A Fog-Based Application for Human Activity Recognition Using Personal Smart Devices

FEDERICO CONCONE, GIUSEPPE LO RE, and MARCO MORANA,
University of Palermo, Italy

The diffusion of heterogeneous smart devices capable of capturing and analysing data about users, and/or the environment, has encouraged the growth of novel sensing methodologies. One of the most attractive scenarios in which such devices, such as smartphones, tablet computers, or activity trackers, can be exploited to infer relevant information is human activity recognition (HAR). Even though some simple HAR techniques can be directly implemented on mobile devices, in some cases, such as when complex activities need to be analysed timely, users' smart devices can operate as part of a more complex architecture. In this article, we propose a multi-device HAR framework that exploits the fog computing paradigm to move heavy computation from the sensing layer to intermediate devices and then to the cloud. As compared to traditional cloud-based solutions, this choice allows to overcome processing and storage limitations of wearable devices while also reducing the overall bandwidth consumption. Experimental analysis aims to evaluate the performance of the entire platform in terms of accuracy of the recognition process while also highlighting the benefits it might bring in smart environments.

CCS Concepts: • **Information systems** → **Collaborative and social computing systems and tools**; **Mobile information processing systems**; • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; **Collaborative and social computing devices**;

Additional Key Words and Phrases: Human activity recognition, mobile crowdsensing, fog computing

ACM Reference format:
Federico Concone, Giuseppe Lo Re, and Marco Morana. 2019. A Fog-Based Application for Human Activity Recognition Using Personal Smart Devices. *ACM Trans. Internet Technol.* 19, 2, Article 20 (March 2019), 20 pages.
https://doi.org/10.1145/3266142

1 INTRODUCTION

In recent years, the diffusion of smart mobile devices, such as smartwatches, smartphones, and tablet computers, has enabled new pervasive sensing strategies in which raw data captured by on-board sensors can be analysed to infer high-level knowledge about the user and/or the environment. Mobile crowdsensing (MCS) [17] aims to understand large-scale phenomena by collecting information through a community of individuals [10]. One of its most relevant applications concerns human activity recognition (HAR) in scenarios ranging from health care to urban mobility management, ambient intelligence, and assisted living [24].

Authors' addresses: F. Concone, G. Lo Re, and M. Morana, University of Palermo, Viale delle Scienze, ed. 6, 90128, Palermo, Italy; emails: {federico.concone, giuseppe.lore, marco.morana}@unipa.it.

1533-5399/2019/03-ART20 $15.00
https://doi.org/10.1145/3266142

Sensor-based HAR has been widely addressed in the literature, and most of the proposed solutions use a single mobile device to perform data collection and simple activity recognition [31, 32]. Unfortunately, to perform more intensive tasks, such as real-time classification of *complex* activities, mobile devices with limited resources need to be supported by a solid infrastructure to capture, manage, process, and store data coming from heterogenous sensors.

In this scenario, cloud computing could provide a feasible solution to move heavy computation towards the cloud while using the mobile device as a pure sensing platform. However, all the benefits brought by this approach could be negligible in real-time applications where data are continuously transferred from/to the cloud [5].

In 2012, the *fog* computing paradigm was introduced by Cisco as an extension of the *cloud* computing at the edge of the network. Today, the fog has been widely accepted as a reasonable alternative to the cloud [5] when dealing with large amounts of data that need to be processed locally and timely.

Thus, due to its intrinsic pervasive nature, HAR represents the ideal scenario where fog computing can provide a significant improvement to system performance. The general rule is that the closer a device is to the user, the lower is its computing power. Thus, a fog architecture [45] can be exploited to distribute *data collection*, *analysis*, and *storage* tasks among different devices located at distinct logic levels.

This work includes two contributions. The first contribution is the definition of a fog architecture for complex HAR, in which different devices cooperate to understand the users' behaviour. Data are processed as close as possible to each user. For example, in our case study, the processing units are the users' smartwatches and smartphones, so as to guarantee real-time recognition, whereas a remote cloud infrastructure is responsible for maintaining an overall, consistent view of the whole activity set. By adopting such a general architecture in different application scenarios, the output provided by a single fog devices could be merged with those coming from the users' community to enable more advanced services. For example, elderly people living in a nursing home could be monitored by means of unobtrusive wrist-worn devices (one per user), while a few smartphones (or any other device) owned by the home could be used to perform activity recognition. In such a scenario, data from every user can be processed at the cloud level to define a global *normal* behaviour that can be exploited to reveal warning or danger situations. In wider terms, data coming from the community could be used to support the recognition process itself. For example, if a number of people visit a certain location, GPS data from multiple users could reveal the relationships between an activity and the place where it is performed. This could improve the system performance by limiting the recognition process to some of the most likely activities. For example, if the activity is performed in a urban park, then it will be probably a sport or some kind of dynamic activity. The second contribution of this work is a novel HAR technique that combines three machine learning algorithms—k-means clustering, support vector machines (SVMs), and hidden Markov models (HMMs)—to recognise complex activities modelled as sequences of simple micro-activities.

The remainder of the article is organised as follows. Related work is outlined in Section 2. The system architecture and its deployment in HAR scenarios are described in Section 3. Section 4 provides an in-depth analysis of the activity recognition algorithms. Experimental setup and results are presented in Section 5. Conclusions follow in Section 6.

2 RELATED WORK

In recent years, HAR has become a relevant research area due to its suitability for different application scenarios.

The recognition of human activities has been generally approached focusing on vision or sensor-based solutions. In the first case, video sequences that capture the user's movements and gestures are analysed. This kind of techniques presents some issues [29] that limit their implementation in many real-world scenarios. The first is that video processing techniques are computationally expensive, and thus they can be rarely executed in real time on resource-constrained devices. Moreover, the performance of these systems is strictly dependent on the position of the camera and the appearance of the scene, and therefore the recognition is often limited to indoor environments.

To overcome these limitations, many HAR techniques exploiting sensors directly carried by the users have been presented in the literature. Early solutions were based on acceleration sensors only [41]; however, since a single sensor is not suitable to describe very complex activities, several works proposed to merge information provided by multiple sensors. For example, Bao and Intille [2] present a system that acquires data from five biaxial accelerometers, worn simultaneously on different parts of the body, to recognise both simple and complex activities. The system presented in the work of Lester et al. [30] combines heterogeneous sensors, such as accelerometers and gyroscopes, microphones, and GPS, to improve the recognition performance. Unfortunately, approaches based on wearable sensors are not suitable for real application scenarios due to their intrusiveness [36].

Recent HAR techniques exploit the widespread diffusion of smart devices. Cvetković et al. [12] present a system that aims to improve the quality of life of diabetic patients combining machine learning and symbolic reasoning techniques. Smartphone sensors are used to recognise some activities to trace patients' fatigue while performing their daily routines. Kwon et al. [27] describe an unsupervised learning approach to recognise human activities using smartphone sensors. The recognition process is strictly dependent on the number of clusters chosen during the design phase, and thus distinct activities could be erroneously merged into one, or different instances of the same activity could be seen as unrelated. One of the best-performing HAR frameworks is proposed by Google [18], as it is API level 1. However, these APIs represent a black box, and the developers are not able to use intermediate results as part of their systems, nor to provide any feedback to the activity recognition routine. For this reason, the Google framework can be used only to develop some simple Android applications or as reference for comparing novel activity recognition techniques. In the work of Concone et al. [11], a framework based on smartphone embedded accelerometer and gyroscope sensors for real-time simple activity recognition is presented.

More recently, the focus has moved to the recognition of more complex activities that can be modelled as a composition of simple actions. Ryoo and Aggarwal [43] propose a description-based approach that allows to encode a complex activity through a context-free grammar (CFG), and to model it as an interaction between simpler activities. Similar approaches are used in video-based activity recognition, where a set of silhouettes can be extracted and analysed to describe a particular human activity. For example, in the work of Ogale et al. [35], a probabilistic context-free grammar (PCFG) is built from atomic actions. In the work of Gaglio et al. [16], a Kinect device is used to observe the user, and each activity is modelled as a spatio-temporal evolution of known postures extracted by some joints of the human body.

