Context-aware Android applications through transportation mode detection technique

Article *in* Wireless Communications and Mobile Computing · July 2016

DOI: 10.1002/wcm.2702

3 authors, including:

Luca Bedogni
Università degli Studi di Modena e Reggio Emilia
74 PUBLICATIONS **1,150** CITATIONS
SEE PROFILE

Luciano Bononi
University of Bologna
160 PUBLICATIONS **3,377** CITATIONS
SEE PROFILE

WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2010; **00**:1–18
DOI: 10.1002/wcm

Context-Aware Android Applications through Transportation Mode Detection Techniques

Luca Bedogni*, Marco Di Felice, Luciano Bononi

Department of Computer Science and Engineering
University of Bologna, Italy
{lbedogni,difelice,bononi}@cs.unibo.it

ABSTRACT

In this paper we study the problem of how to detect the current transportation mode of the user from the smartphone sensors data, since this issue is considered crucial for the deployment of a multitude of mobility-aware systems, ranging from trace collectors to health monitoring and urban sensing systems. Although some feasibility studies have been performed in the literature, most of the proposed systems rely on the utilization of the GPS, and on computational expensive algorithms that do not take into account the limited resources of mobile phones. On the opposite, this paper focuses on the design and implementation of a feasible and efficient detection system that takes into account both the issues of accuracy of classification, and of energy consumption. To this purpose, we propose the utilization of embedded sensor-data (accelerometer/gyroscope) with a novel meta-classifier based on a cascading technique, and we show that our combined approach can provide similar performance than a GPS-based classifier, but introducing also the possibility to control the computational load based on requested confidence. We describe the implementation of the proposed system into an Android framework that can be leveraged by third-part mobile applications to access context-aware information in a transparent way.

1. INTRODUCTION

In 2013, just six years after the launch of the first iPhone device from Apple Inc., the number of smartphones in use worldwide has been estimated to overcome the one billion of units [7]. Beyond the straight-forward implications on the evolution of the Mobile Internet, we expect that the high complexity of these devices will fuel the proposal of novel ICT applications, business opportunities, and research studies. Indeed, today's smartphones integrate communication capabilities (through multiple wireless technologies, such as Wi-Fi, Bluetooth, NFC, etc) with computational and sensing functionalities, provided by a wide range of embedded sensors (e.g. accelerometer, gyroscope, GPS, etc). These enables new services, such as big data [6], the whole new world of Internet of Things (Iot), along with the web of things [4], and also the complex task of monitoring the traffic to classify it [5] [6]. Taking benefits from the pervasiveness of these devices and from the cooperation among users, people-centric urban sensing applications [8][14][15][1][2][3] are being progressively deployed in several domains (e.g. vehicular traffic [10][11], noise pollution monitoring, etc), providing larger coverage and higher resources than traditional static sensor networks [9]. At the same time, raw data from embedded sensors (e.g. accelerometer, gyroscope, etc) can provide useful information about the users' context, thus adding a new degree of context-awareness to the mobile applications [12][13]. How to collect, analyze and merge these (potentially big) data while guaranteeing the privacy and anonymity of the end-users constitutes a challenging and active research field in the area of mobile computing [16].

Transportation mode recognition techniques attempt to automatically identify the current vehicle used by the user (e.g. car, bus, train, etc) by analyzing the smartphone's sensors data, without any human feedback [31][33][34][35][36]. Knowledge of current transportation mode constitutes a precious information in several application domains. On the one hand, novel mobile services can be deployed with fine-grained context-aware functionalities, which for instance might enable the device to self-configure on the basis on the detected mode [29]. On the other hand, when multiple users share their context-related data in a participatory way, aggregated survey can be produced, giving indications about the life quality of the individual (e.g. health

Prepared using wcmauth.cls [Version: 2010/07/01 v2.00]

monitoring systems [38][39]) or of the collectivity (e.g. through the estimation of the environmental exposure and emissions [31]). From the technical point of view, research studies have demonstrated the possibility to recognize human activities from the analysis of wearable sensors data since 2000 [17][18]. As a result, the transportation mode recognition problem can be considered a sub-case of the activity classification problem, and the same methodology of study (e.g. supervised learning) can be applied with good results, as already discussed in previous works [35][36][37]. However, most of these studies focus on proposing, comparing and evaluating the performance of classification algorithms on synthetic environments, while less attention is posed on practical considerations about implementation. Indeed, many assumptions might not be feasible for a practical deployment of transportation mode detection systems on current today's smartphones:

