Transition-Aware Human Activity Recognition Using Smartphones

Jorge-L. Reyes-Ortiz[a,b], Luca Oneto[c], Albert Samá[b], Alessandro Ghio[c,*], Xavier Parra[b], Davide Anguita[a]

[a] *DIBRIS - University of Genova, Via Opera Pia 13, I-16145 Genova, Italy*
[b] *CETpD - Universitat Politècnica de Catalunya, Rambla de l'Exposició, 59-69, 08800 Vilanova i la Geltrú, Spain*
[c] *DITEN - University of Genova, Via Opera Pia 11a, I-16145 Genova, Italy*

Abstract

This work presents the Transition-Aware Human Activity Recognition (TAHAR) system architecture for the recognition of physical activities using smartphones. It targets real-time classification with a collection of inertial sensors while addressing issues regarding the occurrence of transitions between activities and unknown activities to the learning algorithm. We propose two implementations of the architecture which differ in their prediction technique as they deal with transitions either by directly learning them or by considering them as unknown activities. This is accomplished by combining the probabilistic output of consecutive activity predictions of a Support Vector Machine (SVM) with a heuristic filtering approach. The architecture is validated over three study cases that involve data from people performing a broad spectrum of activities (up to 33), while carrying smartphones or wearable sensors. Results show that TAHAR outperforms state-of-the-art baseline works and reveal the main advantages of the architecture.

Keywords: Activity Recognition, Smartphones, Transitions, Support Vector Machines, Machine Learning

*Corresponding author
Email addresses: jorge.reyes.ortiz@smartlab.ws (Jorge-L. Reyes-Ortiz), Luca.Oneto@unige.it (Luca Oneto), albert.sama@upc.edu (Albert Samá), Alessandro.Ghio@unige.it (Alessandro Ghio), xavier.parra@upc.edu (Xavier Parra), Davide.Anguita@unige.it (Davide Anguita)

1. Introduction

Human Activity Recognition (HAR) has nowadays become a prominent research field due to its substantial contributions in human-centered areas of study aiming to improve people's quality of life: Ambient Intelligence, Pervasive Computing and Assistive Technologies [1, 2, 3]. These areas make use of HAR systems as an instrument that provides information about people's behavior and actions [4]. This is commonly done by gathering signals from ambient and wearable sensors and processing them through machine learning algorithms for classification. There are currently many applications where HAR systems are used, for instance, the continuous monitoring of patients with motor problems for health diagnosis and medication tailoring [5], and the automated surveillance of public places for crime prevention [6].

In the past decade, several HAR systems have been proposed and surveyed [7, 8, 9]. They have encompassed multiple activities from different application domains, including locomotion, daily living activities, transportation, sports, and security [10, 11]. Regarding their duration and complexity, activities are categorized in three main groups: short events, basic activities and complex activities. The former group is comprised of brief-duration activities (on the order of seconds) such as postural Transitions (PTs) (e.g. sit-to-stand), and body gestures [8]. Basic activities are instead characterized by a longer duration and can be either dynamic or static (e.g. running, reading) [12]. The latter group, complex activities, is composed of progressions of the aforesaid simpler activities and involve aspects such as interaction with objects and other individuals (e.g. playing sports, social activities) [13]. This research targets the first two categories.

1.1. Wearable sensors and Smartphones

Ambient and *wearable* sensors have been actively exploited for HAR [1]. Video cameras, microphones, GPSs, and sensors for measuring proximity, body motion and vital signs are just a few examples. Current research on ambient sensors has mainly focused on video cameras due to the ease of retrieving visual information from the environment. These have also been combined with other sensors (e.g. with accelerometers and microphones [14]) and recently introduced in wearable technologies for novel ubiquitous applications [15]. However, people's privacy is a downside of vision-based technologies that limits their use in every location. In contrast, recent developments in wearable sensing technologies such as inertial and vital signs sensors are offering less invasive alternatives for HAR [16].

The *accelerometer* is the most commonly used sensor for reading body motion signals [8]. This body sensor is generally used either in multi-sensor arrangements (e.g. triaxial accelerometers and Body Sensor Networks (BSN))

or in combination with others (e.g. gyroscopes, magnetometers, temperature and heart rate sensors) [17]. Bao and Intille [12] proposed one of the earliest HAR systems for the recognition of 20 activities of daily living using five wearable biaxial accelerometers and well-known machine learning classifiers. They achieved reasonably good classification accuracy reaching up to 84% considering the number of activities involved. One evident drawback was related to the number and location of the body sensors used which made the system highly obtrusive. *Gyroscopes* have also been employed for HAR and have demonstrated to improve the recognition performance when used in combination with accelerometers [18, 19].

Smartphones have become an alternative for *wearable sensing* due to the diversity of sensors they support. This aspect, along with the device processing and wireless communication capabilities, makes them a robust tool for performing activity recognition [20, 21]. Smartphones have also advantages over other *ambient sensing* approaches, such as multi-modal sensors in a home environment or surveillance cameras, because they are ubiquitous and require none or little static infrastructure to operate [1]. Inertial sensors such as accelerometers and gyroscopes are present in modern smartphones as they can be mass produced at a low cost. They are an *opportunistic sensing* resource for retrieving body motion data [22, 23].

First smartphone-based approaches worked offline. In [24] the *Centinela* system was presented. It consisted of a chest unit composed of several sensors to measure acceleration data and vital signs (e.g. heart rate, breath amplitude, respiration rate) and a smartphone wirelessly connected via Bluetooth. Data was later processed and classified offline using different machine learning algorithms. Lee and Cho in [25] developed a HAR system of 5 transportation activities which combines labeled and unlabeled data from smartphone inertial sensors with a mixture-of-expert model for classification. Kwapisz et al. [26] developed an offline HAR system using a smartphone provided with a built-in triaxial accelerometer carried on the pocket. Their recognition model allowed the classification of 6 locomotion activities (2 static postures and 4 dynamic activities). Similarly, we proposed in [27] a HAR system using a waist-mounted smartphone. It used a modified SVM with fixed-point arithmetic prediction aiming to obtain a fast implementation more suitable for battery-constrained devices.

More recently, online smartphone-based HAR systems have been proposed. A Nokia smartphone was used in [28] for the online recognition of 6 activities. In [29], Fuentes et al. presented an online motion recognition system using a smartphone with embedded accelerometer which classified 4 BAs through a One-vs-One (OVO) SVM approach. In the same way, the work presented in [30] used an Android smartphone with an embedded accelerometer for the online classification of 4 activities. It also allowed the

adaptation of the learned model for new users by gathering activity samples through a predefined activity protocol.

1.2. Dealing with transitions in HAR systems

In the design of HAR systems there are still some issues that need to be addressed. In most approaches, transitions between activities are usually disregarded since their incidence is generally low and duration is short when compared against other activities. This is pointed out by Lara et al. in [7], nevertheless, the validity of this assumption is application-dependent. Even if the detection of transitions is not required, it is important to notice them in applications where multiple tasks are performed in a short period of time. For instance, activity monitoring during rehabilitation practices, fitness/gymnasium workout activities, equipment assembly and house cleaning. Fluctuations in the prediction during transitions affect the performance of the recognition system if not dealt with properly. A second issue considers that the activities carried out by people are, in real-life situations, more than the ones learned by any HAR system [31]. The remaining activities, unknown to the system, are usually matched as any of the available ones, and this leads to misclassifications. Instead, a better approach would allow the system to tell that it does not predict any of its available classes when its confidence is below certain level. Dealing with these Unknown Activities (UAs) allows more functional HAR systems for a variety of applications.

A number of systems have focused on the detection basic activities and short events. Khan et al. [32] studied 7 basic activities and 7 transitions using three Artificial Neural Networks to separately detect static, dynamic and transitory states. Applications with a large number of classes such as this can give rise to an increase in the false negative rate, especially when the main interest is only on a subset of activities (e.g. basic activities, rather than transitions). In [33], Zhang et al. proposed an offline HAR system that combines basic activities with a joint class of various postural transitions for daily monitoring applications. In [34], Salarian et al. detected *sit-to-stand* and *stand-to-sit* transitions for better distinguishing between *standing* and *sitting*. This was achieved through a fuzzy logic classifier which required for this task, past and future transition information.

Only a few works on HAR have targeted how the presence of transitions between activities impacts system performance. Rednic et al. in [35] performed posture classification of activities for ordnance disposal operations using a multi-accelerometer BSN, while considering the effects of postural transitions in their system using a weighted voting filter in order to improve the classification accuracy of postures by 1%. Moreover, erroneous fluctuations of predicted activities on a classifier can be also dealt in a similar approach. One example of this is also found in [36] where a method called

statistical-hist was proposed. It processed historical variations of the classifier BA predictions using a voting strategy for spurious classification pruning.

