounddataset.weebly.com/. (Accessed 11 April 2021).
[25] Google, TensorFlow: The core open source library to help you develop and train ML models, 2021, URL: https://www.tensorflow.org/. (Accessed 11 April 2021).
[26] P.J.T. Clausen, C. Dearlove, U. Herberg, The Optimized Link State Routing Protocol Version 2, RFC 7181, RFC Editor, 2014, pp. 1–115, URL: https://www.rfc-editor.org/rfc/rfc7181.txt.
[27] T.E. Ali, L.A. Khalil al Dulaimi, Y.E. Majeed, Review and performance comparison of VANET protocols: AODV, DSR, OLSR, DYMO, DSDV & ZRP, in: 2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications, AIC-MITCSA, 2016, pp. 1–6.
[28] A. Balcı, R. Sokullu, Massive connectivity with machine learning for the Internet of Things, Comput. Netw. 184 (2021) 107646.
[29] SchemaOrg, Organization of schemas, 2015, URL: https://schema.org/docs/schemas.html. (Accessed 12 March 2021).
[30] J. Geerling, Power consumption benchmarks, 2015, URL: https://www.pidramble.com/wiki/benchmarks/power-consumption. (Accessed 12 March 2021).
[31] IEC 60268-16 2020, Sound System Equipment - Part 16: Objective Rating of Speech Intelligibility By Speech Transmission Index, Standard, International Electrotechnical Commission (IEC), Geneva, CH, 2020.
[32] E. Bertin, M. Clusel, Generalized extreme value statistics and sum of correlated variables, J. Phys. A: Math. Gen. 39 (24) (2006) 7607–7619.
[33] N. Hennion, Glances - An eye on your system, 2020, URL: https://github.com/nicolargo/glances. (Accessed 12 March 2021).

Pau Arce received his Telecommunications Engineering degree and the M.S. in Telematics from the Universidat Politècnica de València (UPV), Spain, in 2005 and 2007 respectively. In 2014 he obtained his Ph.D. in Telecommunications from the UPV. Currently he works as a researcher at the Institute of Telecommunications and Multimedia Applications (iTEAM). His research interests include multimedia QoS, routing on wireless ad hoc networks and performance evaluation of computer systems.

Gema Piñero received her M.Sc. from the Technical University of Madrid (UPM), Spain, in 1990, and her Ph.D. from the Technical University of Valencia (UPV), Spain, in 1997, both in Electrical Engineering. She is currently a Full Professor at the UPV. Since 1995 she has led or contributed to 30 private and public funded projects on array signal processing for acoustics and communications, active noise control, sound perception and acoustic sensor networks. She has co-authored more than 90 papers in international journals and conferences, serving in the Technical Committee in some of them. She is a member of the Editorial Board of the Digital Signal Processing Journal. From 2013 to 2016 she served in the Spanish Chapter of the IEEE Women in Engineering. She has been a visiting professor at the University of Illinois at Urbana-Champaign (2011) and the Imperial College London (2016). She is a Senior member of the IEEE, member of the EURASIP, and founding member of the Spanish Association for Research and Teaching Universitas.

Alberto Gonzalez received the Graduate degree (highest Hons.) in telecommunication engineering from the Universitat Politecnica Catalunya, Barcelona, Spain, and the Ph.D. degree (magna cum laude) from the Universitat Politècnica de València, Valencia, Spain, in 1997. During 1995, he worked as a Visiting Researcher in the Institute of Sound and Vibration Research, University of Southampton, Southampton, U.K., and is currently heading a research group in audio and communications digital signal processing. He has published more than 100 papers in international technical journals and renowned conferences in the fields of signal processing and applied acoustics and serves as the Dean of the Telecommunication Engineering School since June 2012. His research interests include optimization of computation methods for detection and decoding in digital communications and distributed sound signal processing. He belonged to the EURASIP Special Area Team on Acoustic, Speech and Music Signal Processing.