- several proposed systems (e.g. [32][36][37]) utilize GPS data, which might not be available at specific locations, and also introduce a considerable impact on the battery lifetime of the device;
- machine-learning algorithms are adopted for the classification, and computational expensive techniques (e.g. Random Forest algorithm) are usually demonstrated to perform better in terms of accuracy. However, the CPU load becomes an issue when considering end-user mobile devices with limited resources;
- crowdsourcing techniques used in [31][32][36][37] demonstrate the generality of the detection process regardless of the users' habits, however they can be difficult to be implemented on a large-scale scenario, due to the need to consider incentive mechanisms to favor collaborative actions by users.

Addressing these issues constitute a trade-off between detection accuracy and energy consumption that can be summarized by the following question: how to take into account the specific computational/communication capabilities of each mobile device while guaranteeing an acceptable performance of the classification process?
In this paper, we attempt to answer to the previous question by designing and implementing an efficient transportation detection system which can be effectively used on today's smartphones. Our study include both methodological and practical contributions. About the methodological aspects, we remove most of assumptions of previous studies, and we show how to build a classification system that: (*i*) does not use GPS-classifier; (*ii*) dynamically adapts to the computational capabilities of the device, while providing the best energy-accuracy tradeoff; (*iii*) runs in a decentralized way without the need of collaborative training. To this aim, we first show -through experimental results- that aggregating the data from embedded sensors (i.e. accelerometer/gyroscope) can outperform the performance of a GPS-only classifier, by also discriminating among transportation modes with similar speeds (e.g. bus vs train). Then, we propose to combine multiple learners usually adopted in the literature of activity recognition systems through a novel cascading approach [40] which takes into account the requested confidence of the classification and the computational capabilities of a mobile device. As main result, we provide evidence of the fact that the multi-stage learner achieves higher accuracy than the individual learners (e.g. Random Forest), and provides performance comparable with a system utilizing both GPS/sensors data [36], but involving much lower energy consumption of the smartphone. Finally, we compare the case in which the training set is obtained by the collaboration of multiple users (sharing their data collected through heterogeneous devices) with the case where individual training is used (without cooperation), and we demonstrate that the individual training can perform equally or slightly better, thus justifying a decentralized deployment of the transportation mode recognition system. After having demonstrated the soundness and efficiency of our methodological approach, we describe how the proposed system can be practically implemented in a framework for the Android platform. The framework runs as a stand-alone Android application that samples the sensors' values in background, extracts the features from the accelerometer/gyroscope data, determines the current transportation mode through the cascading algorithm and exports this information at system-layer through a Content Provider. As a result, other Android applications can access and leverage this information to provide advanced context-aware functionalities, in a transparent way to our framework.

The rest of the paper is organized as follows. In Section 2, we discuss a list of use-case scenarios where transportation mode recognition techniques can be applied, and we further motivate the technical feasibility of the classification process. In Section 3 we review the existing studies pertaining to human activity detection from mobile devices sensors data. In Section 4, we detail our recognition system, by describing the methodology used for the training, the feature extraction and classification phases. Results about the accuracy of detection, energy consumption on mobile devices, and impact of training parameters are provided in Section 5. In Section 6 we describe the framework implementation over an Android platform, together with a brief presentation of a sample application (e.g. the Device Adapter one) that is built on top of the framework. Conclusions follow in Section 7.

2. MOTIVATIONS

In this Section, we discuss possible scenarios where automatic transportation mode recognition techniques can be applied (Section 2.1), and we also provide evidence of the fact that such classification can be performed on the basis of recognizable patterns associated to each mode (Section 2.2).

