Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 5126816, 17 pages
http://dx.doi.org/10.1155/2016/5126816

*Research Article*

# A Mobile Application for Easy Design and Testing of Algorithms to Monitor Physical Activity in the Workplace

**Susanna Spinsante,[1] Alberto Angelici,[1] Jens Lundström,[2] Macarena Espinilla,[3] Ian Cleland,[4] and Christopher Nugent[4]**

[1]*Dipartimento di Ingegneria dell'Informazione, Universita' Politecnica delle Marche, 60131 Ancona, Italy*
[2]*School of Information Technology, Halmstad University, Kristian IV:s väg 3, 301 18 Halmstad, Sweden*
[3]*Department of Computer Science, University of Jaen, Campus Las Lagunillas, s/n, A3-118, 23071 Jaen, Spain*
[4]*Computer Science Research Institute, University of Ulster, Newtownabbey, Ulster BT37 0QB, UK*

Correspondence should be addressed to Susanna Spinsante; s.spinsante@univpm.it

Received 25 March 2016; Revised 19 June 2016; Accepted 10 July 2016

Academic Editor: Ivan Ganchev

This paper addresses approaches to Human Activity Recognition (HAR) with the aim of monitoring the physical activity of people in the workplace, by means of a smartphone application exploiting the available on-board accelerometer sensor. In fact, HAR via a smartphone or wearable sensor can provide important information regarding the level of daily physical activity, especially in situations where a sedentary behavior usually occurs, like in modern workplace environments. Increased sitting time is significantly associated with severe health diseases, and the workplace is an appropriate intervention setting, due to the sedentary behavior typical of modern jobs. Within this paper, the state-of-the-art components of HAR are analyzed, in order to identify and select the most effective signal filtering and windowing solutions for physical activity monitoring. The classifier development process is based upon three phases; a feature extraction phase, a feature selection phase, and a training phase. In the training phase, a publicly available dataset is used to test among different classifier types and learning methods. A user-friendly Android-based smartphone application with low computational requirements has been developed to run field tests, which allows to easily change the classifier under test, and to collect new datasets ready for use with machine learning APIs. The newly created datasets may include additional information, like the smartphone position, its orientation, and the user's physical characteristics. Using the mobile tool, a classifier based on a decision tree is finally set up and enriched with the introduction of some robustness improvements. The developed approach is capable of classifying six activities, and to distinguish between not active (sitting) and active states, with an accuracy near to 99%. The mobile tool, which is going to be further extended and enriched, will allow for rapid and easy benchmarking of new algorithms based on previously generated data, and on future collected datasets.

## 1. Introduction

The ubiquity of smartphones together with their ever increasing computing, networking, and sensing capabilities has changed the landscape of people's daily life. Among others, activity recognition, which takes the raw sensor readings as input and predicts a user's activity, has become an active research area in recent years [1–3]. Activity recognition aims to understand the actions and goals of one or more humans, from a series of observations on their actions and the environmental conditions. Indeed, Human Activity Recognition (HAR) has become a task of great interest, especially for medical, military, and security applications. For instance, patients with diabetes, obesity, or heart disease are often requested to perform a well-defined physical training as a part of their treatment. Therefore, the ability to automatically recognize activities such as walking, running, or resting becomes a powerful tool, to encourage the patients and to provide feedback on their behavior to the caregivers. Application areas for HAR include [4] daily life monitoring [5–8], personal biometric signature [9], elderly and youth care [10–12], and localization [13, 14].

A necessary prerequisite for systems aimed at stimulating physical activity (PA) is to have monitoring capabilities enabled by HAR. The importance of promoting PA among people, through virtual coaching, is motivated by recent research outcomes that correlate sedentary behaviors with "an elevated risk of diabetes, cardiovascular disease, and all-cause mortality" [15]. Worsening of other health conditions, like metabolic syndrome, type-2 diabetes mellitus, and obesity, is also strongly associated with increased inactivity. Unfortunately, modern workplaces are typically populated by almost inactive adults who spend several hours sitting [16], and a 2-hour increase of this kind of "occupational" inactivity has been related to a 5–7% increase of the health risks highlighted above [17]. Only a small part of the adult population (18 to 64 years old) in developed countries meets the Global Physical Activity (GPA) guidelines, recommending at least 150 mins of "moderate to vigorous" PA per week. Weight-gaining, up to obesity, is another side effect of a lazy lifestyle: in addition to medical costs, it also causes relevant economic losses, due to missed working hours, decreased productivity, and disability [18].

According to the previous discussion, replacing the sitting time spent at the workplace with low-intensity PA may help preventing chronic diseases. Some exotic solutions have been proposed, such as workstations that allow the worker to stand or walk, using a specially designed standing or treadmill desk [19]. Stimulating PA through a virtual coach may be a feasible solution, and, to this aim, a precise monitoring of daily activity in the workplace is an extremely important task.

This work presents a mobile application, called *Actimonitor Android*, developed as a tool for rapid and easy testing of algorithms designed to accurately monitor the daily activity in the workplace. The accelerometer sensor on-board mainstream smartphones is used, and the feasibility of implementing even complex HAR systems on a smartphone is demonstrated. The tool is first developed and tested in an offline learning phase. Afterwards, it is executed on a mobile platform. Typical smartphone-related constraints, such as available computational resources, memory, and battery power, raise specific challenges for high-demanding mobile applications, like HAR, that requires feature extraction, classification, and transmission of relevant amounts of raw data. Moreover, current open source machine learning (ML) application programming interfaces (APIs), such as the Waikato Environment for Knowledge Analysis (WEKA) [20] and Java Data Mining (JDM), are neither designed, nor optimized, to run with full functionality on mobile platforms. Thus, a relevant problem addressed in this work is the mobile implementation of a HAR system, meeting response time, and energy consumption requirements.

The paper is organized as follows: Section 2 introduces the HAR problem, discussing the role of sensors and the state-of-the-art algorithms for activity recognition. In Section 3 the datasets, tools, and methodologies used for experiments are presented, with the mobile application developed for HAR algorithms design. Experimental results are discussed in Section 4; finally, Section 5 concludes the paper.

## 2. Human Activity Recognition

*2.1. Review of Literature.* A classic ML approach is adopted in HAR systems, in which classification is performed upon features extracted from raw sensor data, properly collected, preprocessed, and arranged into time-based segments. From data to features, an abstraction process takes place, based on which statistical or frequency-domain properties capture sensible information over each data segment, to feed a classifier. A selection of features may be necessary to reduce the data dimension handled by the classification algorithm that is designed on a *training* data subset and evaluated on a *testing* data subset.

Most of the research on HAR through mobile devices has been carried out using sensor data collected from smartphones but subsequently processed offline by means of ML toolboxes, such as WEKA [20]. As previously mentioned, smartphones have been traditionally considered as devices with limited resources, in terms of computational processing and battery lifetime [21]. While it is still important to consider these limitations when developing HAR systems for smartphones, such devices have become increasingly capable of running complex HAR in real-time. Nevertheless, challenges still remain in the evaluation of HAR solutions, particularly across the wide variety of hardware and software components now available. While a wide range of studies have reported and reviewed offline HAR (e.g., [22, 23]), just a few ones have fully implemented HAR on mobile phones for real-time processing [24]. This should include sensing, preprocessing, and classification, all carried out locally on the device.

Data provided by an accelerometer and a gyroscope on-board an Android smartphone carried in a pocket have been used by Dernbach et al. [25], to recognize simple actions (sitting, walking, running, and standing) and even more complex activities (cleaning, cooking, washing hands, and taking medication). Recognition of simple actions has been attained with a 93% accuracy, by a Multilayer Perceptron classifier and a two-second time window. The inclusion of complex activities in the dataset reduced accuracy to 50%, but still these results are promising for the current work, which seeks to identify simple physical activities such as walking, standing, and siting in an office environment, using data from a single smartphone sensor. Classification for this study was however carried out offline using the WEKA toolkit and therefore needs to be implemented and tested in real-time. In the paper by Kim et al. [26] a HAR solution was developed to assess physical activity and energy consumption in various buildings. This solution, developed for Android devices, recognized walking, climbing and descending stairs, running, and no movement. A Support Vector Machine (SVM) used data from accelerometer, gyroscope, and magnetometer to provide the classification, achieving high accuracies (98.26%).

Among the classifiers implemented and tested on mobile phones in the last few years, it is possible to mention Decision Trees (DT) [22, 23], SVM [27], *K*-Nearest Neighbor (*k*-NN) [24], and naïve Bayes [28]. Multilayer or hierarchical classification are obtained combining classifiers in different ways. Reddy et al. [29] combined a DT and a Dynamic Hidden Markov Model (DHMM), achieving an accuracy of

93.6% over a dataset from sixteen actors. In the majority of studies, classifiers are trained offline, using representative data, because training is computationally expensive and does not match real-time requirements. Then, the classification is implemented in real-time. Recently, Google released a real-time activity recognition API [30]; however, this is limited to motion-related activities (walking, cycling, and driving) and does not include static activities such as standing still or sitting, which are of interest in this work.

Real-time feedback to the user is another important aspect of both context aware and healthcare applications, particularly when trying to promote PA. However, in a large amount of studies this feature is missing [31]. A system developed by Lane et al. [28] provided real-time feedback through an animated user interface, reflecting the user's behavior, which is a slow motion for a static condition, and a more dynamic one for an increased activity. In this work we aim to stimulate the subject's PA by prompting, based on PA self-monitoring through the *Actimonitor Android* app.

The following sections provide details of the HAR classification process discussing current practices within the literature. These include data collection, preprocessing, feature extraction, and classification steps.