In this work, we propose the TAHAR system architecture for the recognition of human activities using smartphones. It targets the classification of Basic Activities (BAs) in real time and pervasively while addressing issues regarding transitions and unknown activities. It offers a flexible and interoperable approach that allows to incorporate new elements (e.g. inertial sensors) into the system and provides an easily exportable output to other ambient intelligent systems that require activity information. Two implementations of the architecture are explored. They differ in the way they deal with transitions that occur in between the activities of interest. In the first case, transitions are treated as unknown activities. Therefore, they are not learned by the machine learning algorithm. Instead, in the second case, transitions are learned by the algorithm as an extra class [33].

We validate the proposed architecture with three case studies: for the most part, we exploit the SBHAR dataset that we have generated from experiments on a group of 30 subjects that performed six locomotion activities while they were carrying a smartphone on their waist. This dataset also contains transition information which is required for the evaluation of the system. Additionally, we exploit other two publicly available datasets for benchmarking: PAMAP2 [37] and REALDISP [38]. The first provides data from nine subjects carrying out 12 physical activities while wearing three inertial measuring units (IMUs) and a heart rate monitor. Conversely, the REALDISP dataset contains recordings from 17 subjects performing a wide range of fitness activities (33) and carrying nine wearable IMUs in different body areas. Although these two dataset are not smartphone-based, they contain equivalent inertial measurements from a larger number of wearable sensors and study a larger number of activities. This allow us to verify the efficacy of the proposed TAHAR architecture in different application domains. This work is, to the best of our knowledge, the first to evaluate how the occurrence of postural transitions affect smartphone-based HAR systems. Results show that the optimal implementation of the architecture is determined by the application, even so dealing with transitions always helps to improve the system accuracy.

The following sections are organized as follows: Section 2 describes in detail the proposed TAHAR system architecture with focus on the prediction layer. Then, Section 3 introduces the three study cases and Section 4 concentrates on the smartphone application implemented to test the architecture. Section 5 depicts the results achieved and, lastly, Section 6 provides concluding remarks and discusses future research directions.

2. TAHAR System Architecture

This section describes the TAHAR system architecture that allows to perform the recognition of physical activities by combining measurements from wearable sensors with supervised machine learning algorithms. The architecture is composed of four functional layers: *Sensing*, *Feature Mapping*, *Prediction* and *Communication*. All these elements are represented in Figure 1 and here described along with their main building blocks:

- The *Sensing* layer collects all the wearable sensors able to provide activity data into the system. For this study, triaxial accelerometers and gyroscopes has been selected as they come embedded in current smartphones and IMUs. These sensors need to provide measurements at frequencies higher than the energy spectrum of human body motion which lies within 0 Hz and 15 Hz [39]. Moreover, other measuring devices such as vital signs or location sensors could be also integrated into this layer in order to increase the amount of activity information and improve the system recognition performance.

- The *Feature mapping* layer concentrates on the conditioning of sensor signals coupled with the extraction of relevant features. Within this layer, the *signal processing* module deals with raw sensor data in order to remove noise and isolate relevant signals (e.g. the extraction of gravity from triaxial acceleration). Following this, the *feature extraction* module applies statistical measures to fixed-width overlapping time windows from the inertial signals in order to form representative feature vectors. The feature mapping process is fully described in [21] for one accelerometer and a gyroscope, however taking into account that more sensors can be incorporated in the architecture, the feature mapping process is updated accordingly (Section 5.2).

- The *Prediction* layer is in charge of reasoning and the main focus of this paper. It consists of two elements: a machine learning module, the *Probabilistic-SVM* (*PrSVM*), that takes feature vectors as input for activity prediction using an SVM; and a filtering module *TFilt* which deals with transitions and fluctuations on the *PrSVM* output. These two modules are detailed in Section 2.2.

- The *Communication* layer receives activity predictions and makes them available either to other mobile applications on the smartphone or externally to other devices. Activity data can also be wirelessly transmitted for storage (e.g. to a data server via Wi-Fi), monitoring, or supply to third-party applications (e.g. to perform higher-level HAR).

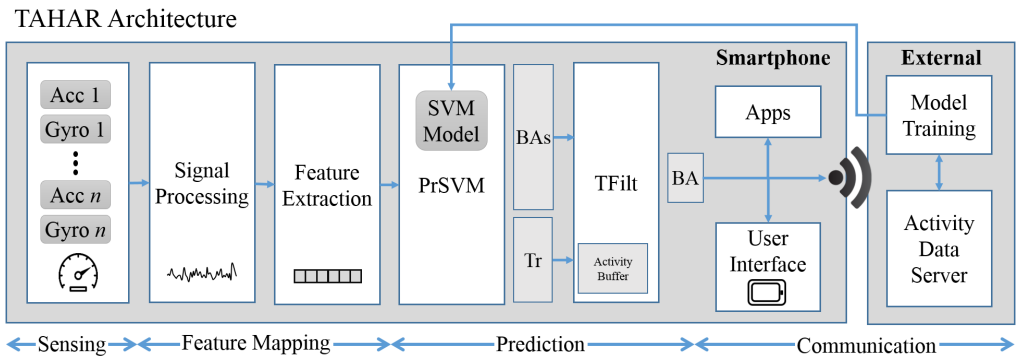

Figure 1: TAHAR System Architecture.

2.1. Implementations

Two different implementations are explored in order to perform the recognition of activities. They differ in the way transitions and unknown activities are handled in the prediction layer. These are both described as follows:

- **Activity Learning (AL)**: this implementation only considers basic activities in the learning algorithm *PrSVM*. Transitions, on the other hand, are not learned. We introduce the idea of detecting unknown activities by assuming that transitions lie between the clusters of activities in the feature space. For this, we combine the probabilistic output of the SVM with temporal activity filtering in order improve the prediction of activities without explicitly learning transitions. Instead, we take into account known relationships between activities to create filters that avoids fluctuations in the classification. For example: the correlation and smooth variations of continuous predictions for each activity, and that the studied activities do no occur simultaneously.

- **Activity and Transition Learning (ATL)**: this implementation includes basic activities and transitions altogether in the learning module. Transitions are therefore considered as an additional class. The filtering treatment given to the output of this machine learning module varies slightly with respect to the previous implementation (AL). Filters are adapted to deal with transitions by taking into account statistical measures about activities such as their average duration, which is shorter on PTs than other BAs, and occurrence between activity pairs.

2.2. TAHAR Prediction Layer Modules

In this section the prediction layer modules of the TAHAR architecture are detailed: *PrSVM* is an SVM [40] that instead of providing a single predicted output, provides a probability vector that represents how likely is an input sample to belong to a particular class. *TFilt* collects consecutive activity predictions from PrSVM in order improve the recognition of basic activities and transitions through heuristic filters that exploit knowledge about the studied activities.

2.2.1. PrSVM*: The Multiclass SVM with Probability Estimates*

The machine learning algorithm employed is a multiclass linear SVM. It consists of a set of One-vs-All (OVA) binary SVMs which characterize each of the one studied activities. This is comprehensively described in [41]. Here we describe the SVM binary model and then extend it to the multiclass approach:

Consider a dataset composed of n patterns of ordered pairs $(\boldsymbol{x}_i, y_i)$ $i \in \{1, ..., n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, and $y_i = \{\pm 1\}$. A binary SVM can be formulated as a Convex Constrained Quadratic Programming (CCQP) minimization problem in the following way:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i \in \{1, ..., n\}, \end{aligned} \tag{1}$$

where the C hyperparameter is the regularization term and $\sum_{i=1}^{n}\xi_i$ represents the upper bound of the number of misclassifications using the hinge loss function.

This formulation is the *primal* SVM problem. However, its solution can be simplified by reformulating it as its *dual* form which uses the Lagrange multipliers:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{C},\ \boldsymbol{y}^T\boldsymbol{\alpha} = 0, \end{aligned} \tag{2}$$

where Q is the symmetric positive semidefinite kernel matrix of size $n \times n$ and $q_{ij} = y_i y_j \boldsymbol{x}_i^T \cdot \boldsymbol{x}_j$.

The prediction of new patterns can be achieved with the SVM Feed Forward Phase (FFP) which is given by:

$$f\left(\boldsymbol{x}\right) = \sum_{i=1}^{n} y_i \alpha_i \boldsymbol{x}_i^T \boldsymbol{x}_j + b. \tag{3}$$

where the bias term b is obtained with the method proposed in [42]. The FFP can be expressed in terms of the weights $\boldsymbol{w}$ and b as: $f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$, where $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$.

It is possible to generalize binary machine learning models to solve problems with more than two classes. In this work we use the OVA approach [43] and take advantage of it because its output directly represents each class.

The output sign of the FFP shows if a new sample is classified either as a given class or not. The magnitude of the output, however, does provide a comparable quantity against the other SVMs. Therefore, an output normalization method is required. We compute probability estimates $p_c(\boldsymbol{x})$ which represent how probable is for a new sample pattern to be classified as a given class.