Wirel. Commun. Mob. Comput. 2010; **00**:1–18 © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm
Prepared using wcmauth.cls

2.1. Use Case Scenarios

Information about users' transportation modes can be useful both for real-time context-aware applications, that might adapt their functionalities to the detected mobility and for non real-time applications that might collect the mobility data to provide aggregate statistics and services, possibly relaying the collaboration among users through a mobile crowd-sourcing approach [15]. At the light of the existing prototypes, we foresee five different use case scenarios:

- *Mobility data collection*. In several areas of the world, surveys are used by transportation agencies and planning bodies to collect information about the urban transportation mobility, in order to improve the transportation system as a whole and to generate fine-grained and realistic traffic models. The utilization of automatic detection techniques (instead of questionnaires) can greatly simplify the data collection procedures, and also increase their capillarity, due to the intrinsic pervasiveness of smartphone devices. At the same time, knowing the transportation mode of end-users (together with their GPS traces) can be fundamental to detect critical situations of the urban mobility, such as congestion in vehicular or pedestrian traffic [10].
- *Health monitoring and user habits profiling*. In [31], the authors present a social platform that collects GPS traces from users and automatically detects the transportation mode, so that personalized reports about environmental exposure and impact (for instance, the daily carbon impact and smog exposure) can be returned to each user. Similarly, quantifying the physical activity duration and intensity during travel trips is considered of high interest to deploy health monitoring systems, as discussed in [38].
- *Device profiling and self-adaptation.* Transportation mode information can provide a further layer of device customization, since the end-users might associate a specific device profile to each mode. A profile can be defined in terms of hardware settings (e.g. turn on/off the ring-tones while walking) and/or as a set of actions to be performed once specific conditions are met, following the popular if-this-then-that approach (IFTTT) [42]. Moreover, the device can self-adapt its configuration to the detected mobility mode in order to prolong battery lifetime, for instance by dynamically turning off the GPS while in walking mode.
- *Enhanced mobile advertising.* As pointed out in [33], customized advertisements can be sent to the end-users, targeted to their actual mobility context. For instance, information about the presence of gas station in the neighbourhood can be delivered to car drivers, while real-time bus schedule information can be dynamically provided to users traveling on a bus.
- *Service adaptation.* Nowadays, several Internet services require the user to specify its transportation mode for content access. This is the case for instance of route planning applications (e.g. Google Maps) that requires the users to recalculate the path each time he/she changes the transportation mode. In case of dynamic detection, no human interaction is required, and the service content can be dynamically adapted to the user mobility.

2.2. Preliminary Data Insights

Transportation mode detection can be performed by using several different sensors and network data (e.g. accelerometer, GPS, GSM information, etc). Apparently, some modes can be easily distinguished by considering only the speed factor, like walking or driving a car [30][32]. However, the problem becomes more challenging when we consider both motorized and non-motorized modes, and different typologies of motorized modes. In Figure 1, we show the average speed of each mode, provided by the GPS. The data-set is composed of 5400 samples collected by 8 different participants, through heterogeneous Android devices and in heterogeneous environments (e.g driving a car in urban, rural and highway scenarios). Details about the data collection process are provided in Section 4. Figure 1 reveals that while non-motorized modes present low deviation from the average values, the speed of motorized ones (e.g. national bus and car) can fall into a wide range, with possible overlappings among different classes. In these cases, the speed values alone cannot be enough for an accurate transportation mode detection, as better demonstrated in Section 5.

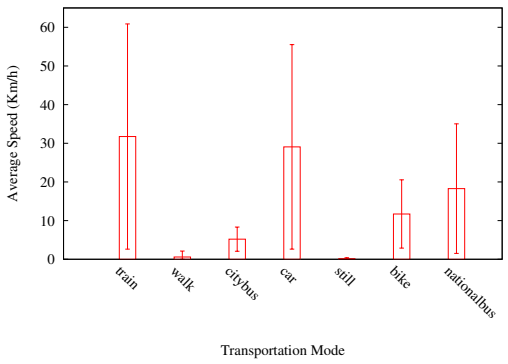
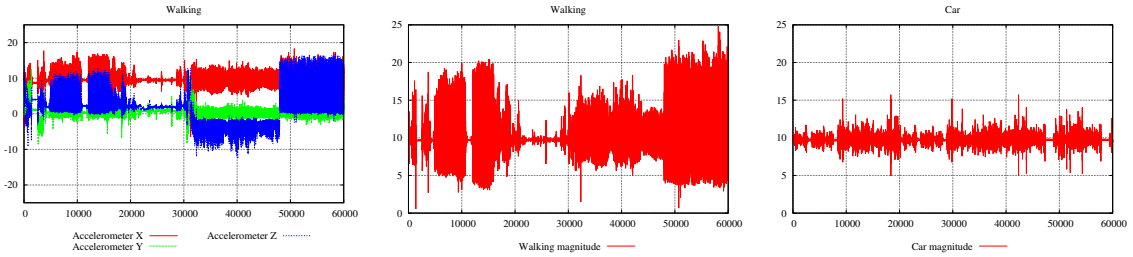


Figure 1. Average speed of different transportation modes.