*2.2. Problem Definition.* Resorting to [32] and borrowing the same notation, the HAR problem (HARP) may be mathematically and formally defined, starting from sensor data collected and indexed over the time dimension and assuming nonsimultaneous activities:

*Definition 1* (HARP). Given a set $S = S_0, \dots, S_{k-1}$ of $k$ time series, each one from a particular measured attribute and all defined within time interval $I = [t_\alpha, t_\omega]$, the goal is to find a temporal partition $\langle I_0, \dots, I_{r-1} \rangle$ of $I$, based on the data in $S$, and a set of labels representing the activity performed during each interval $I_j$ (e.g., sitting and walking). This implies that time intervals $I_j$ are consecutive, nonempty, and nonoverlapping, such that $\bigcup_{j=0}^{r-1} I_j = I$.

The very large (or even infinite) amount of combinations of attribute values and activities and their generally unknown duration prevent a deterministic solution to the HARP and require the use of ML tools. A relaxed version of the problem is consequently introduced, in which time series are divided into fixed length time windows, as follows.

*Definition 2* (relaxed HARP). Given a set $W = W_0, \dots, W_{m-1}$ of $m$ time windows $W_i$ having the same size and being totally or partially labeled, such that each $W_i$ contains a set of time series $S_i = \{S_{i,0}, \dots, S_{i,k-1}\}$ from each of the $k$ measured attributes, and a set $A = \{a_0, \dots, a_{n-1}\}$ of activity labels, the goal is to find a mapping function $f : S_i \rightarrow A$ that can be evaluated for all possible values of $S_i$, such that $f(S_i)$ is as similar as possible to the actual activity performed during $W_i$.

The relaxation introduces some errors into the model, which are however negligible for most applications. A relevant approach in activity recognition is to combine the output of different models to produce more accurate predictions. This leads to multiclassifier systems, which are shown to be effective, at the expense of an increase in computational complexity. The formal definition of combining predictions from several learners is as follows.

*Definition 3* (HARP with multiclassifier). Given a classification problem with a feature space $X \in R_n$ and a set of classes $\Omega = \{\omega_0, \dots, \omega_{n-1}\}$, an instance $x \in X$ to be classified and a set of predictions $S = \{s_0, \dots, s_{k-1}\}$ for $x$, from $k$ classifiers, the goal of a multiclassifier system is to return the correct label $\omega^*$ iff $\exists s_i \in S \mid \omega^* = s_i$.

Some of the challenges faced in activity recognition are common to other fields too, but there are several specific issues for which dedicated computational methods have been developed. The recognition of highly diverse human activities requires selecting and combining several heterogeneous sensors that can be dynamically added or removed, based on application-driven requirements. Suitable metrics are finally defined to evaluate the HAR system performance. Table 1 summarizes the options in HAR system design and implementation.

Table 1: Main characteristics of HAR systems.

| Type | Characteristic |
|---|---|
| Execution | Offline |
| | Online |
| Generalization | User independent |
| | User specific |
| | Temporal |
| Recognition | Continuous |
| | Isolated |
| Activities | Periodic |
| | Sporadic |
| | Static |
| System model | Stateless |
| | Stateful |

*2.2.1. Activities.* Activities recognized from sensor data can be classified in different ways, for example, in terms of their complexity. A simple locomotion could be walking, jogging, walking downstairs, taking elevator, and so forth. Complex activities are usually related to a combination of actions (e.g., taking bus and driving) but may even correspond to the movements of certain body parts (e.g., typing and waving hand). Some activities may refer to the general context of healthcare, such as falling, exercise, and rehabilitations. Location-based activities include dining, shopping, and watching movies. Vision-based activities include leaving or entering a place. An IR sensor could detect a user moving or being still, whereas a home assisting robot could understand when the person is sleeping, taking pills, or doing cleaning [4, 33, 34].

Solutions developed for HAR must be robust to "intraclass" and "interclass" variability, the former occurring when the same activity is performed differently by different

Table 2: The different categories and types of activities in the current literature.

| Category | Activity type |
|---|---|
| Simple activities | Walking, jogging, sitting, standing, lying, walking upstairs, walking downstairs, jumping, taking escalator up, taking escalator down, taking elevator up, taking elevator down |
| Complex activities | Shopping, taking buses, moving by walking, driving a car, living activities, brushing teeth, vacuuming, typing, eating, cooking, washing hand, meditation, clapping, watering plants, sweeping, shaving, dry blowing the hair, washing dishes, ironing, flushing the toilet |
| Working activities | Working, relaxing, cleaning, on a break, meeting, home talking, home entertaining |
| Health activities | Exercising, fall, rehabilitation activities, following routines |

Table 3: Hardware (HW) and software (SW) sensors in Android [46].

| | Sensor | Description |
|---|---|---|
| HW | Accelerometer | Measures acceleration force applied to the device, including gravity |
| | Ambient temperature sensor | Measures ambient room temperature |
| | Magnetometer | Measures ambient geomagnetic field along three axes ($x; y; z$) |
| | Barometer | Measures ambient air pressure |
| | Gyroscope | Measures device rotation along three axes ($x; y; z$) |
| | Light sensor | Measures ambient light level (illumination) |
| | Proximity sensor | Measures proximity of an object relative to the screen of a device |
| | Humidity sensor | Measures humidity of ambient environment |
| SW | Linear acceleration | Measures acceleration force applied to the device (gravity excluded) |
| | Gravity sensor | Measures gravity force applied to the device, along three axes ($x; y; z$) |
| | Rotation | Measures the orientation of a device by providing the 3 elements of the devices' rotation vector |

individuals, or even by the same one in different times, and the latter due to data showing very similar characteristics even if belonging to fundamentally different classes. When not all the data in a continuous stream are relevant for HAR, the so-called NULL class problem may occur, which is difficult to model, as it represents a theoretically infinite space of arbitrary activities. A taxonomy of the most common activities targeted by HAR systems is summarized in Table 2.

*2.2.2. Sensors, Data Preprocessing, and Segmentation.* Sensors are the source for raw data collection in activity recognition, and they may be classified into three categories: video, environmental, and wearable sensors.

Wearable sensors are small size mobile devices designed to be worn on human body in daily activities. They can record users' physiological states, such as location changes, moving directions, and speed. Many wearable sensors are available on-board smartphones: Table 3 summarizes real (hardware) and virtual (software) sensors that are provided in current mainstream mobile devices [4, 34, 35].

Due to the intrinsic characteristics of accelerometers, the sensor orientation and the way the device is carried by the subject may heavily affect the raw data value. The most common positions of worn sensors used in the literature are hand-held, on the belt, in the pants pocket, and on the pelvic area. Sensitivity to orientation may be addressed by adding another sensor, through an aggregation technique.

Raw data collected from sensors are preprocessed, to reduce the effects of noise by means of filtering methods, like average smoothing. Additionally, preprocessing enables data synchronization when samples arrive from multiple sensors, or artifacts removal.

Preprocessing of wearable sensors signals like acceleration may involve calibration, unit conversion, normalization, resampling, synchronization, or signal-level fusion [36]. Data segmentation allows identifying segments of the preprocessed data streams that are likely to contain information about activities (*activity detection* or *spotting*). This is usually a critical step in HAR, due to the intrinsic complexity of separating and identifying activities that humans typically perform with no separation in time.

*2.3. Features and State-of-the-Art Algorithms.* Activity recognition relies on processing features that are extracted and selected from signals, through proper operations like conversion or transformation, to and from different domains. Feature computation may be automatic or derived from expert knowledge. A "feature space" is composed of the total number of features extracted from the data. If the extraction has been well executed, features corresponding to the same activity should appear clearly clustered in the space, and they should be clearly separated, if pertaining to different activities. Similarly, selected features are *good* if they are robust to intraclass variability and to different subjects performing the same activity. A wide range of features have been identified in the literature, according to the data type from which they are extracted. Among them, it is possible to mention the following: signal-based, body-model, event-based, and multilevel features. Another classification of the features is based on the domain to which the inspected data pertain, as detailed in Table 4.

Table 4: Common features used in HAR systems classified by domain.

| Type | Feature |
|---|---|
| Time and statistical | Mean, median, maximum, minimum, variance, standard deviation, cross-correlation, Root Mean Square (RMS), Signal Magnitude Area (SMA), Median Absolute Deviation (MAD), Time Between Peaks (TBP), max-delta between axis, skewness, kurtosis, displacement |
| Frequency | Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), energy, max-coefficient, mean-frequency, skewness, kurtosis, Interquartile Range (IRQ) |
| Structural | Autoregressive (AR) |
| Transient | Trend, Magnitude Of Change (MOC) |

Table 5: Taxonomy of classifiers proposed in state-of-the-art HAR systems.

| Type | Classifiers |
|---|---|
| Decision tree | C4.5, ID3, REPTree [47] |
| Bayesian | Naïve Bayes and Bayesian networks [48] |
| Instance based | $k$-Nearest neighbors [49] |
| Neural networks | Multilayer Perceptron [50] |
| Domain transform | Support Vector Machines [51] |
| Fuzzy logic | Fuzzy basis function, fuzzy inference system [52, 53] |
| Regression methods | MLR, ALR [33, 54] |
| Markov models | Hidden Markov Models (HMM), conditional random fields [55] |
| Classifier ensembles | Boosting and bagging [56, 57] |

In order to limit the computational complexity of the classification process and the amount of training data needed for parameter estimations, the feature space dimensionality should be kept at the minimum, by identifying the core set of features that still allows targeting the desired performance. This reduces also the memory and bandwidth requirements for real-time processing on embedded systems.