The selected probability estimation method was proposed by Platt in [44] and it uses the predicted FFP output of the training set and its ground-truth label to fit a sigmoid function of the following form:

$$p(\boldsymbol{x}) = \frac{1}{1 + e^{(\Gamma f(\boldsymbol{x}) + \Delta)}}, \tag{4}$$

where Γ and Δ are function parameters whose optimal values can be found using the $f(\boldsymbol{x}_i)$ values and the targets of the training samples (modified as $t_i = (y_i + 1)/2$) in the following error minimization function:

$$\arg\min_{\Gamma,\Delta} - \sum_{i=1}^{n} t_i \log(p(\boldsymbol{x}_i)) + (1 - t_i) \log(1 - p(\boldsymbol{x}_i)), \tag{5}$$

Generally, for a given number of classes m and a test sample $\boldsymbol{x}$, the probability output of each SVM ($p_c(\boldsymbol{x}) \ \forall \ c \in [1, ..., m]$) is compared against the others to find the class c^* with the Maximum A Posteriori Probability (MAP). For example, assuming that all the classes have the same a priori distribution then:

$$c^* = \arg\max_{c} p_c(\boldsymbol{x}). \tag{6}$$

However, the classification approach presented above only produces a discrete output that indicates the class that best represents a test sample given the assumption that data are independently identically distributed (i.i.d.). Rather than utilizing just one discrete prediction from the SVM, we use the probability estimates more extensively. We provide as the *PrSVM* module output, the vector $\boldsymbol{p}(\boldsymbol{x}) \in \mathbb{R}^m$, that contains the probabilities of belonging to each class of an input sample.

Regarding the AL and ATL implementations, this module only varies in the number of classes learned. ATL includes an additional class which joins all the available transitions into a single one.

2.2.2. TFilt*: Temporal Activity Filtering*

The temporal activity filtering (*TFilt*) module exploits neighboring information from input samples to improve the system recognition performance. It is composed of an *activity buffer* $P \in \mathbb{R}^{s \times m}$ that is created by appending the probability vectors of s consecutive *PrSVM* predictions at different times $\left\{\boldsymbol{p}_{t-1}, \cdots, \boldsymbol{p}_{t-s+1}\right\}$. This buffer can be interpreted as a collection of m activity probability signals in the time domain. This assumption provides an advantage as we can exploit signal filtering techniques to make activity classification more robust. We also assume that only one activity happens at a time and that changes in contiguous activity predictions of a single activity are smooth. The statistical evaluation of the duration of transitions against static and dynamic activities is also taken into account in the design of the filters.

We propose two sets of filters to improve the PrSVM output: *probability filtering* that handles the probability signals and *discrete filtering* that refines the activity output after the discretization of probabilities into activities.

Probability Filtering. Ideally, the output of the *PrSVM* module $\boldsymbol{p}(\boldsymbol{x})$ produces a high probability in one of its elements and tends towards zero in the rest. However, this is not always the case and there are evident variations that occur within the available classes (e.g. multiple classes having high probability simultaneously or fluctuating values between consecutive samples). Moreover, by looking at the studied activities, we can have some idea about the behavior of the inertial signals associated with them. Static activities are more likely to produce a stable probability output due to their steady nature, in contrast, dynamic activities can produce a more irregular output. Transitions' behavior is also dynamic but time constrained and always occurs in between other activities. The use of heuristic filters specifically designed to work for these three activity groups is here explored. They make allowances for conditions regarding duration, frequency, combination with other activities and restrictions of occurrence. Two different heuristic filters are employed:

- The *Fluctuation* filter remove peaks and transients of dynamic activities. During transitions, the probability output of the SVM can exhibit spiky behavior in dynamic activities. These events usually take a short time, therefore the filter measures the length of the signal activation (increase in probability) of these signals for a number of overlapping

windows and decides whether to preserve the signal or not. The filter is also conditioned with the simultaneous activation of static activities because a high probability in these indicates it is unlikely that dynamic activities are also occurring. In the ATL implementation, the transition output of the SVM is not filtered due to its short duration but it is used for conditioning the filtering of dynamic activities when it is active (exceeds a threshold).

- The *Smoothing* filter targets the probability signals of basic activities. It helps to stabilize signal variations when their probability values are greater than a threshold in the activity buffer. Oscillations are smoothed using a linear interpolation. This is aimed to work when static activity signals have high probabilities and it is desired to make evident small differences between them (e.g. activities with high inter-class misclassification such as standing and sitting).

Discrete filtering. The next step after the probability signals have been filtered is to define the most likely activity for each window sample c^*. This is done by finding the MAP over the probability vector ($\boldsymbol{p'} = P'_{(s-1,:)}$) extracted from the filtered activity buffer $\boldsymbol{P}$. From this, one of the classes is selected as the predicted activity. This value is appended to the buffer of activities $\boldsymbol{z}$ that contains the last 3 predicted activities in order to carry out the filtering. It removes sporadic activities that appear for a short time and are unlikely to happen for only a window sample.

Under some circumstances, the entire probability vector contains only low values. This indicates that none of the learned activities represents a particular input. For these cases, a minimum activity threshold is defined and used to label samples as unknown activity. This is particularly useful during transitions in the AL implementation as these are not learned by the *PrSVM* module. In general, this approach can be exploited in real life situations when the HAR system is used while other activities outside the studied set occur. The filter allows to relabel unknown activities as their neighbors when they are detected and its contiguous activities belong to the same class.

3. Study Cases

This section focuses on the datasets employed in the evaluation of the TAHAR architecture. All of them contain inertial data from accelerometers and gyroscopes gathered from groups of subjects performing a set of daily activities. These are: SBHAR, PAMAP2 and REALDISP. Table 1 collects their key features while a description of their main characteristics is presented as follows:

Dataset	Sensors	NP	NA	Activities		
				Static	Dynamic	Transitions
SBHAR [21]	2 acc 1 gyro	30	12	standing sitting lying down	walking walking-upstairs walking-downstairs	stand-to-sit sit-to-stand sit-to-lie lie-to-sit stand-to-lie lie-to-stand
PAMAP2 [37]	3 IMUs: - 2 acc - 1 gyro - 1 mag 1 heart-rate	9	12	standing sitting lying down ironing	walking running cycling nordic-walking walking-upstairs walking-downstairs vacuum-cleaning rope-jumping	
REALDISP [38]	9 IMUs - 1 acc - 1 gyro - 1 mag	17	33	trunk-twist-arms trunk-twist-elbows waist-bends-forward waist-rotation waist-bends reach-heels-backwards lateral-bend lateral-bend-arm-up repetitive-forward-stretching upper-and-lower-body-twist arms-lateral-elevation arms-frontal-elevation frontal-hand-claps arms-frontal-crossing shoulders-high-rotation shoulders-low-rotation arms-inner-rotation	walking jogging running jump-up jump-front-back jump-sideways jump-leg-arms-open-closed jump-rope knees-alternatively-breast heels-alternatively-backside knees-bending-crouching knees-alternate-bend rowing elliptic-bike cycling	

NP: Number of participants, NA: Number of activities

Table 1: Classification of activities by duration and complexity.

3.1. SBHAR: Smartphone-based HAR dataset with Postural Transitions

In [21], we presented a publicly available HAR dataset for the classification of activities using data gathered from the smartphone inertial sensors. The 30 participants of the experiment were instructed to follow a protocol of six basic activities while carrying attached to their belts a smartphone. They generated around 5 hours of experimental data. The dataset collected signals from the device's embedded triaxial accelerometer and gyroscope at a constant rate of 50Hz. This dataset has been updated for this work to include the six postural transitions that occur between the available static activities: *stand-to-sit*, *sit-to-stand*, *sit-to-lie*, *lie-to-sit*, *stand-to-lie*, and *lie-to-stand*. Their labels were defined between the end and the start of consecutive static postures. Data are fully available in [45].

The experimental data also provided relevant information regarding the activity groups. First of all, 8% of the recorded experimental data time corresponds to postural transitions. Regarding activity duration, we found that PTs have a limited duration which is in average 3.73s $\pm$ 1.2 seconds. This is shorter than in BAs (17.3s $\pm$ 5.7). It was also observed that the duration of the available PTs is slightly different among them. Table 2 shows the average duration of the six PTs and their standard deviation. Some PTs, such as *stand-to-lie* which has the longest average duration (4.9s), are actually a sequence of other two transitions (*stand-to-sit* and *sit-to-lie*) as

SBHAR Dataset	
Transition	**Duration (s)**
stand-to-sit	3.41 ± 0.8
sit-to-stand	2.57 ± 0.5
sit-to-lie	4.12 ± 0.8
lie-to-sit	3.69 ± 0.7
stand-to-lie	4.95 ± 1.4
lie-to-stand	3.72 ± 0.8

Table 2: Average duration ± standard deviation of SBHAR dataset postural transitions

it can be observed from the experiment videos. These findings were useful for defining conditions that allow the filtering of transitions in the prediction layer of the TAHAR architecture.