For these reasons, an alternative approach is to measure the oscillations induced by each transportation mode through the accelerometer [17][21][37]. However, accelerometer values have the problem that they might depend on the specific orientation of the device. This is confirmed by Figure 2(a), where we show the three accelerometer values over time (one line for each axis), in a scenario where the user is walking and he is dynamically changing the position of the device, always



(a) Accelerometer (walking) (b) Magnitude (walking) (c) Magnitude (car)

Figure 2. The raw accelerometer values (on the x, y,z axes) and the magnitude of the *waking* mode are shown in Figures 2(a) and 2(b), respectively. The magnitude of the *car* mode is shown in Figure 2(c).

carried in the pocket of his trousers. It is easy to see that high fluctuations are introduced on each axis, as a consequence of the orientation changes. To overcome this problem, accelerometer synthesization techniques have been proposed. In this work, we rely on the magnitude metric [23][29], which is computed as follows:

$$magnitude(s) = |v_s| = \sqrt{v_{x,s}^2 + v_{y,s}^2 + v_{z,s}^2} \quad (1)$$

where $v_{x,s}$, $v_{y,s}$, $v_{z,s}$ denotes the values of sensor s on the x, y and z axes. Figure 2(b) shows the magnitude values for the accelerometer data of Figure 2(a). Conversely to the previous case, the magnitude values are almost stable and are not affected by the orientation changes. Moreover, in Figure 2(c) we show the magnitude values when driving a car. Even from these raw data, it is easy to see that there are qualitative and quantitative differences between the patterns of Figure 2(b) and Figure 2(c), that motivate the utilization of transportation mode recognition techniques.

3. RELATED WORKS

In the following, we review the existing literature on the analysis of smartphone sensors data for the extraction of context-related information. First, in Section 3.1 we consider the research works addressing the problem of human activity recognition through mobile devices. Then, in Section 3.2, we focus on the issue of transportation mode recognition, and we highlight the novel contributions and advances provided by our work with respect to the existing studies.

3.1. Activity Recognition

Human activity recognition from accelerometer samplings constitutes a well-investigated research area since 2000 [17]. In most of the existing works, researchers rely on a common methodology, i.e.: they build a training set of accelerometer samplings for each of the activities to recognize (i.e. walking, biking, running, jumping, etc), they extract features from the raw data, and they utilize data-mining techniques to classify the vector features. However, there exists significant differences in the hardware used for the experiments. In [18] [19] [20] [21] [22], the classification is performed through the utilization of wearable accelerometer devices, with fixed position and orientation. The results shown in [18] demonstrate that the utilization of multiple devices placed on different parts of the body might significantly reduce the classification errors caused by random noise. A comprehensive comparison of different classification techniques is reported in [21], where the authors show that combining classifiers through voting techniques produces the best results for the correct classification of most of the activities. Feature extraction from accelerometer data is discussed in [19][20][22]. More specifically, in [19][20] the authors propose to extract frequency-related patterns of the accelerometer data using discrete cosine transform. The same problem is also investigated in [22], where however the goal of the authors is to determine an efficient set of features that involve low computation efforts for the extraction/recognition processes. Despite the encouraging results in terms of classification accuracy, the utilization of wearable accelerometer makes extremely impractical the large-scale implementation of these systems. For these reasons, recent studies investigate the possibility to perform activity and gesture recognition through embedded accelerometer of smartphone devices [23][25][26][27][28]. Here, the main challenge is constituted by the fact that devices can be carried in different locations and styles. To solve this issue, in [24] the authors suggest techniques for orientation-independent features extraction and acceleration synthesization. In [23], a new metric (called *magnitude*) is introduced to compute the intensity of the acceleration, regardless of the smartphone's orientation [23]. The proposed metric is then used to recognize whenever a user is crossing a road, in order to produce a database of traffic lights for a specific urban environment. Results shown in [27][28] demonstrate that using embedded accelerometer it is possible to recognize human activities characterized by well-defined patterns (e.g. walking or running) with high classification accuracy (over 90%), while the classification of complex activities combining different motions (e.g. cooking) might be challenging.