Some other works exploit probability based-algorithms, such as conditional random fields (CRFs) [28] and HMMs [40], to model complex activities. In the work of Li et al. [31], a framework based on adaptive HMMs is presented. Each complex activity is modelled as a sequence of simple activities performed by the user, and the user's personal experience is considered as *a priori* information to train HMMs. In addition, unlike conventional methods that consider all data from sensors in computational process, such a system proposes an adaptive Viterbi algorithm to speed up the classification.

Lately, the ever-increasing need for measuring large-scale phenomena has encouraged new approaches that aim at analysing data captured from different entities. The MCS system [7, 8] provides some activity recognition and geofencing algorithms that are optimized to meet computational and power constraints of smartphone devices. In particular, the activity recognition sub-system allows to detect three kinds of activities (*walking*, *running*, and *phone still*), while geofencing aims to find and delimit the geographic area where a certain activity, or event, occurs.

To obtain scalable and time-efficient solutions, several works exploit the cloud computing paradigm to provide HAR services according to the most common delivery models, such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Unfortunately, cloud data centres are usually far away from the end devices/users [33], making the development of real-time applications quite critical. Some works focused on combining cloud computing and mobile devices taking the best of both worlds. In the work of Chun et al. [9], a system that allows to run mobile applications on the cloud is described. The basic idea of such work consists in creating and migrating an image of the smartphone to the cloud to perform CPU-intensive tasks on servers that have more resources than a mobile device. Another example is presented in the work of Zhang et al. [49], where a model that permits to decompose a mobile application in several components is proposed. Each component can be run either on a mobile device or migrated to the cloud, so as to overcome any computation or storage constraints.

As mentioned previously, the fog paradigm has been recently adopted in several application scenarios [13, 37] to move data processing close to the point where data are produced, such as by performing resource-expensive computing in lightweight servers placed at the edge of the network. A common scenario addressed by fog computing is the distributed video surveillance, in which traditional client-server architectures would not allow to transmit and analyse huge amounts of video streams efficiently [20].

Just a few applications of fog computing in a HAR scenario have been presented in the literature. CARDAP [23] is a fog-based data analytics platform for supporting MCS applications in a smart city. The main goal of this system is to perform real-time recognition of the citizens' activities by analysing data collected by mobile and Internet of Things devices. Perera et al. [38] describe a general platform addressing three different scenarios (i.e., environmental monitoring, rehabilitation, health) in which wearable sensors are used to measure air quality, a user's movements, and sounds. In such a framework, Internet connected objects (ICOs) are used at the edge of the network, while user smartphones are exploited as intermediate gateways. Wearable sensors are also used in FAAL [47], a fog-based patient monitoring system that traces a user's movements to recognise neurological diseases. In the work of Cao et al. [6], a fog-based platform designed to detect a user's falls in an e-health scenario is described. This system distributes the fall detection task between edge devices and the cloud, allowing lower response time and energy consumption than traditional non-fog approaches.

3 SYSTEM ARCHITECTURE

The system we propose here is based on a three-tier architecture (Figure 1), in which heterogeneous smart devices are exploited to perform tasks of increasing complexity.

At the lowest level, n sensing devices are responsible for collecting rough data and, if required, performing simple data preprocessing/aggregation. The devices at the *sensing layer* do not communicate to each other but share captured data with intermediate devices at the upper layer through low-power network protocols, such as Bluetooth, Zigbee, Z-Wave, and NFC, according to a many-to-one relationship. At this level, since the system processes rough—not sensitive—data, communications are not encrypted, also meeting the computing constraints of the adopted smart devices.

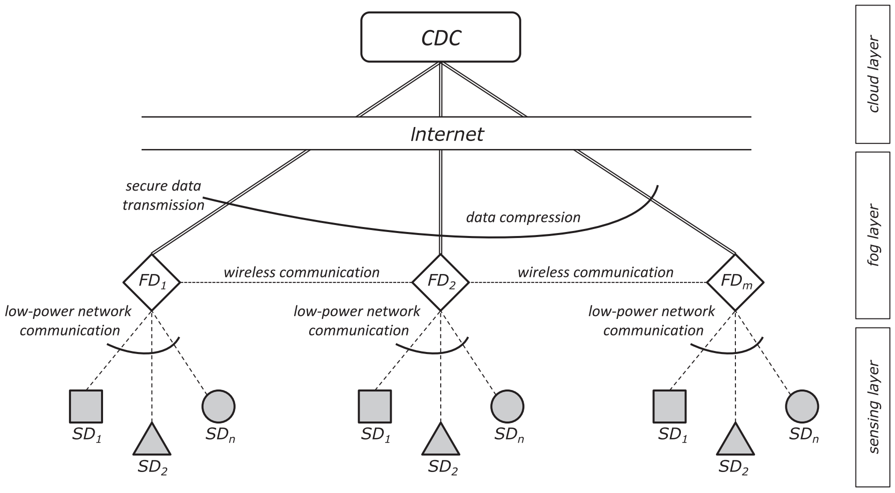


Fig. 1. System architecture. Heterogeneous sensing devices (SD) at the sensing layer are responsible for capturing, pre-processing, and sending data to the fog layer. Here, fog devices (FD) perform data analysis, also exchanging information with other devices at the same layer. At the upmost layer, CDC performs time-consuming processing of data coming from the whole community.

At the intermediate level, *m* fog devices are used to perform in-depth analysis on data obtained from the sensing layer. Fog devices can also exchange information with other devices at the same layer by exploiting more robust wireless technologies, such as WiFi, or GSM/xG cellular networks. Here, data transmission is protected through encryption and authentication techniques that guarantee both data integrity and the user's privacy.

Information produced at the fog layer is sent to a cloud data centre (CDC) that is responsible for resource-consuming analysis of data coming from all the underlying devices. The results of this analysis are stored in the CDC and sent back to the fog devices to update their behaviours, making the whole system consistent. The amount of data exchanged between fog and cloud devices is usually noteworthy, and thus compression algorithms can be applied to improve the transmission efficiency. Moreover, the use of security protocols, both at the network and the transport layer, guarantees information security.

3.1 Application Scenarios

The main purpose of our work is to present a general fog-based architecture that can be adopted to build a distributed HAR application.

A straightforward solution in mobile scenarios could be to exploit the power capability of users' personal devices, such as smartphones, to process data in the fog. Nevertheless, in a multi-user scenario, a single fog device with a higher level of performance, such as a personal computer, can be used to process and integrate data from multiple devices worn by a community of users. We could also consider different situations in which, for instance, the HAR system exploits information directly captured by users' smartphones. Here, these devices would be logically located at the bottom layer of the architecture, while the fog layer could consist of other types of units.

The generality of the architecture we propose allows to use at the fog layer any device with enough computing power to perform raw data analysis and send aggregated data to the cloud. For instance, a HAR system based on video sensors could be implemented by means of RGB/RGBD cameras (sensing layer) sending raw data to some local processing units responsible for performing activity recognition (fog layer), and then to remote storage and synchronisation centres (cloud layer).

Moreover, to carry out the HAR process in more complex scenarios, devices at the fog layer can share information with each other. For example, three situations where fog-to-fog communication can be effective are the following:

- *Alerting*: Monitoring the user's activities in critical environments would allow timely detection of dangerous situations. In such a scenario, the output of the HAR process performed by a fog device could be used to send prompt alerts to other devices at the same layer, thus bypassing the cloud. For instance, in a nursing home or in a factory, the activities can be recognised by means of wrist-worn devices (one per user), while some PCs (e.g., one per environment) can be used at the fog layer. The detection of unexpected behaviours could be immediately notified to other fog devices, without any cloud intervention, enabling a prompt response of the security staff. From an architectural point of view, this can be easily implemented by providing the fog devices responsible for HAR with an additional software module specifically designed to handle the alerting procedures.
- *Distributed and continuous tracking*: The HAR technique we propose aims at recognising complex activities of different duration that can be performed in different places. Some of the activities we considered are made of dynamic (e.g., walking, running) and static (working at a PC) phases. In such a composite scenario, we can imagine a fog layer made of wearable mobile devices (e.g., smartphones) to perform activity recognition during the dynamic phase and stationary devices (e.g., PCs) to continue the recognition once the user reaches a static place (e.g., the office). To this aim, fog devices must be able to share with each other information about the micro-activities performed at a given time, so as to build the overall sequence that describes the complex activity. From an architectural point of view, this can be obtained by providing fog devices with the capability of discovering themselves and pairing to each other automatically.
- *Health promotion*: In a collaborative scenario, devices at the fog layer can interact to motivate the users to achieve a certain result. For instance, if we consider a community of people doing sports in the same place (a gym, a rehabilitation centre, etc.), several fog devices could recognise the activities performed by one or more users and share related information (e.g, elapsed time, speed, calorie consumption, heart rate) with the community to stimulate the users in achieving their goals.