3.2. PAMAP2: Physical Activity Monitoring Dataset

The PAMAP2 dataset was presented by Reiss et al. in [37]. It collected data from four sensor units: three IMUs and one heart-rate monitor. The inertial units were composed of two accelerometers with different scales ($\pm 6g$ and $\pm 16g$), one gyroscope and one magnetometer with a sampling rate of 100Hz. They were located on the chest, dominant wrist and ankle. Sensors were synchronized with a pocket computer for data logging. In their experiment, they included a set of 12 activities which were carried out by nine subjects. Some participants also performed later six optional activities although we only concentrate in the initial group of activities. Over 10 hours of data were gathered from all the dataset activities.

3.3. REALDISP: Realistic Sensor Displacement Benchmark Dataset

The REALDISP dataset was developed by Banos et al. with the idea of evaluating how sensor positioning affects activity recognition systems [38]. For example, when sensors are located either by the experiment instructor in an ideal location, by the participant (self-placement) or by purposely locating them in non-ideal locations. The experiment considered a large set of 33 fitness activities with 17 subjects wearing nine IMUs distributed in different body parts (trunk, upper and lower extremities). Each sensor unit measured acceleration, angular velocity, magnetic field and orientation sampled at 50Hz.

The ideal-location sensor data were considered for this study as they share similarities with SBHAR. Moreover, self-placement sensor setup data was not included because it contained some anomalous sensors and a few subjects did not performed the entire protocol. The whole dataset collects around 39 hours of data, however the ideal-setting recordings sum up to 15 hours.

Regarding activity groups, we categorized the activities as either dynamic or static by looking at the hip movements: whether the hip translates from its original position or not.

4. Software Implementation

In this section we describe the smartphone application that was implemented to test the TAHAR architecture and provide details regarding the error estimation approach used to evaluate the available datasets.

4.1. HARApp

We developed HARApp, a smartphone-based application for real-time activity recognition. This application was built to test the proposed TAHAR architecture using SBHAR data. The application was implemented on a Samsung Galaxy SII device with the Android Operating system (Jelly Bean 4.2.2). The user interface was written in Java and the most resource-consuming tasks such as signal processing, machine learning algorithm and activity filtering were written in C. A screenshot of the app's main window is shown in Figure 3. It contains a graphic representation of the current and last performed activities.

The app structure is depicted in Algorithm 1. Two separate threads process the main functions: $ProcessInertialSignals()$ communicates with inertial sensors and periodically receives their signals for conditioning. In parallel, the $OnlinePrediction()$ function controls the extraction of features and prediction of activities.

$ProcessInertialSignals()$ represents the *sensors* and *signal processing* modules of the TAHAR architecture. In the first stage, it connects with the accelerometer and gyroscope to retrieve the raw triaxial linear acceleration $\boldsymbol{a}_r(t)$ and angular velocity $\boldsymbol{\omega}_r(t)$ time signals. These are read at a constant frequency of 50Hz in order to capture human body motion [39]. Signal conditioning includes noise reduction, whose transfer function is represented by $H_1()$, with a third-order median filter and a third-order low-pass Butterworth filter (cutoff frequency = 20Hz). This produces the signals: triaxial acceleration $\boldsymbol{a}_t(t)$ and angular velocity $\boldsymbol{\omega}(t)$. The acceleration signal is further processed as it combines effect of the gravitational force and the acceleration due to body motion. Assuming that the gravitational component is sensed as a low-frequency harmonic in the signal, the body motion acceleration $\boldsymbol{a}(t)$ can be separated through high-pass filtering ($H_2()$) the acceleration $\boldsymbol{a}_t(t)$ with a cutoff frequency of 0.3Hz. Finally, the gravity $\boldsymbol{g}(t)$ can be found by subtracting $\boldsymbol{a}(t)$ from $\boldsymbol{a}_t(t)$. The signal conditioning process is continuously executed over the inertial signals and its outcome is stored in a circular buffer.

Function	**Description**	**Formulation**
mean $(\boldsymbol{s})$	Arithmetic mean	$\bar{s} = \frac{1}{N}\sum_{i=1}^{N} s_i$
std $(\boldsymbol{s})$	Standard deviation	$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i - \bar{s})^2}$
mad $(\boldsymbol{s})$	Median absolute deviation	$\text{median}_i\,(\,\lvert s_i - \text{median}_j(s_j)\rvert\,)$
max $(\boldsymbol{s})$	Largest values in array	$\max_i(s_i)$
min $(\boldsymbol{s})$	Smallest value in array	$\min_i(s_i)$
skewness $(\boldsymbol{s})$	Frequency signal Skewness	$\text{E}\left[\left(\frac{\boldsymbol{s}-\bar{s}}{\sigma}\right)^3\right]$
kurtosis $(\boldsymbol{s})$	Frequency signal Kurtosis	$\text{E}\left[(\boldsymbol{s}-\bar{s})^4\right] / \text{E}\left[(\boldsymbol{s}-\bar{s})^2\right]^2$
maxFreqInd $(\boldsymbol{s})$	Largest frequency component	$\arg\max_i(s_i)$
energy $(\boldsymbol{s})$	Average sum of the squares	$\frac{1}{N}\sum_{i=1}^{N} s_i^2$
sma $(\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3)$	Signal magnitude area	$\frac{1}{3}\sum_{i=1}^{3}\sum_{j=1}^{N}\lvert s_{i,j}\rvert$
entropy $(\boldsymbol{s})$	Signal Entropy	$\sum_{i=1}^{N}(c_i \log(c_i))\,,\ c_i = s_i / \sum_{j=1}^{N} s_j$
iqr $(\boldsymbol{s})$	Interquartile range	$\text{Q3}(\boldsymbol{s}) - \text{Q1}(\boldsymbol{s})$
autoregression $(\boldsymbol{s})$	4th order Burg Autoregression coefficients	$\boldsymbol{a} = \text{arburg}(\boldsymbol{s}, 4)\,, \boldsymbol{a} \in \mathbb{R}^4$
correlation $(\boldsymbol{s_1}, \boldsymbol{s_2})$	Pearson Correlation coefficient	$C_{1,2}/\sqrt{C_{1,1}C_{2,2}}, C = \text{cov}(\boldsymbol{s}_1, \boldsymbol{s}_2)$
meanFreq $(\boldsymbol{s})$	Frequency signal weighted average	$\sum_{i=1}^{N}(is_i) / \sum_{j=1}^{N} s_j$
energyBand $(\boldsymbol{s}$,$a, b)$	Spectral energy of a frequency band $[a, b]$	$\frac{1}{a-b+1}\sum_{i=a}^{b} s_i^2$
angle $(\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, v)$	Angle between signal mean and vector	$\tan^{-1}(\lVert[\bar{s}_1, \bar{s}_2, \bar{s}_3] \times \boldsymbol{v}\rVert\,, [\bar{s}_1, \bar{s}_2, \bar{s}_3] \cdot \boldsymbol{v})$

N: signal vector length, Q: Quartile.

Table 3: List of measures for computing feature vectors.

OnlinePrediction() is triggered by scheduled interruptions every 1.28s which correspond to the duration of half window sample. Its periodicity satisfies the sliding-windows criteria: a time span of 2.56s and 50% window overlap. Similar sampling approaches have confirmed to be successful in other HAR works such as in [46, 47, 41]. Window samples A, G, and Ω are collected from the buffered inertial data and become the input of the *feature extraction* module which provides a feature vector $\boldsymbol{x} = \phi(A, G, \Omega)$ composed of measures in the time and frequency domain. They provide a collection of 561 informative features which has been selected based on previous works [21, 12, 48, 49]. They include: Signal Magnitude Area, arithmetic mean, Standard Deviation, autoregression coefficients, interquartile range, signal entropy, signal-pair correlation, amongst others. They are listed in Table 3.

The estimated feature vectors become the input of the prediction module *PrSVM*. The SVM FFP is applied using model parameters ($\boldsymbol{w}_c$ and b_c) and output normalization constants (Γ_c and Δ_c) to obtain the activity probability vector $\boldsymbol{p}(\boldsymbol{x})$. These parameters are learned offline for both TAHAR implementations (AL and ATL). The vector is appended to the *activity buffer* P in the *TFilt* module.

The implemented probability filters are applied over the activity buffer

($P' = \Phi(P)$) based on the activity groups (static/dynamic activity or transitions) as described in Section 2.2.2. A class is considered active when $p_c > \tau = 0.2$. The length of P, s, was selected based on the available information from the dataset regarding to the activities and transitions duration as described in Section 3.1. In this application $s = 5$ which is equivalent to a prediction latency of 5.12s. From P', the filtered probability vector $\boldsymbol{p}'$ is extracted. This is discretized with MAP in order to obtain an activity prediction c^*. This value is appended to the buffer of activities $\boldsymbol{z}$ that contains the last 3 predicted activities for discrete filtering and the final activity estimation ($\hat{c} = \Psi(\boldsymbol{z})$).