Wirel. Commun. Mob. Comput. 2010; **00**:1–18 © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm
Prepared using wcmauth.cls

3.2. Transportation Mode Recognition

While most of the existing works focus on gesture and human motion recognition, there is also an increasing interest in transportation mode detection techniques that might enrich the context awareness of mobile applications [29] [30] [31] [32] [33] [34] [35] [36] [37]. From the algorithmic point of view, the problem is similar to the activity recognition described so far. However, the existing studies mainly differ in the sensors data required by the training and classification phases, and in the transportation modes they are able to recognize. GPS-only classifiers (like [30][31][32]) are shown to provide high accuracy in distinguishing between motorized and non-motorized modes, while they might fail in classifying motorized modes with similar speeds (e.g. bus, car, etc). In [31], the authors propose a participatory sensing application (called PEIR) which leverages the GPS location data to infer the transportation modes used along a path, and thus to compute personalized statistics of the environmental impact and exposure. Similarly, in [32], the authors propose a point-based segmentation mode to divide a path into separate segments, and a supervised classifier (based on decision trees) is used to decide the mode of each segment. In [33], the performance of a GPS-classifier are enhanced with transportation network information (such as bus schedules and bus stop locations), in order to extract features specific to each motorized transportation mode. More recent studies attempt to provide a fine-grained characterization of the environment through the utilization of accelerometer and wireless radio fingerprinting information [34][35][36][37]. In [35], the authors combine accelerometer readings with geo-location data (provided by the GPS and cellular network), and utilize a classifier based on Hidden Markov Model (HMM) to distinguish between nine transportation mode categories, also modeling the probability to switch modes along a path. At present, [36] constitutes the most exhaustive study on the topic of transportation mode recognition from smartphone data. The authors evaluate different feature selection strategies, classification algorithms and training techniques, and demonstrate that a GPS-classifier enhanced with accelerometer data can provide the best performance in terms of classification accuracy. In [38], the authors show that the utilization of accelerometer-data alone (without GPS) can provide lower accuracy than the combined case, but significantly prolonging the battery lifetime of the smartphones. Similarly, in [37], an accelerometer-based classifier is presented, and two different techniques to extract accelerometer features (e.g. synthesization and decomposition) are discussed. More recently, also Google released their activity awareness API [59], which might recognize if the user is still, if he is driving a car, walking or riding a bicycle. The API do not got into the technical details of how it is implemented, and on the website it is only stated that they use low-power sensors, in order to be optimized for the battery. In our study, we mainly follow the approaches described in [36][37]. At the same time, we provide these novel contributions compared to the existing literature:

- Conversely to [32][36][37], we do not rely on GPS information for transportation mode recognition. Instead, we utilize the embedded sensors of the smartphone (e.g. accelerometer and gyroscope), and we demonstrate that combining the sensors data can improve the accuracy of the classifier, while greatly reducing the energy consumption than a GPS-based classifier.
- Conversely to [37], we do not focus on a single classification algorithm. Instead, we integrate multiple classifiers through a cascading technique, by taking into account both the confidence requirements (defined by the end-user) and the hardware constraints of a device. We are aware that the accuracy/energy trade-off has been discussed also in previous papers [36][63][64]. However, we address the problem for a specific class of devices (e.g. smartphones), and we propose a solution to account the computational load during the classification process, which has not been considered in existing works.
- Conversely to [31][32], we do not rely on a centralized infrastructure for the training phase. The classification algorithm is integrated into an Android application, which can share the detected transportation mode information with other mobile applications installed on the device, in a seamless way (implementation details are provided in Section 6).

3.3. Prototypes

In literature it is possible to find a plethora of works targeting applications prototypes' for commercial smartphones. These includes:

- In [52] the authors present Nericell, a sensing application to determine the traffic conditions from sensor data. Nericell utilizes a 3 axis accelerometer sensor, without computing the magnitude of the received signal, but analyzing separate values for each axis. This allows to recognize road bumps and heavy brakes, for instance. They also use the microphone to sense the rumors and determine whether a road is particularly noisy, and thus potential consequence of heavy traffic jams. Finally, in order to reduce the battery consumption of their applications, triggered sensing is used. This results in using more intensively low battery demanding sensors such as the accelerometer, and turn on other sensors such as GPS and microphone only when something important is noticed by examining the accelerometer traces.
- In [53] the authors describe CenceMe, an application that uses sensors on smartphones to infer several information about the user behavior. They

use the accelerometer, the microphone, the GPS and the bluetooth modules. In additon, they also use the camera to take quick snapshot about the user current location and activity. Combining informations from these different sources, CenceMe can infer the location, the social context, the mobility model and the "sociability" of the user. Some data is processed locally to the smartphone, and then sent to a remote backend server in order to be analyzed. To reduce the battery consumption, the authors describe a reduced duty-cycle method. To classify the data, they use a decision tree algorithm, more precisely the J48.