To validate the effectiveness of our solution, we focused on a straightforward case study in which users may wear smart devices while performing some activities. A reasonable assumption in such a scenario could be to exploit smartwatches to track the users' movements, passing heavy data processing to more powerful fog devices, such as the users' smartphones (Figure 2).

To this aim, we designed a novel HAR technique that guarantees complex activities, of different lengths, to be processed timely. The recognition scheme based on k-means, SVMs, and HMMs allows to easily extend the recognition capability of the system (e.g., by including a larger number activities), without the need for redesigning the other components of the system.

With regards to the type of activities to recognise, we considered a set of complex activities that can be reasonably decomposed in simple, atomic micro-activities. For instance, for a given user, the everyday activity *go to work* may consist of a sequence of *walking* for a while, driving or being in a *vehicle* for a certain amount of time, then *walking* again, going up the *stairs*, and finally arriving at the office staying *still*. All of these phases can be traced by the sensors on edge of the network, such as those embedded in the smartwatches. To recognise a set of known micro-activities, each

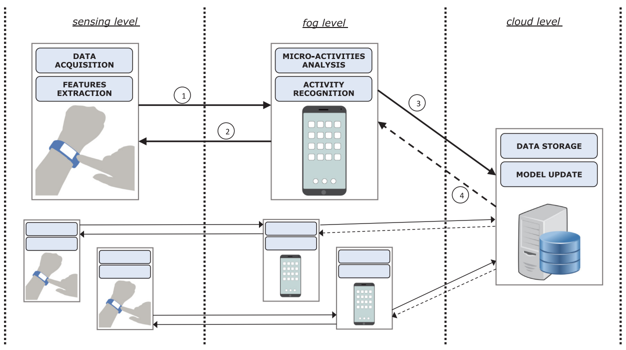


Fig. 2. HAR through users' personal devices. Wrist-worn devices are responsible for capturing sensory data and summarising relevant information. The extracted features are transmitted to the user's smartphone (1), where data are analysed to detect sequences of relevant micro-activities and recognise the complex activity performed by the user (2). Data are temporarily stored in the smartphone and then sent to the cloud (3), where the system parameters and models are updated and sent back to the devices (4).

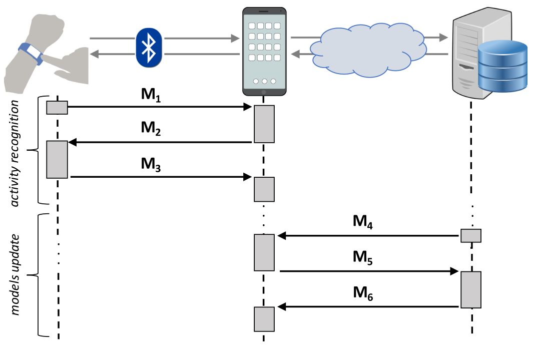


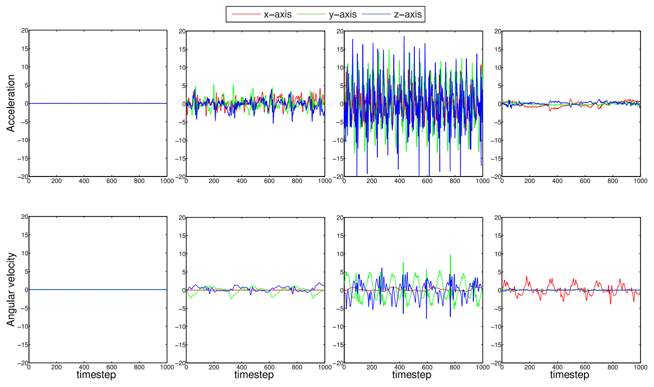
Fig. 3. Information exchanged between the smart devices and the cloud.

sensing device collects data from accelerometer and gyroscope sensors, extracts a feature vector for each time window, and sends the set of feature vectors to the fog devices.

The fog devices of our case study (i.e., the users' smartphones) recognise the performed micro-activity (e.g., *walking*) producing the corresponding *word* of the vocabulary. Fog devices also act as buffers for temporarily storing feature vectors, users' feedback, and any other data that need to be transferred to the cloud. The CDC consists of a server that analyses the different models coming from the smartphones to maintain a unique set of known activities and refines the local models at the fog layer.

The entire dataflow through the layers of the proposed architecture, from the sensing devices to the smartphones to the cloud and back, is summarised in Figure 3.

During the first phase (i.e., the activity recognition), the wrist-worn device creates a message M_1 containing the extracted feature vector and sends it to the smartphone through a Bluetooth connection. Message M_1 is received and parsed by the smartphone that associates a particular word of the dictionary to each feature vector by means of SVMs. This process continues until the smartphone has enough words to build a sequence. Once the sequence of words is completed, the HMM classification is performed and the recognised activity is provided as output. This information, contained in the message M_2, is received by the user on its wrist-worn device. According to the quality of the recognition, the user can give a positive or negative feedback



(a) (b) (c) (d)

Fig. 4. Three-axes acceleration (top row) and angular velocity (bottom row) for *still* (a), *walking* (b), *running* (c), and *in-a-vehicle* (d) behaviours.

through the message M_3. Data collected so far, such as feature sets and user feedback, are stored in the smartphone, ready to be sent to the cloud when requested.

During the second phase, the cloud server sends a message M_4 to check if new data are available. If yes, the smartphone sends the message M_5 containing all data buffered during the previous phase, deleting them from local memory. At this point, the server processes the incoming information so as to evaluate if it is necessary to update the dictionary and the HMM parameters. Once that the models have been updated, a message M_6 is sent to the smartphones to synchronise them.

4 ACTIVITY RECOGNITION MODULES

In this section, an in-depth description of the algorithms behind the activity recognition modules is provided.

4.1 Feature Extraction

This first processing phase focuses on collecting and extracting relevant information (feature vectors) about user's activities. In particular, while a user performs a particular activity, the wrist-worn device collects raw data from embedded accelerometer and gyroscope sensors and sends them to the smartphone for elaboration purposes.

Each activity is roughly characterised by different accelerometer and gyroscope data patterns. Figure 4 shows the values of three-axes acceleration (top row) and angular velocity (bottom row) captured while performing four simple micro-activities: *still*, *walking*, *running*, and *in-a-vehicle*. If we focus on the acceleration values, it is possible to notice how *still* and *in -a-vehicle* activities share a similar pattern, while *walking* and *running* are characterised by high noise being intrinsically associated with a significant user's movement. However, angular velocity values show that *still* and *in-a-vehicle* exhibit distinct patterns, while other activities are generally characterised by oscillations of different width and frequency. To capture all of these characteristics, we decided to combine data from the two sensors.

To ensure real-time activity recognition, the collected input data are processed into fixed-length time windows to extract the features that will be used in the next classification stage. Feature vectors are built similarly to those in Cardone et al. [7], such as by considering [*max, min, mean, standard deviation, root mean square*] values over the three accelerometer and gyroscope axes.