The *HAR output* of the system can be visualized on screen, accessed by other applications on the smartphone through broadcasting and stored in a log file for subsequent analysis. Moreover, a communications interface allows access to live prediction data from external devices through wireless connections (e.g. Wi-Fi and 3G).

4.2. System Error Estimation

In order to provide a clearer idea of how the recognition system works, Figure 2 shows an example of an activity sequence. It includes acceleration signals along with the probability estimates obtained from the *PrSVM* module. It is noticeable that the probability of each class increases or decreases depending on the activity performed at different times. Even though there is some noise in the forecasted probabilities, it is possible to visualize how the algorithm behaves in the presence of basic activities and transitions. The figure also shows two common misclassification examples. The first error type occurs during basic activities (BA error) and is due to similarities between two static postures (e.g. standing and sitting) which usually present high interclass error. The second type (PT Error) occurs during postural transitions. This misclassification is generally characterized by incorrectly predicting postural transitions as dynamic activities (walking upstairs in the example). The expected correct predictions of the AL and ATL implementations are also depicted (below the MAP prediction). They both remove the two types of errors and provide a solution to the classification problem.

The method for the evaluation of the online system error requires some modifications given that postural transitions and unknown activities are taken into account. Table 4 explains graphically our error assessment method given the different conditions that can appear. From these conditions we have developed an error metric to evaluate the system performance which has the following formulation:

Algorithm 1: TAHAR Architecture Algorithm

Require:

$\boldsymbol{a}$: Triaxial linear acceleration
$\boldsymbol{\omega}$: Triaxial angular velocity
$\boldsymbol{g}$: Gravity
$H_1(\cdot)$: Noise reduction transfer function
$H_2(\cdot)$: Body acceleration transfer function
$\phi(\cdot)$: Feature extraction function
T: Windows size
m: Number of classes
$\mathbf{p}$: activity probability vector $\mathbf{p} = [p_1, \ldots, m]^T$
P: Buffer of probability vectors $P \in \mathbb{R}^{m \times m}$
P': Filtered buffer of probability vectors
$\boldsymbol{z}$: Buffer of discrete activity predictions $\boldsymbol{z} \in \mathbb{R}^s$
$\Phi(\cdot)$: Probability filtering function
$\Psi(\cdot)$: Discrete filtering function

function `ProcessInertialSignals`$(\boldsymbol{a}_r(t), \boldsymbol{\omega}_r(t))$

$\boldsymbol{a}_\tau(t) = H_1(\boldsymbol{a}_r(t)),$ // Noise Filtering
$\boldsymbol{\omega}(t) = H_2(H_1(\boldsymbol{\omega}_r(t)))$
$\boldsymbol{a}(t) = H_2(\boldsymbol{a}_\tau(t))$ // Body acceleration Extraction
$\boldsymbol{g}(t) = \boldsymbol{a}_\tau(t) - \boldsymbol{a}(t)$ // Gravity extraction
return $\boldsymbol{a}(t), \boldsymbol{g}(t), \boldsymbol{\omega}(t)$

end

function `OnlinePrediction`$(t, \boldsymbol{a}(t), \boldsymbol{g}(t), \boldsymbol{\omega}(t), B, \boldsymbol{z})$

$A = \{\boldsymbol{a}(t') : t' \in [t-T, \ldots, t]\},$ // Window sampling
$G = \{\boldsymbol{g}(t') : t' \in [t-T, \ldots, t]\},$
$\Omega = \{\boldsymbol{\omega}(t') : t' \in [t-T, \ldots, t]\}$
$\boldsymbol{x} = \phi(A, G, \Omega)$ // Feature Extraction and Normalization
for $c \in \{1, \ldots, m\}$ **do** // Multiclass SVM
$f_c(\boldsymbol{x}) = \boldsymbol{w_c}^T \boldsymbol{x} + b_c$ // FFP
$p_i = 1/\left(1 + e^{(\Gamma_c f_c(\boldsymbol{x}) + \Delta_c)}\right)$ // Prob. Estimation
end
$P = \begin{bmatrix} \boldsymbol{p}^T \\ P_{(1:end-1,:)} \end{bmatrix}$ // Append probability vector
$P' = \Phi(P)$ // Activity probability filtering
$c^* = \arg\max_{c \in [1,\ldots,m]} \quad P'_{(s-1,c)}$ // MAP
$\boldsymbol{z} = \begin{bmatrix} c^* \\ \boldsymbol{z}_{(1:end-1)} \end{bmatrix}$ // Append last activity prediction
$\hat{c} = \Psi(\boldsymbol{z})$ // Discrete filtering and activity estimation
return $\hat{c}$

end

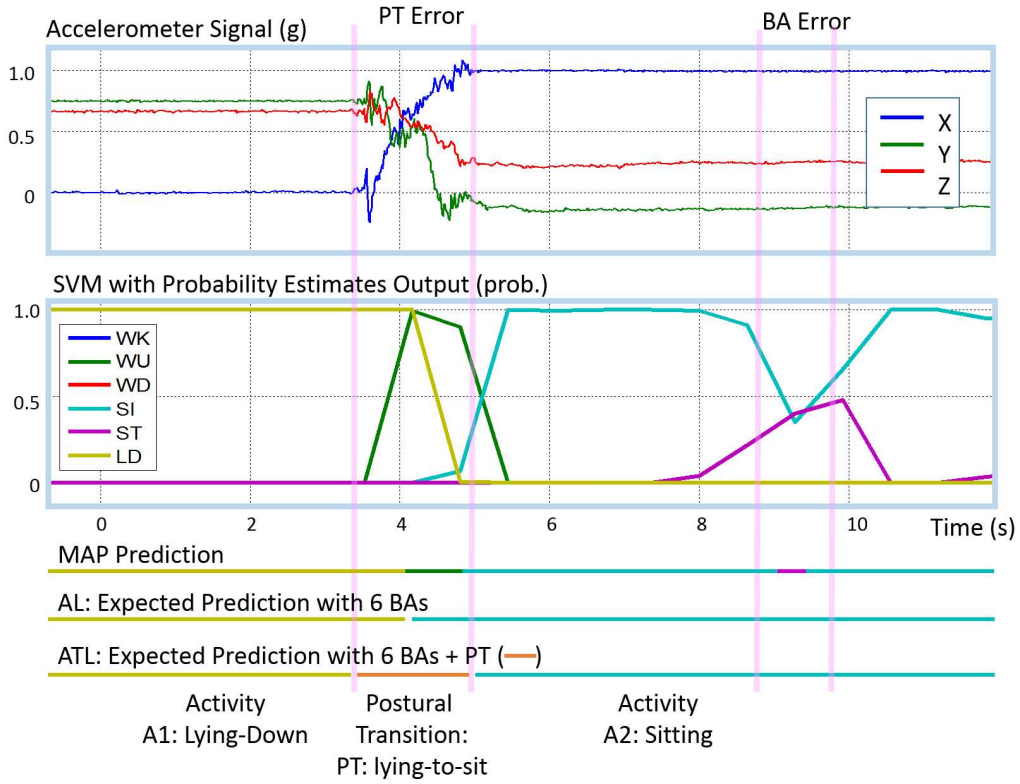

Figure 2: Misclassification examples during postural transitions and static postures. (Top) Acceleration signals show the transition between the two postures. (Middle) The output of the SVM shows how likely each activity is for each window sample. (Bottom) The prediction of the activities using the MAP approach is compared against the expected output with the AL and ATL implementations.

Figure 3: HARApp smartphone user interface

Ground-Truth	Prediction	Error Evaluation
Basic Activities		
A1 - A1 - A1	A1 - A1 - A1	Correct
A1 - A1 - A1	A1 - A2 - A1	Incorrect
A1 - A1 - A1	A1 - UA - A1	Incorrect
A1 - A1 - A1	A1 - PT - A1	Incorrect[1]
Transitions		
A1 - PT - A2	A1 - A1∨A2 - A2	Correct
A1 - PT - A2	A1 - A3 - A2	Incorrect
A1 - PT - A2	A1 - UA - A2	Correct
A1 - PT - A2	A1 - PT - A2	Correct[1]

A = Activity, U = Unknown. [1] Only applicable to the ATL implementation.

Table 4: Classification error assessment conditions for BAs and PTs.

$$e\left(\alpha_t\right) = \begin{cases} 0 & \text{if } \begin{cases} g_t = \alpha_t \vee \\ \left(g_t = PT \wedge g_{t-1} \neq g_{t+1} \wedge \left(\alpha_t = g_{t-1} \vee \alpha_t = g_{t+1}\right)\right) \vee \\ \left(g_t = PT \wedge \alpha_t = UA\right) \end{cases} \\ 1 & \text{otherwise,} \end{cases} \tag{7}$$

where g_t is the ground-truth label for any test sample at time t and α_t is the predicted activity.