Once the most effective features are extracted, a fundamental part of a HAR system is the algorithm needed to classify new instances of recorded data [37]. The algorithm that outputs the classification label is represented by a model that has to be trained. Several inference methods have been proposed in ML and computational statistics, as listed in Table 5. In supervised ML algorithms, a function is inferred from a set of ground truth-labeled training examples, with the aim of minimizing the classification error and being able to map new examples (the testing ones).

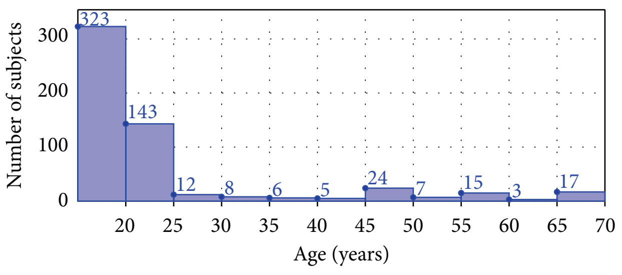


Figure 1: Age distribution among the subjects for the real-world recorded database.

## 3. Materials and Methods

*3.1. Datasets.* The lab and real-world recorded datasets, upon which this work is based, are taken from [38], used by Kwapisz et al. in [39], which contains the accelerometer data recorded with *Android* smartphones placed in the pants pocket, at an average sampling frequency of 20 Hz. This dataset is actually divided into two parts: a smaller one recorded in a controlled laboratory environment and a bigger one recorded and labeled by the users, in real-world settings.

The lab dataset was recorded using three types of smartphones: *Nexus One*, *HTC Hero*, and *Motorola Backflip*. The 36 volunteers performed a specific set of activities (walking, jogging, ascending and descending stairs, sitting, and standing for given periods of time) while carrying an *Android* smartphone in their front pants leg pocket. The total length in time of the dataset recordings is ≈15 h, corresponding to an average of ≈25 min recording for each user.

The real-world dataset was recorded and labeled freely by the users during their everyday life and without a specific protocol. For this dataset there is also some demographics information about almost all the 563 users (372 males and 191 females) involved in the test; they are summarized in Figures 1, 2, and 3. Among the users, 67 declared to have an injury affecting the way they walk. The total length in time of this dataset is ≈42 h. In this dataset, the *upstairs* and *downstairs* activities are grouped into *Stairs*, and a new activity is introduced, that is, *lying down*

Since both datasets are recorded with a smartphone, the sampling frequency is not regular and can vary during the recordings. The main problem is that the classifier, to express its best performances, has to work on features extracted from time windows with the same sampling frequency. Some of them can vary too much when the sampling frequency is

Table 6: Summary of the time windows and processing parameters.

| Parameter | Settings |
|---|---|
| Sampling frequency | 20 ± 2 Hz |
| Time window size | 128 samples (minimum 115, 90%) |
| Discarded windows | ≪1% |
| Filter | Butterworth 3rd order HP filter at 0.3 Hz |

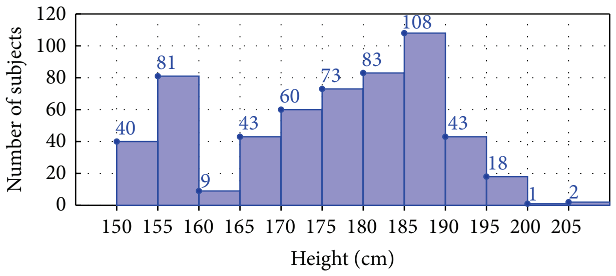


Figure 2: Height distribution among the subjects for the real-world recorded database.

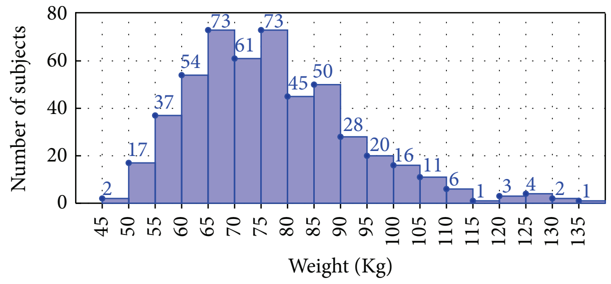


Figure 3: Weight distribution among the subjects for the real-world recorded database.

different, even if calculated for the same activity. For this reason all the time windows with a mean sampling frequency of ±2 Hz around the target value have been discarded.

Another aspect to consider is the windowing technique: since all the raw data are written sequentially on a text file, it is important to split them correctly, because the extracted time windows have to belong to the same user and the same activity. Since overlapping has not been used, the designed solution was to incrementally fill in the window during the file reading operation (line by line, since they are ordered in time) and truncate the window when an activity or user change is detected. All the windows with less than 90% of the target samples have been discarded. Table 6 summarizes the configuration used to process the datasets. The calculated features are then written on a file ready to be used with the WEKA toolkit. Figures 4 and 5 summarize the class distribution of the processed datasets, once the noncompliant time windows have been discarded.

*3.2. Training and Test Methods.* To train and test different classifiers, a common procedure has been used. Each classifier was selected through the *Classify* tab of the *WEKA explorer*, and a standard *10-fold cross-validation* has been used to obtain more reliable results.

No particular instance filtering techniques have been used, in fact all the windows for a certain activity have been used. The classifiers have been evaluated for the performances, the training time, the generated model interpretability, and the model file size. This last feature is relevant to the aim of implementing the classifier in a mobile application. In fact, once trained, the model has to be serialized, deserialized, and stored in the smartphone app. In particular, the performances have been evaluated via the most common indexes used in ML—*precision*, *recall,* and *F-score*— but also by analysing the obtained confusion matrices, to investigate possible classification problems and algorithm shortcomings.

*3.3. Implementation of the Mobile App for the Rapid Design of Physical Activity Monitoring Algorithms in the Workplace.* The main use cases for the developed *Actimonitor Android* application are tracking the user's PA and allowing recording and collecting new data, with the aim of creating and populating a new and more complete dataset, with respect to the one retrieved from the literature [38]. The mobile application is just a part of a bigger intended system, designed to collect data from many users, so it has to allow the management of a user profile too (setting password, email, and login). Moreover, the app is intended to be used also for synchronizing the data and collected and stored in the smartphone, to an online remote web server.

*3.3.1. Data Acquisition and Manipulation.* Classes dealing with data acquisition and manipulation are the most important ones, because they are responsible for retrieving data from sensors, calculating the features, classifying the instance, and writing the final results on the application internal database and on files. The main class that manages all the operations is the *BackgroundSensorsService* one, whose structure is shown in Figure 6 that extends a regular *Android Service* class. The class is created and started once the user activates the recording, and it runs in background performing the sensors data collection and activity tracking, even once the app is closed. The class can be started or stopped only through the app. It also performs the loading and deserialization of the WEKA trained model.

The tracking is active only when the screen is off and the proximity sensor detects a near object (i.e., the smartphone is recognized as being in the pocket). A separate thread is in charge of writing the collected data in files (both raw and processed data), managing the creation of the directories where the files will be stored, and updating the smartphone file system.

Performance of the app is an important requirement, in particular with respect to the real-time constraint. The problem is solved by splitting and running the processes in different threads; this results in a larger real-time margin and does not affect the smartphone stability.

*3.3.2. User Interface.* The app user interface is developed following the principle of *Material Design* [40]. The app allows the user to access functions from a top and a side selection menu (Figure 7). Clicking on an item in one of the two menus

| Processed lab dataset | | |
|---|---|---|
| Walking | 2,910 | 43.2% |
| Jogging | 1,724 | 25.6% |
| Upstairs | 743 | 11.1% |
| Downstairs | 618 | 9.2% |
| Sitting | 390 | 5.8% |
| Standing | 342 | 5.1% |
| Total time windows | 6,727 | 100% |
| Users | 36 | |

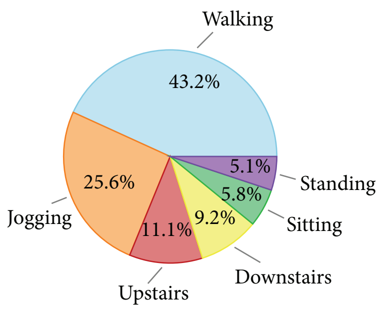


FIGURE 4: Class distribution for the laboratory recorded processed dataset.

| Processed RW dataset | | |
|---|---|---|
| Walking | 7,599 | 42.5% |
| Jogging | 2,079 | 11.8% |
| Stairs | 419 | 2.3% |
| Sitting | 4,083 | 22.8% |
| Standing | 1,947 | 10.9% |
| Lying down | 1,751 | 9.7% |
| Total time windows | 17,878 | 100% |
| Users | 563 | |

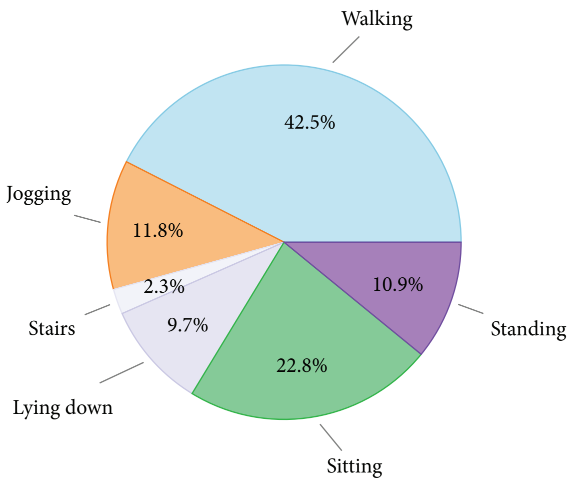


FIGURE 5: Class distribution for the real-world recorded processed dataset.