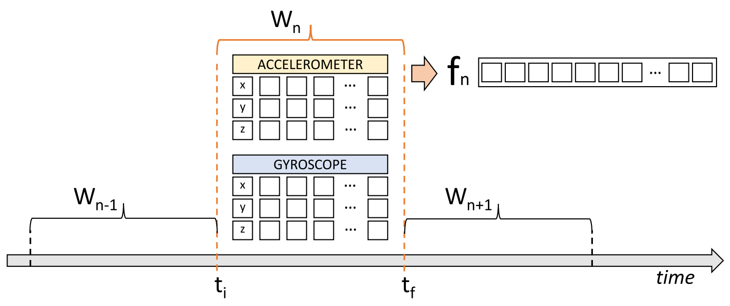


Fig. 5. Feature extraction mechanism. Accelerometer and gyroscope data are processed within the n-th fixed-length time window W_n to obtain the corresponding feature vector f_n.

Therefore, each feature vector f contains 30 elements—for instance, 15 values of acceleration and 15 values of angular velocity (Figure 5).

The size of the processing window also impacts on the performance of the whole system since short windows may improve system performance in terms of execution time and CPU load but may not contain enough information to properly capture the characteristics of the activity. However, long windows may alter the system performance since information about multiple activities performed in sequence might be analysed within a single window. Preliminary experiments were performed considering windows of different lengths. Results showed that fixed-width windows of 3 seconds are the most proper solution [11].

4.2 Vocabulary Construction

The main idea behind the HAR method we propose here is that each feature vector should be able to capture the characteristics of a certain micro-activity ma. Thus, a complex activity CA could be seen as a specific sequence of micro-activities $\{ma_1, ma_2, \ldots, ma_n\}$, each performed within one of the n time windows W_n.

Unfortunately, this representation would make it difficult to recognise complex activities of different lengths, and inefficient to recognise long-lasting activities. For this reason, it would be reasonable to find a unique set of Ω relevant micro-activities $\{ma_1, ma_2, \ldots, ma_\Omega\}$, with $\Omega << n$, that can properly describe every CA. We refer to these Ω elements as *words* of a *vocabulary*.

Under these assumptions, the HAR problem can be modelled as a data association process where observations (feature vectors) are mapped into vocabulary words.

We solve this problem by combining k-means [19] clustering and SVM classifiers [44] to find the set of Ω representative words, and to associate observations with words. These words are then used to train m HMMs, where m is the number of complex activities the system can recognise. This approach, also known as KM-SVM, or CSVM, allows to speed up both the training and the prediction of SVM classifiers on large-scale datasets, and its effectiveness has been discussed in the literature (e.g., [46, 48]).

Given a set of feature vectors $(f_1, f_2, \ldots, f_n)$, k-means partitions the n observations into Ω sets, $C = (C_1, C_2, \ldots, C_\Omega)$, while minimising the intra-cluster error. These clusters are used to create a new training set NT, upon which the SVM model will be trained:

$$NT = \{(C_1, T_1), (C_2, T_2), \ldots, (C_\Omega, T_\Omega)\}, \quad (1)$$

where the i-th pair (C_i, T_i), with $1 < i < \Omega$, represents the cluster and cluster label, respectively.

SVM is a supervised learning technique that aims to find the best separating hyperplane between two classes according to labeled training samples. Generally, given a training set

$X = \{x_1, x_2, \ldots, x_s\}$ and the corresponding label set $Y = \{y_1, y_2, \ldots, y_s\}$, a sample can be expressed as

$$\{x_i, y_i\},\ x_i \in R^d,\ y_i \in \{-1, +1\}, i \in \{1, 2, \ldots, s\}, \tag{2}$$

where d is the dimension of the input space and s is the number of samples. In addition, if we define a and b as the weight vector and the bias of optimal hyperplane, respectively, then the separating function can be expressed as

$$ax + b = 0. \tag{3}$$

According to this definition, all points belonging to the positive class must satisfy the constraint

$$ax_i + b \geq +1, \quad y_i = +1, \tag{4}$$

and the others satisfy

$$ax_i + b \leq -1, \quad y_i = -1. \tag{5}$$

Even though SVMs allow to classify samples belonging to two classes, real-world applications usually require to distinguish between a greater number of classes. Multi-class SVMs overcome this limitation by facing the problem through a series of binary SVMs combined according to some strategies (one-versus-all, one-versus-one, and direct acyclic graph); in most cases, the one-versus-one approach is preferable [15, 21]. Assuming that there are exactly Ω classes, one-versus-one multi-class SVMs train a separate classifier for each different pair of classes creating L SVMs, where $L = \Omega(\Omega - 1)/2$. After all classifiers are trained, the classification is done according to a *max-win voting* approach.

The output of the preceding process described is a set of words $\{ma_1, ma_2, \ldots, ma_\Omega\}$ that combined with each other can be used to model a complex activity. The vocabulary construction is performed on the cloud and is repeated whenever the overall activity models need to be updated. After the new models have been computed—that is, when the vocabulary has been modified, data are sent to all the devices situated in the fog to keep them updated.

4.3 Activity Recognition

The recognition of a new, unknown activity is performed according to a two-step classification procedure. First, pre-trained SVMs are used to associate each feature vector with the corresponding micro-activity (word) contained in the vocabulary. The second step is based on HMMs to model the transitions from one micro-activity to the other.

HMMs [40] are an extension of the Markov chains that aims to find the most probable hidden states according to a sequence of events that can be observed. Unfortunately, in real scenarios, the events are not directly observable, and HMMs overcome this limitation by introducing hidden events that can be considered as causal factors in the probabilistic model.

Formally, an HMM is totally described by the following quintuple, (N, M, A, B, Π), where N is the number of states in the model, M is the number of distinct observation symbols per state, A is the transition probability matrix $\{a_{1,1}, a_{1,2}, \ldots, a_{1,N}, \ldots, a_{N,N}\}$, B is the emission probabilities matrix $\{b_{1,1}, b_{1,2}, \ldots, b_{1,M}, \ldots, b_{N,M}\}$, and Π is the initial probability distribution $\{\pi_1, \pi_2, \ldots, \pi_N\}$, where the generic π is

$$\pi_i = P[S_1 = i], 1 \leq i \leq N. \tag{6}$$

Finally, being $V = \{v_1, v_2, \ldots, v_M\}$ the individual symbols and q_t the generic state at time t, the transition probability matrix A and observation probability B can be written as

$$a_{i,j} = P[S_N = j \mid S_{N-1} = i], 1 \leq i, j \leq N, \tag{7}$$

$$b_j(k) = P[v_k\ at\ t \mid S_j = q_t], 1 \leq j \leq N, 1 \leq k \leq M. \tag{8}$$

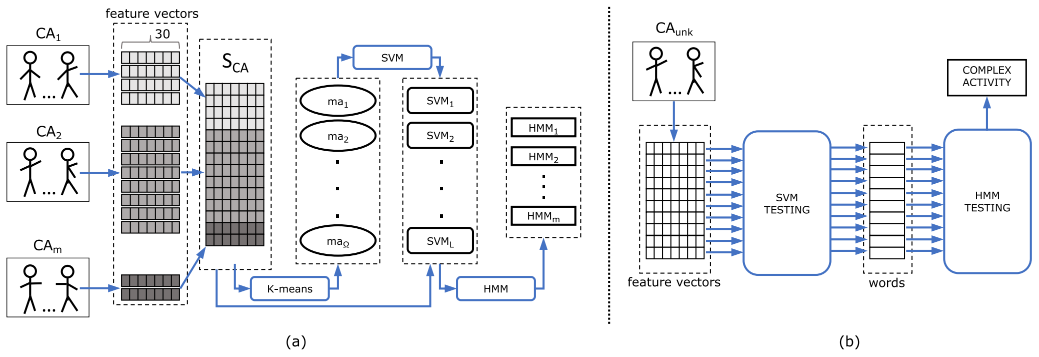


Fig. 6. The activity recognition process during training (a) and recognition (b) phases.

Generally, when HMMs are used to recognise simple activities, the hidden states are the activities themselves and the observations correspond to sensor data [25]. Given a set of micro-activities y, our problem can be modelled as finding the most likely activity w in a set W:

$$argmax_{w \in W} P(w, y). \tag{9}$$

By applying the Bayes' rule, we can rewrite the preceding relation as

$$argmax_{w \in W} P(w, y) = argmax_{w \in W} \frac{P(y|w)P(w)}{P(y)}. \tag{10}$$

Then, the classification of a new, unknown sequence of micro-activities is performed by testing it against all the HMMs and selecting the class associated with the largest posterior probability.