Notice that the error function penalizes the detection of either unknown activity or postural transition during the occurrence of basic activities as we expect them to occur only during transitions. It shows that any aim to reduce the error during transitions can reduce the overall performance as it can have unfavorable effects in the predictions of the other activities. This error metric is used for the proposed AL and ATL implementations.

5. Results

In this section we present the experimental results obtained in the study cases of the TAHAR architecture: SBHAR, PAMAP2 and REALDISP. This evaluation provides a general overview of the architecture performance with different setups taking into account the AL and ATL implementations.

The performance was measured in terms of system error at different locations of the architecture pipeline. In particular, the outputs of the *PrSVM*, *TFilt* probabilistic and *TFilt* discrete modules. This approach allowed to see how the modules were progressively affecting the classification performance.

For error estimation, we used a leave-one-subject-out approach on which every subject was selected as a test case and the remaining subjects were used for model training. The error was then obtained by averaging over subject errors.

5.1. SBHAR Evaluation

The HAR system presented in [21] was used as a reference point for the evaluation. Its classification approach was also SVM-based but it did not include filtering. The basic activities studied were the same as SBHAR but postural transitions were not included. The output of this approach is equivalent to the output of the *PrSVM* module output on the AL implementation. The error achieved by the reference system was 3.59%.

Table 5 shows the error measurements of the TAHAR architecture on the SBHAR dataset for the two implementations (AL and ATL). It also separates the error according to the activity groups, whether it is a basic activity or a postural transition. The overall error is a weighted estimation that considers the contributions of the dataset activities (e.g. PTs are the 8% of the data).

We obtained a system error of 7.41% on the PrSVM output for the AL implementation. This shows an increase of the error by 3.82% percentage points against the reference HAR system. This occurred mainly due to the misclassifications that occurred during postural transitions (39.93%). This shows one of the disadvantages of not dealing with transitory events in a real-time classification system, for instance by filtering or learning them. The error on BAs instead remains much lower (4.52%). We can also observe that *TFilt* improves the classification of PTs by greatly reducing the error down to 7.82%. The final error of the AL implementation is 3.64% which outperforms by a small amount the one achieved in [21] that did not even considered PTs.

On the other hand, the ATL implementation presents a different behavior. Since postural transitions are learned in the *PrSVM* module, its recognition error is much lower than in the AL implementation. Even before filtering the error is quite low (0.98%). This is a noticeable advantage against the previous implementation. The TFilt is still contributing to reduce the final error on PTs and BAs but in smaller proportion than in AL. The final error achieved in ATL is 3.22% which outperforms AL by 0.42%. Table 6 shows the performance of the two proposed implementations by means of confusion matrices on the leave-one-subject-out test data.

In terms of computational speed of the smartphone application HARApp, the duration of a complete prediction cycle takes in average about 152ms for the AL method and 162ms for the ATL using a SGSII smartphone. These times are similar as they share same the feature extraction process which is nearly 92% of the processing time. The remaining time is dedicated to the

SBHAR			
Output	**Error**		
	BAs	PTs	Overall
AL Implementation			
PrSVM	$4.52\% \pm 4.8$	$39.93\% \pm 8.3$	$7.41\% \pm 4.8$
TFilt Probability	$3.41\% \pm 4.5$	$17.26\% \pm 6.5$	$4.54\% \pm 4.4$
TFilt Discrete	$3.26\% \pm 4.4$	$7.82\% \pm 8.6$	$3.64\% \pm 4.4$
ATL Implementation			
PrSVM	$4.49\% \pm 4.5$	$0.98\% \pm 2.0$	$4.20\% \pm 4.2$
TFilt Probability	$3.56\% \pm 4.6$	$0.40\% \pm 0.9$	$3.30\% \pm 4.2$
TFilt Discrete	$3.50\% \pm 4.7$	$0.24\% \pm 0.7$	$3.22\% \pm 4.3$

Table 5: SHBAR System error based on filtering stage and type of activity

	A1	A2	A3	A4	A5	A6	A7	A8
A1	1833	65	0	2	3	0	0	6
A2	11	1762	32	7	3	0	0	1
A3	0	0	1677	0	3	0	0	2
A4	0	3	0	1897	73	4	0	1
A5	0	6	0	124	2030	0	0	1
A6	0	0	0	1	0	2150	0	0
A7	0	37	0	46	2	0	954	0

	A1	A2	A3	A4	A5	A6	A7	A8
A1	1834	64	5	3	2	0	1	0
A2	10	1743	51	5	5	0	16	0
A3	0	2	1671	1	7	0	1	0
A4	0	0	0	1875	94	6	3	0
A5	0	2	0	109	2049	0	1	0
A6	0	0	0	1	0	2148	2	0
A7	0	1	2	0	0	0	1036	0

A1:Walking, A2:Walking-Upstairs, A3:Walking-Downstairs, A4:Sitting,
A5:Standing, A6:Lying-Down, A7:Postural Transition, A8:Unknown Activity

Table 6: SBHAR Dataset Confusion Matrices. Top: AL Implementation, Bottom: ATL Implementation.

prediction layer (*PrSVM* and *TFilt*), which only varies in proportion to the number of predicted classes per implementation. The app consumes around 6MB of memory and 4.2% of the CPU available time.

5.2. PAMAP2 Evaluation

In [37], PAMAP2 was introduced and also used for benchmarking four classification problems. The *all activity recognition task* was the most complex because it included the study of the 12 activities. In particular, the subject independent study (leave-one-subject-out 9-fold cross validation). They explored five machine learning algorithms and found a maximum recognition error of 10.76% with a k-nearest neighbors (kNN) classifier. That study is the reference point for the evaluation of the PAMAP2 dataset in the TAHAR architecture.

For the purposes of this research we only took into account from each IMU one of the accelerometers ($\pm 6g$) and the gyroscope. Other sensors were disregarded as this simplified the comparison between datasets. Moreover, as the number of sensors is larger than in SBHAR, the feature mapping process was updated by increasing the feature vector length by the number of inertial units u. For each accelerometer-gyroscope pair, the same features were extracted. In contrast with SBHAR, wearable sensors were used for data gathering instead of a single smartphone. However, this data was straightforwardly adapted to the proposed architecture. The evaluation of this dataset was performed on a PC (3.4GHz CPU Intel i5 CPU with 8GB of RAM) using code implemented in Matlab.

The sampling rate of the PAMAP2 dataset was 100Hz. We subsampled the sensor signals in order to match the frequency of the other two datasets (50Hz), still sufficient for sensing body motion. Moreover, the length of the window samples was 5.12 seconds with an overlap between windows of one second. We maintained these values for an objective comparison against the reference point.

The PAMAP2 dataset did not include transition data so only the AL implementation of the architecture was tested. Table 7 shows the obtained classification results. It includes an initial error of 6.96% achieved at the PrSVM output and then this is improved by the filtering modules until reaching 5.67%. This value is nearly 50% lower than the one obtained in [37]. Moreover, Table 8 contains the confusion matrix of the 12 studied activities which allows to easily identify misclassifications, in particular between similar activities (e.g. ironing and standing, or walking and vacuum cleaning).

PAMAP2 - AL Implementation	
Output	**Error**
PrSVM	6.96% ±2.5
TFilt Probability	6.24% ±2.7
TFilt Discrete	5.67% ±2.7
Reference System [37]	10.76%

Table 7: PAMAP2 System error based on filtering stage and type of activity

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
A1	1749	5	11	0	0	0	0	4	0	24	18	0
A2	16	1612	52	2	0	18	0	4	5	17	15	1
A3	14	114	1505	2	8	0	1	15	16	13	97	0
A4	0	0	4	2214	1	1	7	8	12	0	7	1
A5	1	3	10	2	748	5	0	3	17	120	6	0
A6	1	3	1	0	0	1504	0	11	0	15	12	0
A7	0	6	1	16	2	0	1737	1	1	1	9	1
A8	0	4	8	2	0	1	0	1014	7	1	10	0
A9	2	17	11	2	7	2	0	15	850	7	12	2
A10	0	4	11	4	0	0	0	1	2	1586	42	0
A11	0	0	56	2	7	1	0	1	5	23	2156	6
A12	1	22	0	3	0	0	2	2	29	1	8	379

A1:lying, A2:sitting, A3:standing, A4:walking, A5:running, A6:cycling, A7:nordic-walking, A8:ascending-stairs, A9:descending-stairs, A10:vacuum-cleaning, A11:ironing, A12:rope-jumping

Table 8: PAMAP2 Dataset Confusion Matrix. AL Implementation

5.3. REALDISP Evaluation

The REALDISP data used here for evaluation only comprise the experiments performed in the ideal location setting. Other settings that involve sensor displacement are out of the scope of this work. In [38], Bano et al. performed a classification exercise of the PAMAP2 dataset using three machine learning algorithms. kNN also showed the best classification performance (96%) against decision tree and nearest class center classifiers. For feature mapping, they only used basic statistical measures (e.g. mean and standard deviation) of their four available inertial signals from the nine IMUs. This suggests that better results could have been achieved with the introduction of novel features.