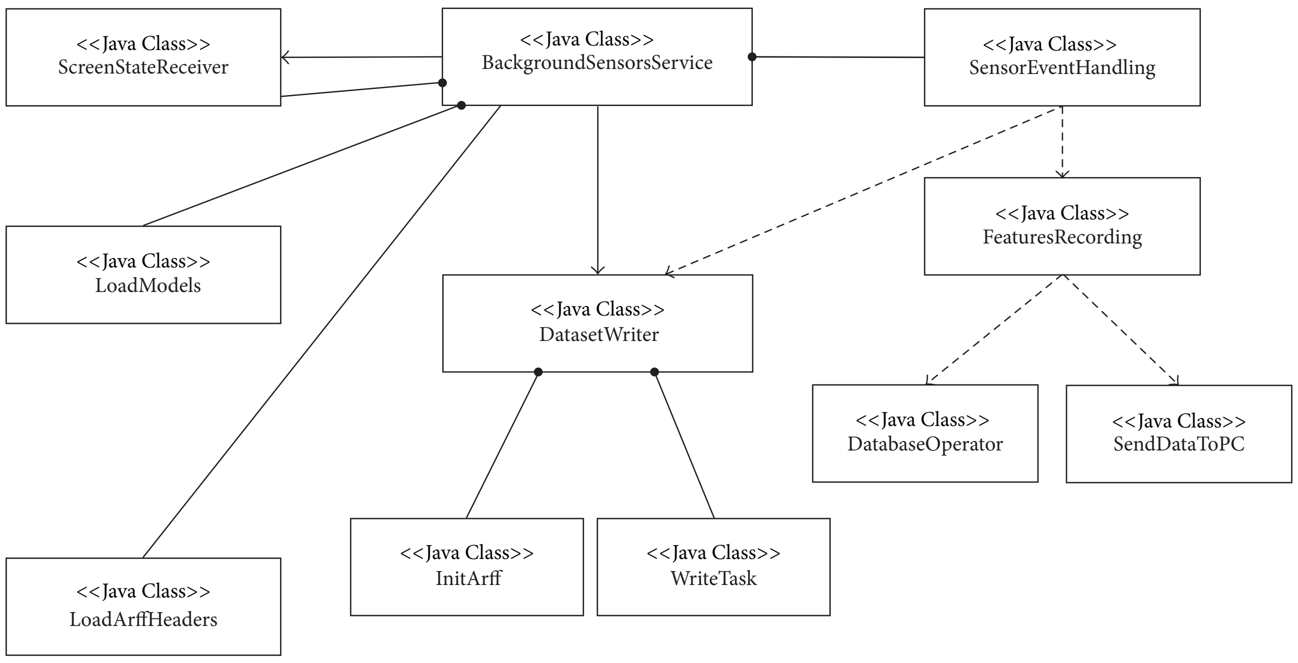
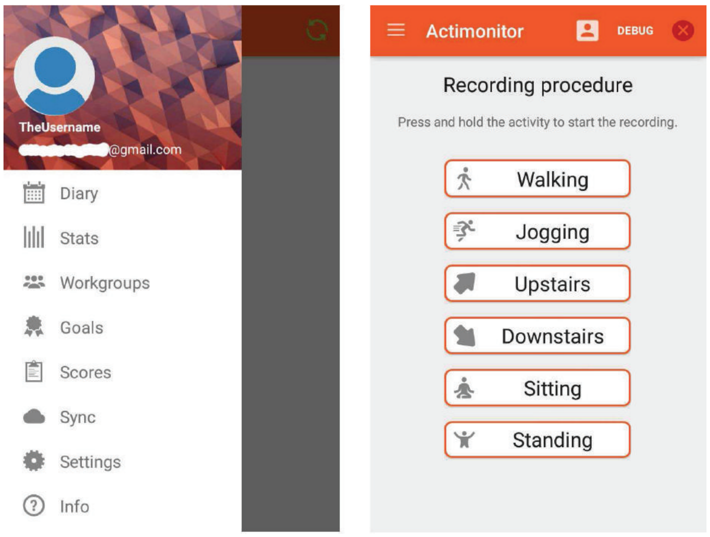

<<Java Class>> ScreenStateReceiver
<<Java Class>> BackgroundSensorsService
<<Java Class>> SensorEventHandling
<<Java Class>> FeaturesRecording
<<Java Class>> LoadModels
<<Java Class>> DatasetWriter
<<Java Class>> DatabaseOperator
<<Java Class>> SendDataToPC
<<Java Class>> InitArff
<<Java Class>> WriteTask
<<Java Class>> LoadArffHeaders

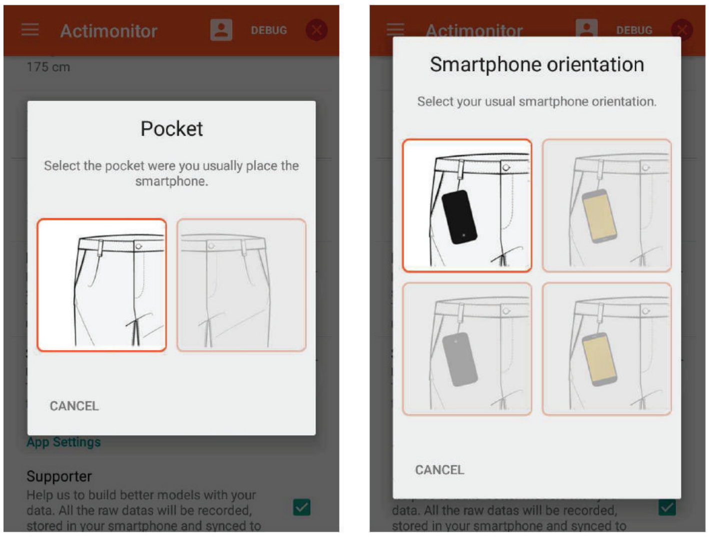
FIGURE 6: Data acquisition and manipulation classes of the background service (*BackgroundSensorsService* class).



Recording procedure

Press and hold the activity to start the recording.

FIGURE 7: Mobile app user interface.



Pocket

Select the pocket were you usually place the smartphone.

Help us to build better models with your data. All the raw datas will be recorded, stored in your smartphone and synced to

Smartphone orientation

Select your usual smartphone orientation.

data. All the raw datas will be recorded, stored in your smartphone and synced to

FIGURE 8: Selection of the smartphone position setting through the application interface.

starts a new interface for the desired function or visualization. In particular, in the current implementation, the side menu is used to access the settings of the app and the top menu provides all the actual functions. The different interfaces are implemented through independent *Android fragments* that replace each other.

The app is capable of classifying and tracking the activity of the user, using a previously trained classifier. The classifier is trained for the case of a user placing the smartphone in the trousers pocket, with the upper part of the device directed to the ground and the screen facing the leg. As a consequence, the user has to select the correct settings about the smartphone position, as shown in Figure 8. Then, by accessing the tracking section of the interface, the recording may be started, as shown in Figure 9. Afterwards, the user may put the smartphone in the pocket. The app will stop recording when it detects that the smartphone is not in a pocket anymore and will restart automatically when this condition is detected again. Also, a notification will appear on the smartphone locked-screen.

*3.3.3. Data Structures.* To accomplish all the requirements and easily manage the workflow of the system it was necessary to design some useful data structures, to hold both the data collected from the sensors and those generated by the app itself. Moreover, also the internal database structure has to be designed, to permanently store the tracking data in an efficient way.

The internal database is mainly used to store the tracking and the training data (raw data), in separate tables. One of the main problems to address was the growing size of the database during the everyday recordings. The database

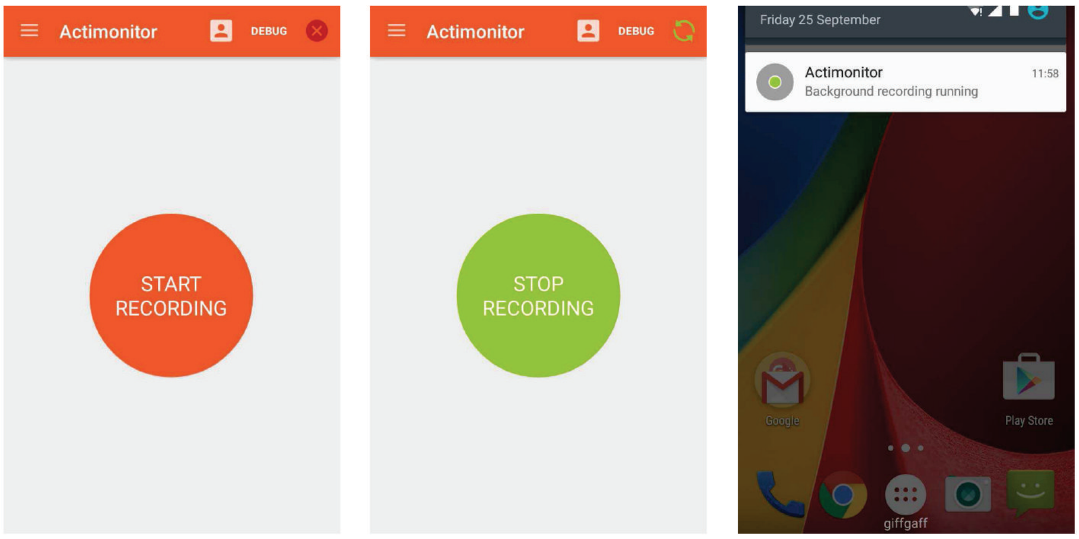


Figure 9: Activity tracking.

has been then designed to store only the essential data. In particular, the table to store the tracking data contains only the absolute timestamp of the window, the label, and a small set of information about it. For the raw data, every row contains a timestamp, the value for the three axes of the accelerometer (optionally, also the barometric value), and a small set of additional information; 128 of these rows are stored for each window. So not all the features calculated are stored, since they require a huge amount of memory and can be recalculated later in a simple way, if necessary.