Figure 6 describes the steps of the proposed HAR algorithm, that can be summarised as follows:

— *Training*:
(1) Collect a set S_{CA} containing p repetitions of the m complex activities the HAR system should recognise (note that S_{CA} consists of the feature vectors extracted from raw data).
(2) Apply k-means on S_{CA} to find Ω representative groups of the micro-activities $ma_1, ma_2, \ldots, ma_\Omega$.
(3) Use data from each group (cluster) to train L SVMs that classify the corresponding micro-activity.
(4) Test each feature vector from the original set S_{CA} against the L SVMs to associate each vector to a word, and represent each CA as a sequence of words.
(5) Use these sequences to train m HMMs.

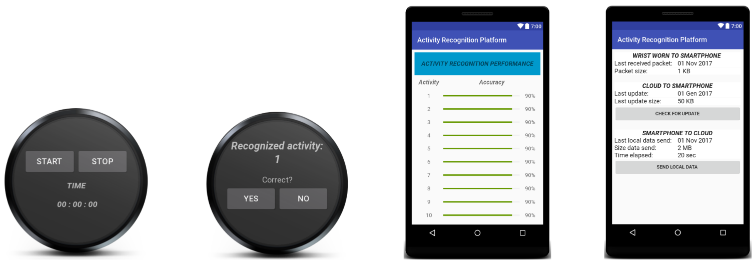
— *Recognition*:
(1) Capture a certain unknown complex activity CA_{unk} performed by the user, and represent it as a sequence of feature vectors.
(2) Use the L SVMs to classify each feature vector, translating it to the corresponding word.
(3) Classify the CA, represented as a sequence of words, by means of m HMMs.

5 EXPERIMENTAL EVALUATION

To evaluate the effectiveness of the proposed architecture and HAR technique, three different set of experiments were performed. The first aimed to find the best values (C, N) in terms of system accuracy and the F-score metric. The second was focused on understanding how the number of observed samples affects the performance of the activity recognition technique. Finally, we

Table 1. Smart Devices Used in the Proposed Case Study

	Smartwatches			Smartphones		
Brand	LG	Samsung	Huawei	Samsung	Samsung	Samsung
Model	W110 G Watch R	Gear S2	Watch 2	Galaxy S2	Galaxy S4	Galaxy S5 Neo
CPU (GHz)	1.2	1.0	1.1	1.2	1.9	1.6
No. of Cores	4	2	4	2	4	8
RAM (MB)	512	768	512	1,024	2,048	2,048



Activity	Accuracy
1	90%
2	90%
3	90%
4	90%
5	90%
6	90%
7	90%
8	90%
9	90%
10	90%

WRIST WORN TO SMARTPHONE

Last received packet:	01 Nov 2017
Packet size:	1 KB

CLOUD TO SMARTPHONE

Last update:	01 Gen 2017
Last update size:	50 KB

SMARTPHONE TO CLOUD

Last local data send:	01 Nov 2017
Size data send:	2 MB
Time elapsed:	20 sec

(a) (b) (c) (d)

Fig. 7. Smartwatch and smartphone Android applications. (a) The overall activity recognition process starts by pressing the *start* button on the wrist-worn interface. (b) Data are processed by the smartphone, and the user is asked about the correctness of the classification. Detailed information about the recognition results (c) and data transmission statistics (d) can be examined through the smartphone-side app.

investigated the impact of data exchange between the several entities involved in the HAR on the overall efficiency of the system.

5.1 Experimental Setup

The experiments were carried out using three different models of Android-based smartphones and three smartwatches equipped with built-in accelerometer and gyroscope sensors (on the left side of Table 1). Two Android applications (one per device type) were developed to perform activity recognition, as described in Section 4, and some supporting tasks, such as data management, compression, and secure transmission. The smartphone application can be installed on any Android device with Ice Cream Sandwich OS or higher, while the smartwatches require at least Jelly Bean OS.

Figure 7 shows four different screens of the Android application. The two leftmost images represent the smartwatch side of the app allowing users to start/stop the activity recognition process, monitor the activity duration (Figure 7(a)), and give feedback about the recognition correctness (Figure 7(b)). The other two images show the smartphone side of the app that permits the user to have in-depth information about the accuracy of the recognition process (Figure 7(c)) and the transmission of collected data (Figure 7(d)).

Note that data processed by the smartphone are temporarily stored in its memory and sent to the CDC when requested. This feature makes the system totally independent of a centralised coordinator, allowing fog devices to perform the various processing tasks in complete autonomy.

Data between smartwatches and smartphones are exchanged through short-range Bluetooth technology, so as to make the proposed architecture compatible with a number of smart devices that do not provide other wireless communication interfaces, such as 802.15.6 and ultra-low power WiFi.

Table 2. Complex Activities Analysed During the Activity Recognition Process

ID	**Activity Name**	**Reference Scenario**
CA_1	*Go To Work 1*	The user goes to work by walking for a while.
CA_2	*Shopping*	The user alternates still and walking phases.
CA_3	*Relax*	The user is sitting for a long time.
CA_4	*Eating*	The user is sitting and moves the hands up and down while eating.
CA_5	*Working at PC*	The user is sitting and types on the PC keyboard or uses the mouse.
CA_6	*Cooking*	The user is cooking briefly moving in the kitchen.
CA_7	*Jogging*	The user alternates running and walking phases.
CA_8	*Go To Supermarket*	The user goes to the supermarket alternating vehicle, walking, and still micro-activities.
CA_9	*Go To Work 2*	The user goes to work alternating walking and vehicle micro-activities.
CA_{10}	*Driving*	The user stays in a vehicle for a long time.

The dataset used in the experiments has been collected, by means of wrist-worn devices, asking 20 volunteers to perform 10 complex activities (CA) in a period of 3 weeks. To collect data in a natural manner, we did not provide the users with instructions on how to perform the activities, while we simply informed the users about the activities we wanted to track and their meanings. The set of complex activities we considered is described in Table 2.

5.2 Activity Recognition Results

The first group of experiments aimed at finding the best set of parameters for the activity recognition procedure—that is, the best pair (Ω, N), where Ω is the number of clusters/words in the dictionary and N represents the number of hidden states in HMMs.

To this purpose, a grid-search approach [4] was applied to measure the system performance in terms of accuracy, precision, recall, and F-score values [14, 39]. Generally, grid search is run with a cross-validation technique, such as K fold cross validation (KFCV) [42] or leave-one-out cross validation (LOOCV) [1]. The basic idea of KFCV is to partition a dataset into K folds to obtain a more realistic assessment of the considered model. Each time, one of the K subsets is used for testing and the other $(K - 1)$ for training. The average error across all iterations provides an estimation of the overall system performance. LOOCV is a special case of KFCV in which the number of folds is equal to the number of points in the dataset.

In this work, we adopted a grid search on Ω and N guided by a KFCV to find the pair that provides the best accuracy and F-score values. The number of folds has been set to 10 so as to minimise the bias (i.e., the difference between estimated and actual accuracy) [42].

Figure 8 shows the results obtained for different iterations of the grid search algorithm on a training set S_1. Since considering accuracy values only (Figure 8(a)) can cause misleading evaluations, the F-score was also computed. Figure 8(b) shows that the best value of the F-score is obtained for $\Omega = 19$ and $N = 14$, that represent the optimal number of words and hidden states to use during the recognition phase. Detailed results of the KFCV for the considered set of complex activities CA are summarised in Table 3.

Once the best pair (Ω, N) has been found, the next set of experiments aimed to evaluate the capability of the system to recognise an activity from an unseen test set S_2, made of different repetitions of the complex activities listed in Table 2. Firstly, 10-fold cross validation on the new test set was performed, and the relative confusion matrix is showed in Table 4. Results show an average accuracy of 78% and an F-score value of 0.72. The confusion matrix also highlights that most of the recognition errors depend on the complex activities CA_8, CA_9, and CA_{10}.