Similarly to the previous dataset PAMAP2, we employed the same approach regarding the use of multiple IMU sensors and feature mapping. This dataset allowed to evaluate the proposed TAHAR architecture in a more complex setup due to its large number of activities and sensors. The window sampling performed in [38] did not considered window overlapping, however, this is required in the architecture as we assume connected consecutive windows in the filtering module. Instead, we use a 50% window overlap for the evaluation.

TAHAR classifies the 33 fitness activities contained in the dataset and, since no postural transitions are included on this dataset, only the AL imple-

mentation is tested. This includes 17 static activities and 15 dynamic ones as seen in Table 1.

Results of the classification are depicted in Table 9. They are an improvement of the REALDISP dataset classification with respect the reference experiment in [38] (from 96% to 99.52%). Moreover, even though the error is close to zero, it is possible to see an small improvement of the PrSVM output with the use of the temporal filter module (*TFilt*). Table 10 contains the confusion matrix of the test data. Most of the instances lie on the diagonal except for a few cases (e.g. misclassifications between activities 14 and 28: reach-heels-backwards and knees-bending-crouching).

6. Conclusions

In this work, we presented the TAHAR architecture for the recognition of physical activities. It combines inertial sensors for body motion capture, a machine learning algorithm for activity prediction and a filter of consecutive predictions for output refinement. We demonstrated its successful use on three human activity datasets with diverse groups of activities, number of sensors and number of participants; and showed that its recognition performance outperforms previous related works.

Results also showed the improvements that can be made to the system when fluctuations in the prediction of activities and transitions are taken into consideration. Specifically, this was done in the activity filtering module *TFilt* that improves (up to 3.77 percentage points) the output of the machine learning algorithm (*PrSVM*) by considering the correlation between contiguous events, the non-simultaneous occurrence of activities and the studied activity groups. Moreover, the incorporation of the unknown activity class allowed the system to better deal with activities not learned by the algorithm. For example, by handling PTs as unknown events in the AL implementation of the SBHAR dataset. This concept is also valid in real-life situations (such as activity monitoring) where there are high chances to perform activities that are not known in advance. It is preferable a system that notifies that an activity is unknown rather than classifying it as one of the learned activities.

Both implementations of the architecture are suitable options for HAR. However, here we provide some considerations in order to guide their selection based on their target application:

- The ATL implementation is required when the detection of transitions is required. The AL implementation instead avoids learning these events but still prevents problems that could arise in the presence of transitions during classification.

REALDISP - AL Implementation	
Output	**Error**
PrSVM	0.63% ±0.9
TFilt Probability	0.53% ±0.9
TFilt Discrete	0.48% ±0.9
Reference System [38]	4.00%

Table 9: REALDISP System error based on filtering stage and type of activity

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31	A32	A33
A1	837	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A2	3	675	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A3	0	1	635	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A4	0	2	0	100	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A5	0	3	0	0	198	1	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A6	0	0	0	3	0	211	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A7	0	0	0	0	0	0	229	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A8	0	0	0	4	0	0	0	107	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A9	0	0	0	0	0	0	0	0	439	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A10	0	0	0	0	0	0	0	0	0	363	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A11	0	0	0	0	0	0	0	0	0	0	369	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A12	0	0	0	0	0	0	0	0	0	0	0	303	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A13	0	0	0	0	0	0	0	0	0	0	0	0	340	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A14	0	0	0	0	0	0	0	0	0	0	0	0	0	264	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0
A15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	221	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	216	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	177	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	232	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	247	0	0	0	0	0	0	0	0	0	0	0	0	0
A21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	226	0	0	0	0	0	0	0	0	0	0	0	0
A22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	261	0	0	0	0	0	0	0	0	0	0	0
A23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	210	4	0	0	0	0	0	0	0	0	0
A24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	132	0	0	0	0	0	0	0	0	0
A25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	168	0	0	0	0	0	0	0	0
A26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	66	3	0	0	0	0	0	0
A27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78	0	0	0	0	0	0
A28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	318	0	0	0	0	0
A29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	390	0	0	0	0
A30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	194	0	0	0
A31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	414	0	0
A32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	613	0
A33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627

A1:walking, A2:jogging, A3:running, A4:jump-up, A5:jump-front-back, A6:jump-sideways, A7:jump-leg-arms-open-closed, A8:jump-rope, A9:trunk-twist-arms, A10:trunk-twist-elbows, A11:waist-bends-forward, A12:waist-rotation, A13:waist-bends, A14:reach-heels-backwards, A15:lateral-bend, A16:lateral-bend-arm-up, A17:repetitive-forward-stretching, A18:upper-trunk-and-lower-body-opposite-twist, A19:arms-lateral-elevation, A20:arms-frontal-elevation, A21:frontal-hand-claps, A22:arms-frontal-crossing, A23:shoulders-high-amplitude-rotation, A24:shoulders-low-amplitude-rotation, A25:arms-inner-rotation, A26:knees-alternatively-breast, A27:heels-alternatively-backside, A28:knees-bending-crouching, A29:knees-alternatively-bend-forward, A30:rotation-on-the-knees, A31:rowing, A32:elliptic-bike, A33:cycling

Table 10: REALDISP Dataset Confusion Matrix. AL Implementation

- AL is easier to implement because learning does not include transitions. In applications with a large number of activities, the recording and labeling of transitions between basic activities becomes more complex as the number of possible transitions is given by $\upsilon(\upsilon - 1)$, where υ is the number of studied basic activities.

- The selection of the implementation can be also guided by recurrence of transitions with respect other activities. If they do not occur too often or if the time between transitions is rather large, learning transitions is then not fully required and the AL implementation is sufficient.

- When information regarding transitions is not available (e.g. if the their labels are not included in the dataset), the AL implementation is the only approach that can be employed for developing a HAR system.

From this research, some ideas arise as future work. They include the exploration of novel approaches to make more robust HAR after activity prediction on the PrSVM module. For example, by applying probabilistic models such as Markov chains [50] composed of connected nodes, each one representing an activity, and using activity probability estimates as observations. Furthermore, the study of the repeated detection of unknown activities as an indication that the system is not working correctly. For instance, if the sensor is not well located or if the activities performed by a new user seem not to be properly recognized.

Acknowledgments

This work was supported in part by the Erasmus Mundus Joint Doctorate in Interactive and Cognitive Environments, which is funded by the EACEA Agency of the European Commission under EMJD ICE FPA n 2010-0012.

References

[1] L. Chen, J. Hoey, C. Nugent, D. Cook, Z. Yu, Sensor-based activity recognition, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42 (2012) 790–808.

[2] D. J. Cook, S. K. Das, Pervasive computing at scale: Transforming the state of the art, Pervasive Mobile Computing 8 (2012) 22–35.

[3] A. Campbell, T. Choudhury, From smart to cognitive phones, IEEE Pervasive Computing.

[4] B. P. Clarkson, Life patterns: structure from wearable sensors, Ph.D. thesis, Massachusetts Institute of Technology (2002).

[5] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, P. Havinga, Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey, in: International Conference on Architecture of Computing Systems, 2010.

[6] W. Lin, M.-T. Sun, R. Poovandran, Z. Zhang, Human activity recognition for video surveillance, in: IEEE International Symposium on Circuits and Systems, 2008.

[7] O. Lara, M. Labrador, A survey on human activity recognition using wearable sensors, IEEE Communications Surveys Tutorials 1 (2012) 1–18.

[8] A. Mannini, A. M. Sabatini, Machine learning methods for classifying human physical activity from on-body accelerometers, Sensors 10 (2010) 1154–1175.

[9] R. Poppe, A survey on vision-based human action recognition, Image and vision computing 28 (2010) 976–990.

[10] B. Nham, K. Siangliulue, S. Yeung, Predicting mode of transport from iphone accelerometer data, Tech. rep., Tech. report, Stanford Univ (2008).

[11] E. Tapia, S. Intille, K. Larson, Activity recognition in the home using simple and ubiquitous sensors, in: Pervasive Computing, 2004.

[12] L. Bao, S. Intille, Activity recognition from user-annotated acceleration data, in: Pervasive Computing, 2004.

[13] J. Aggarwal, M. S. Ryoo, Human activity analysis: A review, ACM Computing Surveys 43 (2011) 16.

[14] S. Tasoulis, C. Doukas, V. Plagianakos, I. Maglogiannis, Statistical data mining of streaming motion data for activity and fall recognition in assistive environments, Neurocomputing.

[15] A. Behera, D. Hogg, A. Cohn, Egocentric activity monitoring and recovery, in: Asian Conference on Computer Vision, 2013.

[16] D. Townsend, F. Knoefel, R. Goubran, Privacy versus autonomy: A tradeoff model for smart home monitoring technologies, in: Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE, 2011.