The size growth of the database was estimated by leaving the app working for many hours, and it was possible to verify that, for the activity tracking only, over a standard 8-hour working day, the growing is of the order of hundreds of KBs, while, if saving the raw data, this value could rise to ten MBs. For these reasons it has been chosen to leave the user free to choose to store or not the raw data, via an option in the app settings. Moreover, this way, assuming a regular sync with a remote web server at least once a week, the database size does not yet represent a problem. Another problem to address is the time required to quickly write a group of tuples at once in the database. When using the helper classes provided by *Android* SDK, this operation is very slow ($\approx$1.5 s). For this reason, to store the 128 tuples of raw data per windows at once, a low level approach based on the *SQLite* JDBC driver was adopted. This way, it was possible to lower the time to only $\approx$100 ms.

*3.3.4. Classification System Implementation.* The features extraction phase is implemented by four classes (one for each group of features), to easily choose only a subset of them if necessary. These classes are called *TimeDomainFeatureExtractor*, *FrequencyDomainFeatureExtractor*, *Structural Feature Extractor* and *TransientFeatureExtractor*. They all extend the abstract class *FeatureExtractor*. The computed features are stored, progressively for each subgroup, in the main *FeatureSet* class instance.

The classification phase is the one for which the WEKA library has been used more extensively. Once the *FeatureSet* from the target time window is obtained, it is necessary to translate it into a WEKA *Instance*, which represents the single example to be classified. This has been done through the method *toInstance(Instances es)* of the *FeatureSet* class. This method takes as input a WEKA *Instances* class (a *Java* interface for a dataset and its instances, to be not confused with the *Instance*). In particular, the idea is to provide a dummy (empty) dataset with the same header of the one used to train the WEKA model. This way the software can check if the *FeatureSet* contains all the features required, and throw an exception if not. The *Instances* class representing the dummy dataset is obtained by reading a simple file stored in the app resources.

Since both the dummy dataset and the model are stored in files, they have to be loaded at the startup of the app (precisely, at the startup or the *BackgroundSensorsService*). The WEKA tool provides helper classes for this kind of operations. It is possible to obtain a representation of the dataset by means of an *Instances* class. Moreover, to deserialize the model, the *read* method of the helper class *SeralizationHelper* has been applied: by reading the model file, it gives back the already trained *Classifier* class implementation for the stored model.

The classification is simply done by the classifier's method *classifyInstance (Instance is)*, which takes as input the instance to classify and gives as output a *double* value ranging from 0 to $n-1$, where $n$ is the number of classes. Algorithm 1 shows a simplified implementation of the classification process adopted, which clarifies the use of the already mentioned classes.

## 4. Results and Discussion

*4.1. Base Classifier Selection.* A first step in the classifier selection was the choice of the base one to start from. Therefore, before moving to the implementation of the mobile app,

```
Data: dataset header file, serialized classifier, unclassified time window
Result: classified time window
load dataset header;
get attributes position and properties;
deserialize WEKA trained model;
while true do
    collect data from sensors;
    if time window ready to process then
        create valid instance from feature set data;
        classify instance;
        set corrisponding label;
    end
end
```

Algorithm 1: Simplified classification process in the smartphone app implementation.

Table 7: Deafult parameters for base classifiers: $k$-NN, NN, and DT.

| Classifier | Parameters |
|---|---|
| $k$-NN | Search algorithm: linear search with Euclidean distance |
| | Number of neighbors: 25 |
| NN | Number of hidden layers: (# attributes + # classes)/2 |
| | Learning rate: 0.3 |
| | Momentum: 0.2 |
| | Maximum number of epochs to train through: 500 |
| | Attributes normalized between −1 and 1 |
| DT (REPTree) | No depth restrictions |
| | Minimum total weight of the instances in a leaf: 2 |
| | Minimum variance on all the data needed for splitting: 0.01 |
| | Pruning |
| | Number of folds used for pruning: 3 |

Table 8: Base classifiers accuracy by class.

(a) DT

| Precision | Recall | $F$-score | Class |
|---|---|---|---|
| 0.938 | 0.942 | 0.94 | *Walking* |
| 0.969 | 0.971 | 0.97 | *Jogging* |
| 0.72 | 0.705 | 0.712 | *Upstairs* |
| 0.7 | 0.699 | 0.7 | *Downstairs* |
| 0.984 | 0.972 | 0.978 | *Sitting* |
| 0.98 | 0.988 | 0.984 | *Standing* |
| *0.905* | *0.905* | *0.905* | *Weighted average* |

(b) $k$-NN

| Precision | Recall | $F$-score | Class |
|---|---|---|---|
| 0.959 | 0.993 | 0.976 | *Walking* |
| 0.99 | 0.992 | 0.991 | *Jogging* |
| 0.911 | 0.855 | 0.882 | *Upstairs* |
| 0.915 | 0.833 | 0.872 | *Downstairs* |
| 0.995 | 0.979 | 0.987 | *Sitting* |
| 0.988 | 0.991 | 0.99 | *Standing* |
| *0.961* | *0.962* | *0.961* | *Weighted average* |

(c) NN

| Precision | Recall | $F$-score | Class |
|---|---|---|---|
| 0.981 | 0.987 | 0.984 | *Walking* |
| 0.991 | 0.988 | 0.989 | *Jogging* |
| 0.896 | 0.913 | 0.904 | *Upstairs* |
| 0.906 | 0.869 | 0.887 | *Downstairs* |
| 0.99 | 0.982 | 0.986 | *Sitting* |
| 0.985 | 0.991 | 0.988 | *Standing* |
| *0.968* | *0.968* | *0.968* | *Weighted average* |

three of the most common algorithms used in the literature have been tested: DT, $k$-NN, and NN. In particular, the ones implemented in the WEKA classes have been used, all of them with the default parameters, summarized in Table 7. To train the classifier, all the time windows extracted from the laboratory recorded dataset (see Figure 4) have been used.

Performance evaluation considering the *weighted average F-score* shows that the NN classifier is the best among the three, with an index of 0.996. The difference in performance is mostly due to the misclassification of only a subset of activities. In particular, the most problematic ones are the *upstairs* and *downstairs* activities, often classified as simple *walking*. Even if the overall performance indexes are good for all the classifiers, as given in Table 8, the DT shows the worst performance for these two activities, with an *F-score* equal to 0.712 and 0.7, respectively. Also the confusion matrices given in Table 9 show a frequent misclassification among this subset of activities.

About the time required to train the models, the faster is $k$-NN with only 0.1 s, because it is a lazy classifier. The DT is also fast, with 2.88 s needed to train the classifier. The training process of the NN is much slower, as it needs 967.16 s. The model interpretability is another aspect to consider, in view of the mobile application tool development.

Table 9: Base classifiers confusion matrices.

(a) DT

| | Classified as | | | | | |
|---|---|---|---|---|---|---|
| | Walking | Jogging | Upstairs | Downstairs | Sitting | Standing |
| Walking | 2742 | 15 | 84 | 68 | 1 | 0 |
| Jogging | 27 | 1674 | 18 | 5 | 0 | 0 |
| Upstairs | 75 | 32 | 524 | 112 | 0 | 0 |
| Downstairs | 80 | 7 | 96 | 432 | 3 | 0 |
| Sitting | 0 | 0 | 4 | 0 | 379 | 7 |
| Standing | 0 | 0 | 2 | 0 | 2 | 338 |

(b) *k*-NN

| | Classified as | | | | | |
|---|---|---|---|---|---|---|
| | Walking | Jogging | Upstairs | Downstairs | Sitting | Standing |
| Walking | 2890 | 2 | 14 | 4 | 0 | 0 |
| Jogging | 4 | 1710 | 7 | 3 | 0 | 0 |
| Upstairs | 54 | 14 | 635 | 40 | 0 | 0 |
| Downstairs | 63 | 2 | 38 | 515 | 0 | 0 |
| Sitting | 0 | 0 | 3 | 1 | 382 | 4 |
| Standing | 1 | 0 | 0 | 0 | 2 | 339 |

(c) NN

| | Classified as | | | | | |
|---|---|---|---|---|---|---|
| | Walking | Jogging | Upstairs | Downstairs | Sitting | Standing |
| Walking | 2871 | 6 | 15 | 17 | 1 | 0 |
| Jogging | 12 | 1703 | 6 | 3 | 0 | 0 |
| Upstairs | 20 | 10 | 678 | 35 | 0 | 0 |
| Downstairs | 24 | 0 | 57 | 537 | 0 | 0 |
| Sitting | 0 | 0 | 1 | 1 | 383 | 5 |
| Standing | 0 | 0 | 0 | 0 | 3 | 339 |

Finally, since the selected model has to be serialized and stored in a smartphone app, also the file size is an important characteristic. Both the DT and the NN produce a model of ≈1 MB size, with a slightly bigger one for the NN. *k*-NN gives a model of ≈10 MB. In fact, the disadvantages with lazy learning include the large space required to store the entire training dataset. Moreover, particularly noisy data increase the needed set unnecessarily, because no abstraction is made during the training phase.