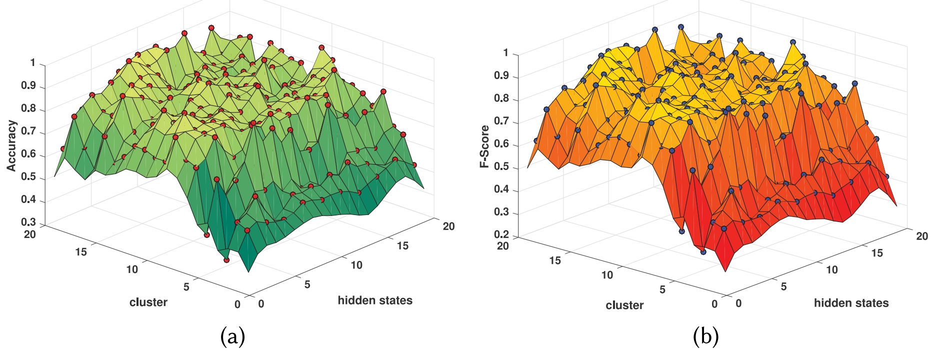


Fig. 8. *K*-fold cross validation. Accuracy (a) and F-score (b) varying the number of clusters $\Omega \in [2, 20]$ and the number of hidden states $N \in [2, 20]$.

Table 3. KFCV Confusion Matrix (Accuracy) for $\Omega = 19$ and $N = 14$

	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	1	0	0	0	0	0	0	0	0	0
CA_2	0	.6	0	0	0	0	0	0	0	.4
CA_3	0	0	1	0	0	0	0	0	0	0
CA_4	0	0	0	1	0	0	0	0	0	0
CA_5	0	0	0	0	1	0	0	0	0	0
CA_6	0	0	0	0	0	1	0	0	0	0
CA_7	0	0	0	0	0	0	1	0	0	0
CA_8	0	.2	0	0	0	0	0	.8	0	0
CA_9	0	0	0	0	0	0	0	0	.8	.2
CA_{10}	0	0	0	0	0	0	0	0	0	1

Table 4. Ten-Fold Cross-Validation Confusion Matrix (Accuracy) for the Testing Phase

	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	.92	0	0	0	0	0	0	0	.08	0
CA_2	0	.8	0	0	0	0	0	.06	0	.14
CA_3	0	0	.86	0	.14	0	0	0	0	0
CA_4	0	0	0	.78	0	.1	.12	0	0	0
CA_5	0	0	0	0	.94	.6	0	0	0	0
CA_6	0	0	0	.12	0	.76	.12	0	0	0
CA_7	0	0	0	.04	0	.22	.74	0	0	0
CA_8	0	.22	0	0	0	0	0	.64	0	.14
CA_9	.14	0	0	0	0	0	0	.04	.66	.16
CA_{10}	0	.04	0	0	0	0	.08	.18	0	.7

A further set of experiments has been performed to measure the system performances while considering different training sets obtained by randomly choosing samples from the set S_2. In *experiment A*, we selected 1/3 of the samples to train the system and the remaining 2/3 for testing; in *experiment B*, 2/3 of the samples were chosen to train the system and the remaining 1/3 for testing; and in *experiment C*, half of the samples were used for training and half for testing. Confusion matrices for each experiment are presented in Tables 5, 6, and 7, respectively. The worst performances are obtained for *experiment A*, in which the mean accuracy is equal to 71% and the F-score to 0.69. Better results can be obtained by increasing the number of training samples as showed in *experiment B* and *experiment C*. In particular, we can notice that when the considered training set is 2/3 of the original set (Table 6), the mean accuracy and F-score are comparable to the values

Table 5. Confusion Matrix: Accuracy for Experiment A (1/3 Training Set and 2/3 Test Set)

	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	.79	0	0	0	0	0	0	0	.12	.09
CA_2	0	.58	0	0	0	0	0	.18	0	.24
CA_3	0	0	.88	0	.12	0	0	0	0	0
CA_4	0	0	0	.74	0	.03	.24	0	0	0
CA_5	0	0	0	0	.97	0	.03	0	0	0
CA_6	0	0	0	0	0	.88	.12	0	0	0
CA_7	0	0	0	0	0	.24	.76	0	0	0
CA_8	0	.09	0	0	0	0	0	.48	0	.42
CA_9	.09	0	0	0	0	0	0	.09	.55	.27
CA_{10}	0	.24	0	0	0	0	.09	.21	0	.45

Table 6. Confusion Matrix: Accuracy for Experiment B (2/3 Training Set and 1/3 Test Set)

	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	.88	0	0	0	0	0	0	.06	0	.06
CA_2	0	.82	0	0	0	0	0	.18	0	0
CA_3	0	0	1	0	0	0	0	0	0	0
CA_4	0	0	0	.94	0	.06	0	0	0	0
CA_5	0	0	0	.06	.94	0	0	0	0	0
CA_6	0	0	0	0	0	.88	.12	0	0	0
CA_7	0	0	0	0	0	0	1	0	0	0
CA_8	0	.12	0	0	0	0	0	.76	0	.12
CA_9	0	0	0	0	0	0	0	0	.83	.17
CA_{10}	0	0	0	0	0	0	0	.18	0	.82

Table 7. Confusion Matrix: Accuracy for Experiment C (1/2 Training Set and 1/2 Test Set)

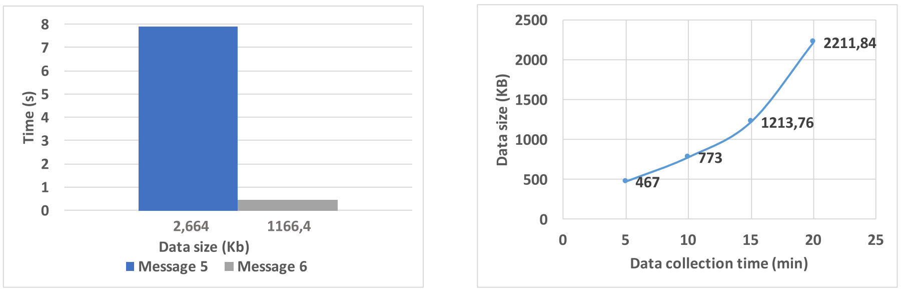
	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	.88	0	0	0	0	0	0	0	0	.12
CA_2	0	.72	0	0	0	0	0	.2	0	.8
CA_3	0	0	.92	0	.08	0	0	0	0	0
CA_4	0	0	0	.76	0	.08	0.16	0	0	0
CA_5	0	0	0	0	.96	0	0	0	.04	0
CA_6	0	.04	0	0	0	.96	0	0	0	0
CA_7	0	0	0	0	0	.08	.92	0	0	0
CA_8	0	.12	0	0	0	.08	0	.64	0	.16
CA_9	0	0	0	0	0	0	0	0	.76	.24
CA_{10}	0	.2	0	0	0	0	0	.12	0	.68

obtained when considering the whole set (i.e., 88%, and 0.9 respectively). Similar considerations can be made for *experiment C*, in which accuracy is equal to 82% (Table 7) and the F-score to 0.79.

This set of experiments revealed that the system performances get worse when S_2 is reduced by a factor of three, while the HAR algorithm still provides a good recognition rate when the original dataset is reduced by 2/3 or 1/2. These last two results underline that the proposed system is able to capture a general model of the activity set.

5.3 Data Transmission

The devices operating within the proposed framework can transmit to each other different kinds of information (e.g., sensor measurements, messages, activity models) and size. In this section, we discuss how data processing and transmission impact the performance of the entire infrastructure, given that as the amount of data to be managed increases, the smartphone's battery life generally reduces [22].



(a) (b)

Fig. 9. (a) Average transmission time for M_5 and M_6. (b) Size of the M_5 while varying the data collection time.