[17] G.-Z. Yang, M. Yacoub, Body sensor networks, Springer, 2006.

[18] W. Wu, S. Dasgupta, E. E. Ramirez, C. Peterson, G. J. Norman, Classification accuracies of physical activities using smartphone motion sensors, Journal of Medical Internet Research 14 (2012) 105–130.

[19] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.-L. Reyes-Ortiz, Training computationally efficient smartphone-based human activity recognition models, in: International Conference on Artificial Neural Networks, 2013.

[20] A. Ghio, L. Oneto, Byte the bullet: Learning on real-world computing architectures, in: European Symposium On Artificial Neural Networks, Computational Intelligence and Machine Learning ESANN, 2014.

[21] D. Anguita, A. Ghio, L. Oneto, J.-L. Parra, Xavier.and Reyes-Ortiz, A public domain dataset for human activity recognition using smartphones, in: European Symposium On Artificial Neural Networks, Computational Intelligence and Machine Learning ESANN, 2013.

[22] A. M. Khan, Y.-K. Lee, S. Lee, T.-S. Kim, Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis, in: IEEE International Conference on Future Information Technology, 2010.

[23] D. Roggen, K. Förster, A. Calatroni, T. Holleczek, Y. Fang, G. Tröster, P. Lukowicz, G. Pirkl, D. Bannach, K. Kunze, A. Ferscha, C. Holzmann, A. Riener, R. Chavarriaga, J. del R. Millán, Opportunity: Towards opportunistic activity and context recognition systems, in: IEEE Workshop on Autononomic and Opportunistic Communications, 2009.

[24] O. D. Lara, A. J. Pérez, M. A. Labrador, J. D. Posada, Centinela: A human activity recognition system based on acceleration and vital sign data, Pervasive and Mobile Computing 8 (2012) 717 – 729.

[25] Y.-S. Lee, S.-B. Cho, Activity recognition with android phone using mixture-of-experts co-trained with labeled and unlabeled data, Neurocomputing.

[26] J. R. Kwapisz, G. M. Weiss, S. A. Moore, Activity recognition using cell phone accelerometers, SIGKDD Explorations Newsletter 12 (2011) 74–82.

[27] D. Anguita, A. Ghio, L. Oneto, J. Parra, Xavierand Reyes-Ortiz, Human activity recognition on smartphones using a multiclass hardware-friendlysupport vector machine, in: Ambient Assisted Living and Home Care, 2012.

[28] T. Brezmes, J. Gorricho, J. Cotrina, Activity recognition from accelerometer data on a mobile phone, Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living 5518 (2009) 796–799.

[29] D. Fuentes, L. Gonzalez-Abril, C. Angulo, J. Ortega, Online motion recognition using an accelerometer in a mobile device, Expert Systems with Applications 39 (2012) 2461 – 2465.

[30] M. Kose, O. D. Incel, C. Ersoy, Online human activity recognition on smart phones, in: Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data, 2012.

[31] A. Bulling, U. Blanke, B. Schiele, A tutorial on human activity recognition using body-worn inertial sensors, ACM Computing Surveys 46 (2014) 33.

[32] A. M. Khan, Y.-K. Lee, S. Y. Lee, T.-S. Kim, A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer., IEEE transactions on information technology in biomedicine 14 (2010) 1166–1172.

[33] S. Zhang, P. McCullagh, C. Nugent, H. Zheng, Activity monitoring using a smart phone's accelerometer with hierarchical classification, in: International Conference on Intelligent Environments, 2010.

[34] A. Salarian, H. Russmann, F. J. Vingerhoets, P. R. Burkhard, K. Aminian, Ambulatory monitoring of physical activities in patients with parkinson's disease, IEEE Transactions on Biomedical Engineering 54 (2007) 2296–2299.

[35] R. Rednic, E. Gaura, J. Kemp, J. Brusey, Fielded autonomous posture classification systems: design and realistic evaluation, in: ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2013.

[36] D. Riboni, C. Bettini, Cosar: hybrid reasoning for context-aware activity recognition, Personal and Ubiquitous Computing 15 (2011) 271–289.

[37] A. Reiss, D. Stricker, Introducing a new benchmarked dataset for activity monitoring, in: International Symposium on Wearable Computers, 2012.

[38] O. Baños, M. Damas, H. Pomares, I. Rojas, M. A. Tóth, O. Amft, A benchmark dataset to evaluate sensor displacement in activity recognition, in: ACM Conference on Ubiquitous Computing, 2012.

[39] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, B. G. Celler, Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring, IEEE Transactions on Information Technology in Biomedicine 10 (2006) 156–167.

[40] E. L. Allwein, R. E. Schapire, Y. Singer, Reducing multiclass to binary: A unifying approach for margin classifiers, J. Mach. Learn. Res.

[41] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz, Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic, Journal of Universal Computer Science 19 (2013) 1295–1314.

[42] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, K. R. K. Murthy, Improvements to platt's smo algorithm for svm classifier design, Neural Computation 13 (2001) 637–649.

[43] R. Rifkin, A. Klautau, In defense of one-vs-all classification, Journal of Machine Learning Research 5 (2004) 101–141.

[44] J. C. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in: Advances in Large Margin Classifiers, 1999.

[45] J.-L. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, X. Parra, Recognition of basic activities and postural transitions using smartphones data set, `http://www.har.smartlab.ws` (2014).

[46] R. W. DeVaul, S. Dunn, Real-time motion classification for wearable computing applications, Tech. rep., MIT Media Lab (2001).

[47] K. Van Laerhoven, O. Cakmakci, What shall we teach our pants?, in: International Symposium on Wearable Computers, 2000.

[48] N. Lovell, N. Wang, E. Ambikairajah, B. G. Celler, Accelerometry based classification of walking patterns using time-frequency analysis, in: IEEE Annual International Conference of the Engineering in Medicine and Biology Society, 2007.

[49] A. Sama, D. E. Pardo-Ayala, J. Cabestany, A. Rodríguez-Molinero, Time series analysis of inertial-body signals for the extraction of dynamic properties from human gait, in: IEEE International Joint Conference on Neural Networks, 2010.

[50] E. Kim, S. Helal, D. Cook, Human activity recognition and pattern discovery, Pervasive Computing, IEEE.

Biography of the authors

Jorge-L. Reyes-Ortiz was born in Barranquilla, Colombia in 1980. He received his B.Sc. in Electronic Engineering in 2003 at the Universidad Pontificia Bolivariana and his M.Sc. in Artificial Intelligence in 2008 at the University of Edinburgh. He is a PhD Student on Smartphone-Based Human Activity Recognition at the University of Genoa and Universitat Politécnica de Catalunya. His current research is focused on machine learning approaches for big data analytics.

Luca Oneto was born in Rapallo, Italy in 1986. He is currently a Researcher at University of Genoa with particular interests in Machine Learning and Statistical Learning Theory. He received his Bachelor Degree in Electronic Engineering at the University of Genoa, Italy in 2008. He subsequently started his master studies in Electronic Engineering in the same university with focus in Intelligent System and Statistics. In 2014 he received his Ph.D. in School in Sciences and Technologies for Knowledge and Information Retrieval (University of Genoa).

Albert Samá was born in Barcelona, Spain in 1985. He received his M.Sc. degree in control and robotics in 2009 and B.Sc. degree in Computer Science in 2008 at the Universitat Politécnica de Catalunya. He completed a Ph.D. in Control, Vision and Robotics in the Automatic Control Department of the same university in 2013. His research interests include machine learning, signal processing and computational intelligence.

Alessandro Ghio was born in Chiavari in 1982. He received the Master degree in Electronic Engineering from the University of Genoa and obtained the PhD degree in 'Knowledge and Information Science' at the University of Genoa in 2010. His main research activities concern both theoretical and practical aspects of smart systems based on Computational Intelligence and Machine Learning methods. Currently, he also works as an IT consultant.

Xavier Parra is a full professor at the Automatic Control Department of the Universitat Politécnica de Catalunya. He received his degree in Computer Science in 1992 and a PhD in Computer Engineering in 2002, both at the same university. He is member of the Knowledge Engineering Research Group since 2000 and research manager at the Technical Research Centre for Dependency Care and Autonomous Living (CETpD) since 2005. He has a background in artificial intelligence, ubiquitous computing and machine learning. At present his research activity is mainly focused in the field of human activity recognition using inertial sensors.

Davide Anguita received the 'Laurea' degree in Electronic Engineering and a Ph.D. degree in Computer Science and Electronic Engineering from the University of Genoa, Genoa, Italy, in 1989 and 1993, respectively. After working as a Research Associate at the International Computer Science Institute, Berkeley, CA, on special-purpose processors for neurocomputing, he returned to the University of Genoa. He is currently Associate Professor of Computer Engineering with the Department of Informatics, Bioengineering, Robotics, and Systems Engineering (DIBRIS). His current research focuses on the theory and application of kernel methods and artificial neural networks.