Figure 10 shows a summary of the characteristics of the three classifiers. The DT gives a good compromise for all the considered characteristics, traded off with performance. Then, this is the base classifier chosen to build up the final PA monitoring system, to be implemented in the mobile app tool.

*4.2. Activities Misclassification Problem.* As stated above, the main performance loss is due to the wrong classification of some activities. Classes in this subset will be called *small displacement activities*, in contrast with the *static activities* (*sitting* and *standing*) and the *big displacement activity* (*jogging*). Activities in this subset feature very similar characteristics and movements; then also the extracted features have similar values, leading to a weak and error prone model.

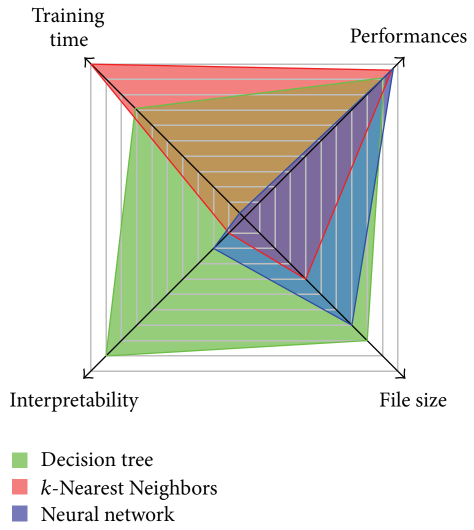
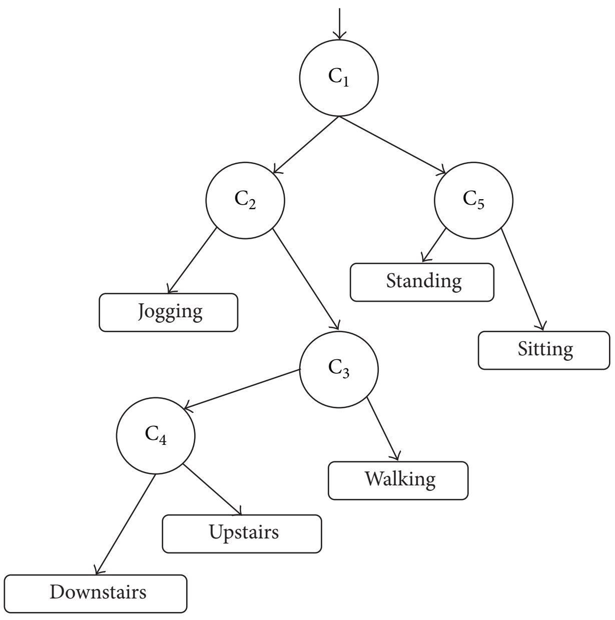
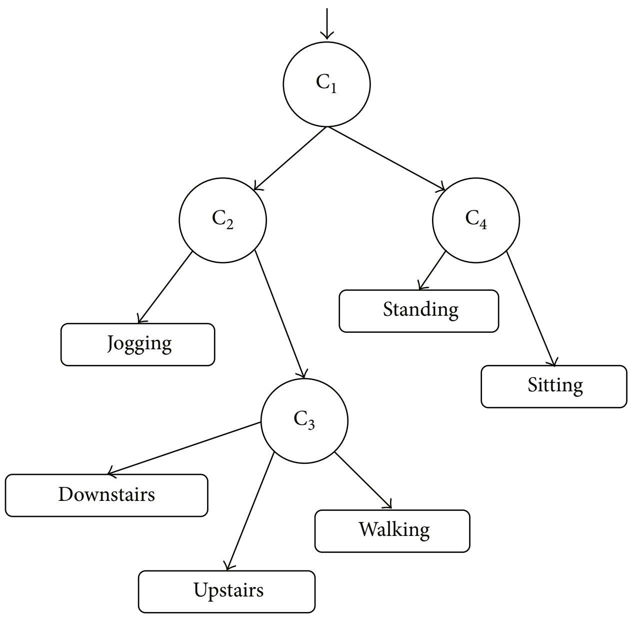


Figure 10: Summary of the characteristics shown by each base classifier.

Another problem, which comes out mainly in the test phase, is the wrong classification of the *slow walking activity*. This is probably due to the fact that the walking data was



(a) Model 1



(b) Model 2

Figure 11: Hierarchical models used to solve the misclassification problem.

recorded in a controlled setup, that is, on a treadmill at a fixed pace, and then a walking activity with a different or irregular pace sometimes is misclassified as *upstairs* or *downstairs*.

For these reasons, the need for a more accurate and stronger model not only to better distinguish between the *small displacement activities* but also to generalize the walking data and to recognize a larger range of paces clearly emerges. To produce a better classifier two solutions have been tested, a hierarchical approach and a generalized data approach, and one of them has been chosen.

*4.2.1. Hierarchical Approach.* The first approach tested is the hierarchical one, which allows splitting the original problem in multiple, smaller subproblems. The idea is to create simpler but stronger classifiers to select between few activities. In Figure 11, the two hierarchical schemes proposed and tested are shown, but only the results of Figure 11(b) are reported, because it performs better than Figure 11(a). The selected hierarchical classifier is based on simple DTs, the dimensions of which are related to the difficulty in classifying the target activities. As an example, the model used for *walking-upstairs* has an average number of nodes equal to 90, while the *jogging-standing* has only 3 nodes.

Considering that the main impact on performance was given only by the *small displacement activities*, the model that involves these three activities is the most important one, which affects the overall performance. Tables 10(a), 10(b), and 10(c) show the performance of this smaller classifier but, considering the *F-score* values, it is clear that there is not an improvement with a "smaller" model. The real problem is in the data chosen to train the classifier and not in the classification scheme used. In particular, they have to be generalized.

Table 10: Small displacement subclassifier performances.

(a) Performance by cumulative indexes

| | |
|---|---|
| Correctly classified instances | 86.65% |
| Incorrectly classified instances | 13.35% |
| Kappa statistic | 0.719 |
| Mean absolute error | 0.116 |
| Root mean squared error | 0.269 |
| Relative absolute error | 35.91% |
| Root relative squared error | 66.92% |

(b) Confusion matrix

| | | Classified as | |
|---|---|---|---|
| | Walking | Upstairs | Downstairs |
| Walking | 2787 | 65 | 58 |
| Upstairs | 105 | 515 | 123 |
| Downstairs | 96 | 123 | 399 |

(c) Detailed accuracy by class

| Precision | Recall | $F$-score | Class |
|---|---|---|---|
| 0.933 | 0.958 | 0.945 | *Walking* |
| 0.733 | 0.693 | 0.712 | *Upstairs* |
| 0.688 | 0.646 | 0.666 | *Downstairs* |
| *0.862* | *0.867* | *0.864* | *Weighted average* |

*4.2.2. Data Generalization Approach.* One of the solutions to generalize the training data is to use also the data from the real-world dataset, since they are collected by the users in real life conditions and can give a better and comprehensive model of the activities. Assuming the most problematic

TABLE 11: Classifier performance when trained with generalized *walking* data.

(a) Performance by cumulative indexes

| | |
|---|---|
| Correctly classified instances | 92.68% |
| Incorrectly classified instances | 7.32% |
| Kappa statistic | 0.829 |
| Mean absolute error | 0.034 |
| Root mean squared error | 0.143 |
| Relative absolute error | 23.16% |
| Root relative squared error | 52.71% |

(b) Confusion matrix

| | Classified as | | | | | |
|---|---|---|---|---|---|---|
| | Walking | Jogging | Upstairs | Downstairs | Sitting | Standing |
| Walking | 10221 | 15 | 125 | 116 | 15 | 17 |
| Jogging | 46 | 1662 | 7 | 9 | 0 | 0 |
| Upstairs | 253 | 33 | 385 | 70 | 0 | 2 |
| Downstairs | 231 | 6 | 53 | 328 | 0 | 0 |
| Sitting | 15 | 0 | 1 | 1 | 366 | 7 |
| Standing | 14 | 0 | 0 | 0 | 12 | 316 |

(c) Detailed accuracy by class

| Precision | Recall | *F*-score | Class |
|---|---|---|---|
| 0.948 | 0.973 | 0.96 | *Walking* |
| 0.969 | 0.964 | 0.966 | *Jogging* |
| 0.674 | 0.518 | **0.586** | *Upstairs* |
| 0.626 | 0.531 | **0.574** | *Downstairs* |
| 0.931 | 0.938 | 0.935 | *Sitting* |
| 0.924 | 0.924 | 0.924 | *Standing* |
| *0.921* | *0.927* | *0.923* | *Weighted average* |

activities are the ones related to walking actions and that the real-world dataset does not contain the *upstairs* and *downstairs* classes, only the *walking* windows have been used. Figure 12 shows the new class distribution, once the *walking* time windows taken from the real-world dataset have been included. Even if the class distribution is now strongly unbalanced, adding new time windows is necessary, to identify all the different walking patterns and paces as *walking*, as confirmed by experiments.

The new results provided in Table 11 show an improvement of the weighted average *F-score* value, from 0.864 to 0.923, but two classes are affected by a noticeable performance loss, that is, *upstairs* and *downstairs*. The reason is that some users collected the data by placing the smartphone with a different orientation compared to the laboratory dataset. This fact can give rise to misclassification and performance reduction for those activities particularly affected by the smartphone position and orientation. There is the need to strengthen the model and restore the classification performances on the *upstairs* and *downstairs* activities, but keeping the generalization yet introduced.