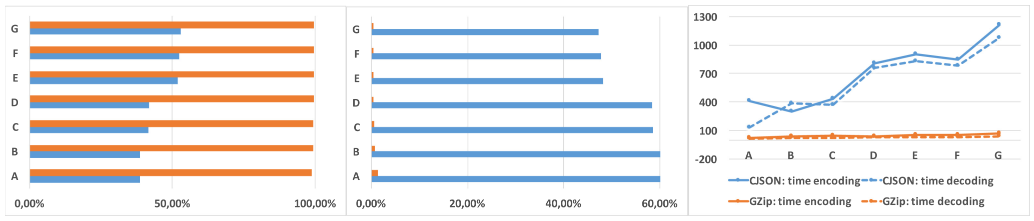
The most widely adopted formats for data exchange in web applications are XML and JSON. In our system, data are JSON formatted to reduce memory occupancy and speed up the transmission as compared to XML [34].

As discussed in Section 3, and shown in Figure 3, six different types of messages are exchanged between the Android clients and the cloud. Some preliminary tests to evaluate the *wrist-worn to smartphone* data transmission were performed during the design of the system. Results showed that the impact of such a transmission on the overall performances is negligible. In particular, accelerometer and gyroscope data can be sent to the smartphone over a Bluetooth connection, with a transfer rate of 25Mbps (approximately 3.125MB/s). Given that tracking an activity produces less than 100Kb per minute, and data are transferred from wrist-worn to smartphone devices every 10 minutes, we can conclude that the transmission of the first four messages can be performed timely without a significant impact on the battery life. However, the transmission of M_5, that contains a set of sensory data and user feedback collected after a particular activity is recognised, and M_6, created when the dictionary and the parameters of the HMMs are updated, could affect the performance of the system.

Figure 9(a) shows the average time needed to transmit the messages M_5 and M_6 using a WiFi connection. Results indicate that the effort to update the activity models (M_6) is quite low, while the transmission of sensory data and user feedback require a noteworthy amount of time. To better investigate this aspect, other experiments were performed so as to determine the relationship between the duration of the data collection process and the size of M_5. Figure 9(b) shows that as the collection time increases, the size of the data transmitted from the clients to the cloud grows very rapidly. This is mainly due to the JSON-formatted messages, that include auxiliary text to generate and parse every pair attribute/value.

To deal with this aspect, the adoption of two lossless compression techniques (i.e., compressed JSON (CJSON) and GZIP [3]) has been considered. The idea behind CJSON is to exploit some redundant information from the original JSON message to obtain a certain level of compression. The GZIP algorithm is a variation of the LZ77 data compression algorithm that includes Huffman coding. Lossless compression allows to significantly reduce the size of the data to be transferred from the fog to the cloud without losing any information that may affect the performance of the HAR system.

Experiments aimed at comparing the effectiveness of the two algorithms in terms of saving percentage (SP) and compression ratio (CR) [26]. Moreover, to evaluate their suitability to the scenario we addressed here, compression and decompression time have also been computed. Assuming that S_M is the original size of the message and S_m the size after compression, then the saving



(a) (b) (c)

Fig. 10. Average saving percentage (a), compression rate (b), and compression/decompression time (c) of the CJSON (blue) and GZIP (orange) algorithms.

percentage, SP, and the compression ratio, CR, can be computed as

$$SP(\%) = \frac{S_M - S_m}{S_M}, \quad CR = \frac{S_m}{S_M}. \tag{11}$$

Tests were run on smartphone devices compressing several messages of various sizes (from 500Kb up to 4Mb), and results are summarised in Figure 10. We can observe that GZIP outperforms CJSON both in terms of saving percentage (Figure 10(a)) and compression rate (Figure 10(b)). Moreover, Figure 10(c) shows that GZIP compression/decompression times measured while increasing the input size (from A to G) are quite lower than those achieved by CJSON. Thus, to contrast the behaviour observed in Figure 9(b), GZIP compression of sensory data is performed to reduce bandwidth usage and enable faster communication between the smartphones and the cloud.

6 CONCLUSION

In this article, we presented a framework for recognising human activities through users' smart devices. The recognition process exploits a fog-based architecture where devices operating at three different logic levels are responsible for collecting sensory data, performing HAR, and maintaining the activity models within the community. Each of these tasks is subject to errors that may impact the overall performance of the system.

Data collection, for instance, is directly controlled by the user through the smart device, by switching on/off the Android application. As a consequence, it frequently happens that initial and final acquisition windows contain noisy data due to the physical interaction between the user and the device. More generally, data within any window could be altered by unintentional movements, leading to the creation of vocabulary words that are not representative of any micro-activity. To deal with this issue, a noise detection algorithm could be introduced to discard "unreliable" windows before the recognition is performed.

With regards to the HAR process, the combined use of KM-SVM and HMMs allows to obtain a compact representation of sequences of any length, and to dynamically change the set of complex activities to be recognised. One limitation of this schema is that sequences not matching one of the trained HMMs will be associated to the *unknown* class. Thus, the system is not currently able to automatically recognise (i.e., to correctly name) new activities that may naturally emerge from the community. A future work could focus on the analysis of the *unknown* set to detect frequent patterns that can be used to train new HMMs on-the-fly.

One of the crucial matters of MCS is still the user's reliability. Since the whole activity recognition is based on data provided by the users, and users' feedback is exploited to drive the model's refinement, as future work we want to extend the framework proposed here to incentivise users' active and reliable participation. To this aim, a trust management module could be included to estimate the user's trustworthiness and discourage malicious behaviours.

REFERENCES

[1] Sylvain Arlot and Alain Celisse. 2010. A survey of cross-validation procedures for model selection. *Statistics Surveys* 4 (2010), 40–79.

[2] Ling Bao and Stephen S. Intille. 2004. *Activity Recognition From User-Annotated Acceleration Data*. Germany, 1–17. DOI: https://doi.org/10.1007/978-3-540-24646-6_1

[3] Timothy C. Bell, John G. Cleary, and Ian H. Witten. 1990. *Text Compression*. Prentice Hall, Upper Saddle River, NJ.

[4] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13 (Feb. 2012), 281–305.

[5] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the Internet of Things. In *Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing (MCC'12)*. ACM, New York, NY, 13–16. DOI: https://doi.org/10.1145/2342509.2342513

[6] Y. Cao, S. Chen, P. Hou, and D. Brown. 2015. FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation. In *Proceedings of the 2015 IEEE International Conference on Networking, Architecture, and Storage (NAS'15)*. IEEE, Los Alamitos, CA, 2–11. DOI: https://doi.org/10.1109/NAS.2015.7255196

[7] Giuseppe Cardone, Andrea Cirri, Antonio Corradi, Luca Foschini, and Dario Maio. 2013. MSF: An efficient mobile phone sensing framework. *International Journal of Distributed Sensor Networks* 9, 3 (2013), 538937. DOI: https://doi.org/10.1155/2013/538937

[8] Giuseppe Cardone, Antonio Corradi, Luca Foschini, and Raffaele Ianniello. 2016. ParticipAct: A large-scale crowdsensing platform. *IEEE Transactions on Emerging Topics in Computing* 4, 1 (2016), 21–32.

[9] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. 2011. CloneCloud: Elastic execution between mobile device and cloud. In *Proceedings of the 6th Conference on Computer Systems (EuroSys'11)*. ACM, New York, NY, 301–314. DOI: https://doi.org/10.1145/1966445.1966473

[10] F. Concone, P. Ferraro, and G. Lo Re. 2018. Towards a smart campus through participatory sensing. In *Proceedings of the 2018 IEEE International Conference on Smart Computing (SMARTCOMP'18)*. IEEE, Los Alamitos, CA, 393–398. DOI: https://doi.org/10.1109/SMARTCOMP.2018.00035

[11] Federico Concone, Salvatore Gaglio, Giuseppe Lo Re, and Marco Morana. 2017. *Smartphone Data Analysis for Human Activity Recognition*. Springer International Publishing, Cham, Switzerland, 58–71. DOI: https://doi.org/10.1007/978-3-319-70169-1_5

[12] Božidara Cvetković, Vito Janko, Alfonso E. Romero, Özgür Kafalı, Kostas Stathis, and Mitja Luštrek. 2016. Activity recognition for diabetic patients using a smartphone. *Journal of Medical Systems* 40, 12 (2016), 256.