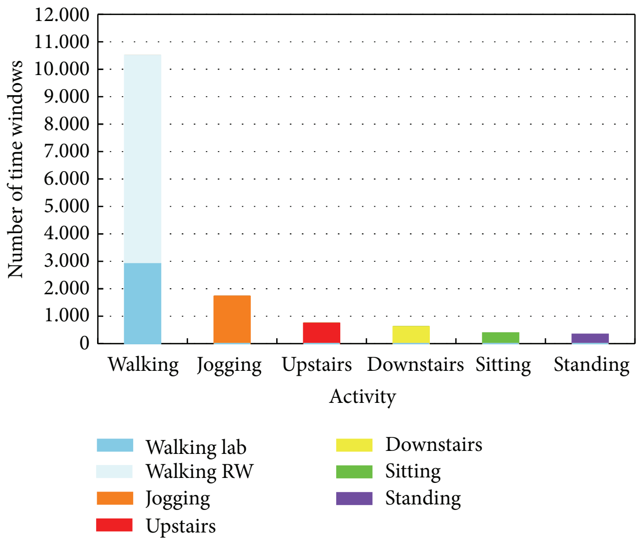


FIGURE 12: Resulting class distribution with the *walking* time windows taken from both laboratory and real-world dataset.

*4.3. Robustness Improvements.* The classifier has to be made more insensitive to the varying smartphone orientation and to possible labeling errors for the real-world dataset. It has to be also more insensitive regarding the way people use to perform the same activity but still retain the class separation properties. To this aim, bootstrap aggregating and pairwise classification have been used. The former identifies *ensemble learning* techniques, among which the *bagging* one was used and applied to DT classifiers. The latter decomposes the classifier into several two-class problems and combines their outputs through voting techniques (as for *bagging*). If the classes are evenly populated, a pairwise classifier is at least as quick to train as any other multiclass method [41, 42]. Moreover, since the DT performs an intrinsic feature selection and taking into account the difficulty of distinguishing the two selected activities, having a simpler two-class classifier allows creating models of acceptable complexity. The number of nodes in the DTs has the same order of magnitude as discussed before.

*4.4. Final Results on the Dataset.* Once the described methods and techniques are applied, it is possible to appreciate an overall performance improvement of the weighted average *F-score*, up to a value of 0.988. In particular, it changes from 0.712 and 0.7 to 0.676 and 0.71, for *upstairs* and *downstairs*, respectively.

An important perspective to analyse is the sensitivity of the algorithm to the smartphone orientation. To test this aspect, both datasets have been processed inverting the axis of the accelerometer and simulating the smartphone in a different orientation (upside down). Then the trained model has been tested with this data. As expected, the performance dropped down to an *F-score* of 0.729, due to *upstairs* and *downstairs*, but in particular for the *standing* class, where the orientation highly affects the classification.

For these reasons it was useful to create a reduced model with only two activities. Considering the goal of monitoring the PA in a workplace, the minimum target is to distinguish between *not active* (sitting) and *active* (all the other activities). Since the class distribution would become even too unbalanced, the *sitting* instances from the real-world dataset have been also added to the test and training set; the new class distribution is shown in Figure 13. This setup shows improved performances, according to Table 12, and achieves an average weighted *F-score* of 0.988, even for the test done with the upside down smartphone orientation. Given these results, the model is suitable to be implemented even when the smartphone orientation is not fixed, or there is no information about the orientation at all. A further solution could be to estimate the orientation from features that are not affected by orientation, like the acceleration magnitude [43–45], and then use a different classifier for each relevant position.

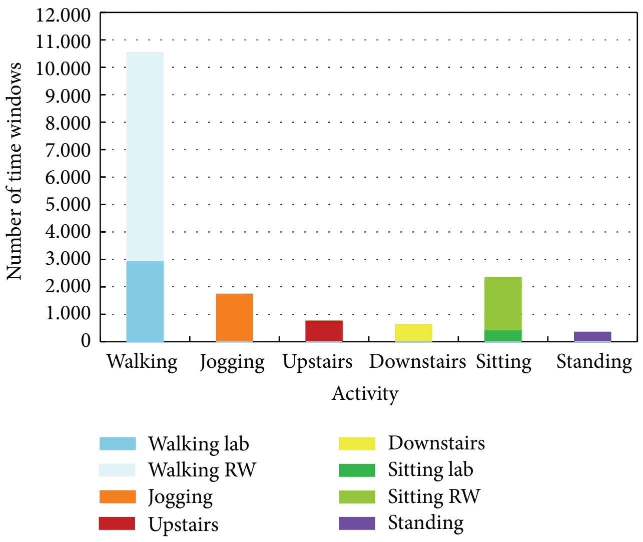


Figure 13: Resulting class distribution with the *walking* and the *sitting* time windows taken from both laboratory and real-world dataset.

Table 12: Final bagging *active*/*not active* classifier performance, also considering the same classifier with a different smartphone position (upside down).

(a) Performance by cumulative indexes

| | |
|---|---|
| Correctly classified instances | 98.82% (98.83%) |
| Incorrectly classified instances | 1.18% (1.17%) |
| Kappa statistic | 0.969 (0.968) |
| Mean absolute error | 0.019 (0.023) |
| Root mean squared error | 0.09 (0.1) |
| Relative absolute error | 5.26% (6.17%) |
| Root relative squared error | 21.94% (23.18%) |

(b) Confusion matrix

| | Classified as | |
|---|---|---|
| | Active | Not active |
| Active | 13853 (13825) | 83 (111) |
| Not active | 134 (105) | 4339 (4368) |

(c) Detailed accuracy by class

| Precision | Recall | *F*-score | Class |
|---|---|---|---|
| 0.99 (0.992) | 0.994 (0.992) | 0.992 (0.992) | *Active* |
| 0.981 (0.975) | 0.97 (0.977) | 0.976 (0.976) | *Not active* |
| *0.988 (0.988)* | *0.988 (0.988)* | *0.988 (0.988)* | *Weighted average* |

## 5. Conclusion

The aim of this research was to address the easy and rapid development and testing of classifiers to be used in physical activity monitoring systems, targeted at individuals in their workplaces. Such a result has been obtained through the design of an Android-based mobile application, which also demonstrates the feasibility of implementing even complex HAR solutions on a mainstream device like smartphones. The paper offered a complete overview of the activities classification problem from multiple points of view, starting from the types of sensors to use and their position to a taxonomy of the most used techniques in this field. A mobile app for HAR algorithms design has been presented, using *state-of-the-art* tools for machine learning, like the WEKA toolkit. It was then possible to offer an extensive performance evaluation of some of the most common classifiers. Afterwards, an analysis of the main problems which occurred was carried out. This led to the design of a system capable of overcoming these problems in a simple and effective way.

In particular, the implemented HAR system was designed to be used in a workplace environment to monitor the physical activity of the workers, equipped with the capability to distinguish between six activities. The chosen classifier is based on a hierarchy of DTs to which some improvements, like boosting and pairwise classification, have been applied, to increase the average performance. The final classifier reaches a weighted average *F-score* of around 92%. A simplified approach capable of distinguishing between active and not active states was studied, which provides much reduced sensitivity to the device orientation; it is capable of an *F-score* up to 99%. This last approach could be used, for example, in an elderly monitoring system, since it guarantees very high performance on the classification of simple activities, useful to detect the state of the patient in a room.

The final algorithm has been tuned by exploiting rapid test and development through an *Android* smartphone application developed ad hoc. All the software components in the app have been implemented following the *Android* design patterns and using the WEKA API as the core of the classification system. Such an app sets the basis for the design of new algorithms, since it allows a very easy replacement of the classifiers, only by changing the file in which they have been serialized. It could be a practical and easy tool for benchmarking new algorithms, even applicable and extensible to different domains compared to PA monitoring. The app is even provided with the capability of recording the collected sensors data, to create future new and richer datasets for HAR.

## Competing Interests

The authors of this paper declare that they have no conflict of interests.

## Acknowledgments

This work was supported by the COST Action IC1303 AAPELE, Architectures, Algorithms and Platforms for Enhanced Living Environments. The experimental part of the work was carried out in the framework of an Erasmus+ Traineeship between Italy and Ulster.

## References

[1] D. J. Cook and N. C. Krishnan, *Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data*, John Wiley and Sons, 2015.

[2] J. Aggarwal and M. Ryoo, "Human activity analysis: a review," *ACM Computing Surveys*, vol. 43, no. 3, article 16, 43 pages, 2011.

[3] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat, "Physical human activity recognition using wearable sensors," *Sensors*, vol. 15, no. 12, pp. 31314–31338, 2015.

[4] X. Su, H. Tong, and P. Ji, "Activity recognition with smartphone sensors," *Tsinghua Science and Technology*, vol. 19, no. 3, pp. 235–249, 2014.

[5] H. Alemdar, C. Tunca, and C. Ersoy, "Daily life behaviour monitoring for health assessment using machine learning: bridging the gap between domains," *Personal and Ubiquitous Computing*, vol. 19, no. 2, pp. 303–315, 2015.

[6] B. Bruno, F. Mastrogiovanni, and A. Sgorbissa, "A public domain dataset for ADL recognition using wrist-placed accelerometers," in *Proceedings of the 23rd IEEE International Symposium on Robot and Human Interactive Communication (IEEE RO-MAN '14)*, pp. 738–743, Edinburgh, UK, August 2014.

[7] T. R. Bennett, J. Wu, N. Kehtarnavaz, and R. Jafari, "Inertial measurement unit-based wearable computers for assisted living applications: a signal processing perspective," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 28–35, 2016.