[13] A. V. Dastjerdi and R. Buyya. 2016. Fog computing: Helping the Internet of Things realize its potential. *Computer* 49, 8 (Aug. 2016), 112–116. DOI: https://doi.org/10.1109/MC.2016.245

[14] Jesse Davis and Mark Goadrich. 2006. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML'06)*. ACM, New York, NY, 233–240. DOI: https://doi.org/10.1145/1143844.1143874

[15] Kaibo Duan and S. Sathiya Keerthi. 2005. Which is the best multiclass SVM method? An empirical study. *Multiple Classifier Systems* 3541 (2005), 278–285.

[16] S. Gaglio, G. Lo Re, and M. Morana. 2015. Human activity recognition process using 3-D posture data. *IEEE Transactions on Human-Machine Systems* 45, 5 (Oct. 2015), 586–597. DOI: https://doi.org/10.1109/THMS.2014.2377111

[17] R. K. Ganti, F. Ye, and H. Lei. 2011. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine* 49, 11 (Nov. 2011), 32–39. DOI: https://doi.org/10.1109/MCOM.2011.6069707

[18] Google. 2016. Activity Recognition API. Retrieved March 15, 2019 from https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi/.

[19] John A. Hartigan and Manchek A. Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28, 1 (1979), 100–108.

[20] Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. 2013. Mobile fog: A programming model for large-scale applications on the Internet of Things. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC'13)*. ACM, New York, NY, 15–20. DOI: https://doi.org/10.1145/2491266.2491270

[21] Chih-Wei Hsu and Chih-Jen Lin. 2002. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks* 13, 2 (2002), 415–425.

[22] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J. H. Hong, and A. K. Dey. 2012. Factors influencing quality of experience of commonly used mobile applications. *IEEE Communications Magazine* 50, 4 (April 2012), 48–56. DOI: https://doi.org/10.1109/MCOM.2012.6178833

[23] Prem Prakash Jayaraman, João Bártolo Gomes, Hai Long Nguyen, Zahraa Said Abdallah, Shonali Krishnaswamy, and Arkady Zaslavsky. 2014. *CARDAP: A Scalable Energy-Efficient Context Aware Distributed Mobile Data Analytics*

Platform for the Fog. Springer International Publishing, Cham, Switzerland, 192–206. DOI: https://doi.org/10.1007/978-3-319-10933-6_15

[24] Wazir Zada Khan, Yang Xiang, Mohammed Y. Aalsalem, and Quratulain Arshad. 2013. Mobile phone sensing systems: A survey. *IEEE Communications Surveys and Tutorials* 15, 1 (2013), 402–427.

[25] E. Kim, S. Helal, and D. Cook. 2010. Human activity recognition and pattern discovery. *IEEE Pervasive Computing* 9, 1 (Jan. 2010), 48–53. DOI: https://doi.org/10.1109/MPRV.2010.7

[26] S. R. Kodituwakku and U. S. Amarasinghe. 2010. Comparison of lossless data compression algorithms for text data. *Indian Journal of Computer Science and Engineering* 1, 4 (2010), 416–425.

[27] Yongjin Kwon, Kyuchang Kang, and Changseok Bae. 2014. Unsupervised learning for human activity recognition using smartphone sensors. *Expert Systems With Applications* 41, 14 (2014), 6067–6074.

[28] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*. 282–289. http://dl.acm.org/citation.cfm?id=645530.655813.

[29] Oscar D. Lara and Miguel A. Labrador. 2013. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials* 15, 3 (2013), 1192–1209.

[30] Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello. 2006. *A Practical Approach to Recognizing Physical Activities*. Germany, 1–16. DOI: https://doi.org/10.1007/11748625_1

[31] Zhen Li, Zhiqiang Wei, Yaofeng Yue, Hao Wang, Wenyan Jia, Lora E. Burke, Thomas Baranowski, et al. 2015. An adaptive hidden Markov model for activity recognition based on a wearable multi-sensor device. *Journal of Medical Systems* 39, 5 (2015), 57.

[32] Andrea Mannini, Mary Rosenberger, William L. Haskell, Angelo M. Sabatini, and Stephen S. Intille. 2017. Activity recognition in youth using single accelerometer placed at wrist or ankle. *Medicine and Science in Sports and Exercise* 49, 4 (2017), 801–812.

[33] Peter M. Mell and Timothy Grance. 2011. *SP 800-145. The NIST Definition of Cloud Computing*. Technical Report. NIST, Gaithersburg, MD.

[34] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, and Clemente Izurieta. 2009. Comparison of JSON and XML data interchange formats: A case study. *Caine* 2009 (2009), 157–162.

[35] Abhijit S. Ogale, Alap Karapurkar, and Yiannis Aloimonos. 2007. *View-Invariant Modeling and Recognition of Human Actions Using Grammars*. Germany, 115–126. DOI: https://doi.org/10.1007/978-3-540-70932-9_9

[36] Shyamal Patel, Hyung Park, Paolo Bonato, Leighton Chan, and Mary Rodgers. 2012. A review of wearable sensors and systems with application in rehabilitation. *Journal of Neuroengineering and Rehabilitation* 9, 1 (2012), 21.

[37] Charith Perera, Yongrui Qin, Julio C. Estrella, Stephan Reiff-Marganiec, and Athanasios V. Vasilakos. 2017. Fog computing for sustainable smart cities: A survey. *ACM Computing Surveys* 50, 3 (June 2017), Article 32, 43 pages. DOI: https://doi.org/10.1145/3057266

[38] C. Perera, D. S. Talagala, C. H. Liu, and J. C. Estrella. 2015. Energy-efficient location and activity-aware on-demand mobile distributed sensing platform for sensing as a service in IoT clouds. *IEEE Transactions on Computational Social Systems* 2, 4 (Dec. 2015), 171–181. DOI: https://doi.org/10.1109/TCSS.2016.2515844

[39] David Martin Powers. 2011. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Machine Learning Technologies* 2 (2011), 37–63.

[40] L. Rabiner and B. Juang. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine* 3, 1 (1986), 4–16.

[41] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. 2005. Activity recognition from accelerometer data. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence—Volume 3 (IAAI'05)*. 1541–1546. http://dl.acm.org/citation.cfm?id=1620092.1620107.

[42] Juan D. Rodriguez, Aritz Perez, and Jose A. Lozano. 2010. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 3 (2010), 569–575.

[43] Michael S. Ryoo and Jake K. Aggarwal. 2009. Semantic representation and recognition of continued and recursive human activities. *International Journal of Computer Vision* 82, 1 (2009), 1–24.

[44] Bernhard Scholkopf and Alexander J. Smola. 2001. *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA.

[45] Bo Tang, Zhen Chen, Gerald Hefferman, Tao Wei, Haibo He, and Qing Yang. 2015. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In *Proceedings of the 2015 ASE BigData and SocialInformatics Conference (ASE BD&SI'15)*. ACM, New York, NY, Article 28, 6 pages. DOI: https://doi.org/10.1145/2818869.2818898

[46] Van Vo, Jiawei Luo, and Bay Vo. 2016. Time series trend analysis based on K-means and support vector machine. *Computing and Informatics* 35, 1 (2016), 111–127.

[47] J. Vora, S. Tanwar, S. Tyagi, N. Kumar, and J. J. P. C. Rodrigues. 2017. FAAL: Fog computing-based patient monitoring system for ambient assisted living. In *Proceedings of the 2017 IEEE 19th International Conference one-Health*

Networking, Applications, and Services (Healthcom'17). IEEE, Los Alamitos, CA, 1–6. DOI: https://doi.org/10.1109/HealthCom.2017.8210825

[48] Yukai Yao, Yang Liu, Yongqing Yu, Hong Xu, Weiming Lv, Zhao Li, and Xiaoyun Chen. 2013. K-SVM: An effective SVM algorithm based on K-means clustering. *Journal of Computers* 8, 10 (2013), 2632–2639.

[49] Xinwen Zhang, Anugeetha Kunjithapatham, Sangoh Jeong, and Simon Gibbs. 2011. Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. *Mobile Networks and Applications* 16, 3 (June 2011), 270–284. DOI: https://doi.org/10.1007/s11036-011-0305-7

Received December 2017; revised July 2018; accepted August 2018