[8] N. Roy, A. Misra, and D. Cook, "Ambient and smartphone sensor assisted adl recognition in multi-inhabitant smart environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 7, no. 1, pp. 1–19, 2016.

[9] T. Takeda, K. Kuramoto, S. Kobashi, and Y. Hata, "Biometrics personal identification by wearable pressure sensor," in *Proceedings of the 5th International Conference on Emerging Trends in Engineering and Technology (ICETET '12)*, pp. 120–123, IEEE, Himeji, Japan, November 2012.

[10] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, "A review of wearable sensors and systems with application in rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 9, no. 1, pp. 1–17, 2012.

[11] P. Bonato, "Wearable sensors and systems," *IEEE Engineering in Medicine and Biology Magazine*, vol. 29, no. 3, pp. 25–36, 2010.

[12] J. Merilahti, P. Viramo, and I. Korhonen, "Wearable monitoring of physical functioning and disability changes, circadian rhythms and sleep patterns in nursing home residents," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 3, pp. 856–864, 2016.

[13] G. A. Farulla, L. O. Russo, S. Rosa, and M. Indaco, "ORIENTOMA: a novel platform for autonomous and safe navigation for blind and visually impaired," in *Proceedings of the 10th IEEE International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS '15)*, pp. 1–6, Naples, Italy, April 2015.

[14] Y. Li, M. Liu, and W. Sheng, "Indoor human tracking and state estimation by fusing environmental sensors and wearable sensors," in *Proceedings of the IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER '15)*, pp. 1468–1473, Shenyang, China, June 2015.

[15] C. Tudor-Locke, J. M. Schuna Jr., L. J. Frensham, and M. Proenca, "Changing the way we work: elevating energy expenditure with workstation alternatives," *International Journal of Obesity*, vol. 38, no. 6, pp. 755–765, 2014.

[16] J. Y. Chau, M. Daley, S. Dunn et al., "The effectiveness of sit-stand workstations for changing office workers' sitting time: results from the Stand@Work randomized controlled trial pilot," *International Journal of Behavioral Nutrition and Physical Activity*, vol. 11, article127, 2014.

[17] F. B. Hu, T. Y. Li, G. A. Colditz, W. C. Willett, and J. E. Manson, "Television watching and other sedentary behaviors in relation to risk of obesity and type 2 diabetes mellitus in women," *Journal of the American Medical Association*, vol. 289, no. 14, pp. 1785–1791, 2003.

[18] X. Song, R. G. W. Quek, S. R. Gandra, K. A. Cappell, R. Fowler, and Z. Cong, "Productivity loss and indirect costs associated

with cardiovascular events and related clinical procedures," *BMC Health Services Research*, vol. 15, no. 1, article 245, 2015.

[19] B. T. MacEwen, D. J. MacDonald, and J. F. Burr, "A systematic review of standing and treadmill desks in the workplace," *Preventive Medicine*, vol. 70, pp. 50–58, 2015.

[20] WEKA (Waikato Environment for Knowledge Analysis), 2015, http://www.cs.waikato.ac.nz/ml/weka.

[21] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the USENIX Conference on USENIX Annual Technical Conference (USENIXATC '10)*, p. 21, USENIX Association, Boston, Mass, USA, June 2010, http://dl.acm.org/citation.cfm?id=1855840.1855861.

[22] H. Martín, A. M. Bernardos, J. Iglesias, and J. R. Casar, "Activity logging using lightweight classification techniques in mobile devices," *Personal and Ubiquitous Computing*, vol. 17, no. 4, pp. 675–695, 2013.

[23] A. Anjum and M. U. Ilyas, "Activity recognition using smartphone sensors," in *Proceedings of the IEEE 10th Consumer Communications and Networking Conference (CCNC '13)*, pp. 914–919, Las Vegas, Nev, USA, January 2013.

[24] M. Kose, O. Incel, and C. Ersoy, "Online human activity recognition on smart phones," in *Proceedings of the Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*, pp. 11–15, Beijing, China, April 2012.

[25] S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas, and D. J. Cook, "Simple and complex activity recognition through smart phones," in *Proceedings of the 8th International Conference on Intelligent Environments (IE '12)*, pp. 214–221, Guanajuato, Mexico, June 2012.

[26] T.-S. Kim, J.-H. Cho, and J. T. Kim, *Sustainability in Energy and Buildings: Proceedings of the 4th International Conference in Sustainability in Energy and Buildings (SEB '12)*, vol. 22 of *Smart Innovation, Systems and Technologies*, Springer, Berlin, Germany, 2013.

[27] A. M. Khan, A. Tufail, A. M. Khattak, and T. H. Laine, "Activity recognition on smartphones via sensor-fusion and KDA-based SVMs," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 503291, 14 pages, 2014.

[28] N. D. Lane, M. Lin, M. Mohammod et al., "BeWell: sensing sleep, physical activities and social interactions to promote wellbeing," *Mobile Networks and Applications*, vol. 19, no. 3, pp. 345–359, 2014.

[29] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Transactions on Sensor Networks*, vol. 6, no. 2, pp. 13:1–13:27, 2010.

[30] Google Activity Recognition API, 2016, https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi.

[31] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "A survey of online activity recognition using mobile phones," *Sensors*, vol. 15, no. 1, pp. 2059–2085, 2015.

[32] M. A. Labrador and O. D. L. Yejas, *Human Activity Recognition: Using Wearable Sensors and Smart-Phones*, Chapman & Hall/CRC, 2013.

[33] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys*, vol. 46, no. 3, article 33, 2014.

[34] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford, "A hybrid discriminative/generative approach for modeling human activities," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI '05)*, L. P. Kaelbling and A. Saffiotti, Eds., pp. 766–772, Professional Book Center, Edinburgh, UK, July-August 2005.

[35] J. Pärkkä, M. Ermes, P. Korpipää, J. Mäntyjärvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 1, pp. 119–128, 2006.

[36] D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. P. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Personal and Ubiquitous Computing*, vol. 14, no. 7, pp. 645–662, 2010.

[37] N. C. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Pervasive and Mobile Computing B*, vol. 10, pp. 138–154, 2014.

[38] WISDM Dataset, Department of Computer and Information Science, Fordham University, New York, NY, USA, 2012, http://www.cis.fordham.edu/wisdm/dataset.php.

[39] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2010.

[40] "Material design for android," 2016, https://developer.android.com/design/material/index.html.

[41] S.-H. Park and J. Fürnkranz, "Efficient pairwise classification," in *Machine Learning: ECML 2007*, pp. 658–665, Springer, Berlin, Heidelberg, 2007.

[42] C. Brunner, A. Fischer, K. Luig, and T. Thies, "Pairwise support vector machines and their application to large scale problems," *Journal of Machine Learning Research*, vol. 13, pp. 2279–2292, 2012.

[43] N. Yadav and C. Bleakley, "Accurate orientation estimation using ahrs under conditions of magnetic distortion," *Sensors*, vol. 14, no. 11, pp. 20008–20024, 2014.

[44] A. M. Khan, M. H. Siddiqi, and S.-W. Lee, "Exploratory data analysis of acceleration signals to select light-weight and accurate features for real-time activity recognition on smartphones," *Sensors*, vol. 13, no. 10, pp. 13099–13122, 2013.

[45] E. M. Shakshuki, A. Bayat, M. Pomplun, and D. A. Tran, "A study on human activity recognition using accelerometer data from smartphones," *Procedia Computer Science*, vol. 34, pp. 450–457, 2014.

[46] Sensors Overview in Android, 2016, https://developer.android.com/guide/topics/sensors/sensors_overview.html.

[47] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, Elsevier, Amsterdam, The Netherlands; Morgan Kaufmann, Boston, Mass, USA, 3rd edition, 2011.

[48] J. Cheng and R. Greiner, *Advances in Artificial Intelligence: 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, AI 2001 Ottawa, Canada, June 7–9, 2001 Proceedings*, vol. 2056 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2001.

[49] O. Sutton, "Introduction to k nearest neighbour classification and condensed nearest neighbour data reduction," 2012.

[50] J. Mäntyjärvi, J. Himberg, and T. Seppänen, "Recognizing human motion with multiple acceleration sensors," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 747–752, IEEE, Tucson, Ariz, USA, October 2001.

[51] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *Ambient*

*Assisted Living and Home Care*, pp. 216–223, Springer, Berlin, Heidelberg, 2012.

[52] M. Helmi and S. M. T. AlModarresi, "Human activity recognition using a fuzzy inference system," in *Proceedings of the 2009 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '09)*, pp. 1897–1902, Jeju Island, Republic of Korea, August 2009.

[53] J. A. Iglesias, P. Angelov, A. Ledezma, and A. Sanchis, "Human activity recognition based on evolving fuzzy systems," *International Journal of Neural Systems*, vol. 20, no. 5, pp. 355–364, 2010.

[54] B. Minor and D. J. Cook, "Regression tree classification for activity prediction in smart homes," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, pp. 441–450, ACM, September 2014.

[55] D. L. Vail, M. M. Veloso, and J. D. Lafferty, "Conditional random fields for activity recognition," in *Proceedings of the 6th ACM International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '07)*, pp. 235:1–235:8, Honolulu, Hawaii, USA, May 2007.

[56] A. Jurek, C. Nugent, Y. Bi, and S. Wu, "Clustering-based ensemble learning for activity recognition in smart homes," *Sensors*, vol. 14, no. 7, pp. 12285–12304, 2014.

[57] A. Dalton and G. Ólaighin, "Comparing supervised learning techniques on the task of physical activity recognition," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 1, pp. 46–52, 2013.