- In [54] the authors describe The Mobile Sensing Platform (MSP), detailingthe milestones from version 1 to version 3. They are able to utilize seven different sensors, including microphone, accelerometer and temperature, in order to recognize the human activities. The focus is on non-motorized activities such as walking, running, watching TV and so on.
- In [55] the authors tackle the challenging problem of online classification using only the smartphone accelerometer. Instead of dividing the samples in time windows, they continuosly try to identify the action performed by the user. They implemented it on a smartphone and run experimental tests, showing that most complex activities can be detected within around 5-25% of their overall activity length. For long actions such as cooking or watching TV, which can last from 40 minutes to 2 hours, this mean recognizing the activity after it started from only 2 minutes.
- In [56] the authors use the accelerometer, the gyroscope and the magnetic sensor of a smartphone to recognize the user activity over a set of 15 possible activities. The focus is just on non-motorized activities such as walking, climbing stairs and standing still. They sample the data at 25 Hz, and compute different statistical data on the samples collected, namely the average, the median, the standard deviation, the skewness, the kurtosis, the interquartile range and the percentage of decline. Over all these computed features, the authors build several classifiers, connected through a hierarchical structure, which achieves an overall accuracy classification of 95.03%.
- In [57] the authors perform classification for different non-motorized activities of a human being by considering artificial neural networks (ANN) and support vector machines (SVM). They rely only on accelerometer and orientation sensors, and they show good performance in recognizing the user activity, even though the dataset used is relatively small.
- In [58] the authors present a novel method to recognize the transportation mode, and compare it with other well know approaches, such as [36] and [37]. They focus on using acceleroemeter data, in order to reduce the consumption of the GPS and its unreliability underground or in challenging situations. They build several classifiers, and make distinction between stationary behavior, non-motorized and motorized behaviors. Improvements against [36] and [37] in terms of higher accuracy are shown.

4. DATA COLLECTION AND ANALYSIS

In this Section, we detail the methodology used to collect and classify the transportation mode from the smartphone sensors data. Section 4.1 introduces the training process, while Section 4.2 and Section 4.3 describe the features' extraction and classification phases.

4.1. Data collection

As a first step toward the performance evaluation of a transportation mode recognition system, we developed an Android application that allows to sample the sensor values at a fixed rate r (set to 10 Hz), and to save each sample on a log file. Like previous studies, we limited our attention to a restricted set (i.e. M) of transportation modes:

$$M = \{m^S, m^W, m^C, m^T, m^{BK}, m^{Bc}, m^{Br}\} \quad (2)$$

with the following meanings:

- m^S is the pattern associated to the mode of *standing still*.
- m^W is the pattern associated to the mode of *walking*.
- m^C is the pattern associated to the mode of *driving a car*.
- m^T is the pattern associated to the mode of *being on a train*.
- m^{Bk} is the pattern associated to the mode of *driving a bike*.
- m^{Bc} is the pattern associated to the mode of *being on a city bus*.
- m^{Bn} is the pattern associated to the mode of *being on a national bus*.

Similarly, we considered a set S of three sensor types, i.e. S={*accelerometer* (Ac), *gyroscope* (Gy), *gps* (Gps)}. For each mode m, we built a data-set D_m containing on avarage 4500 samples (corresponding to around 6 hours and half of continuous sampling). Each entry of D_m has the following structure:

$$< t, v_{Ac}, v_{Gy}, v_{Gps} > \quad (3)$$

where t is the time-stamp of the sample, v_{Ac} and v_{Gy} are the magnitude values of the accelerometer/gyroscope (computed through Equation 1), and v_{Gps} is the current speed value, provided by the GPS. Each data-set D_m was

Wirel. Commun. Mob. Comput. 2010; **00**:1–18 © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm
Prepared using wcmauth.cls

built in an *heterogeneous* way, i.e. samples were collected from eight distinct people, using different hardware platforms. Moreover, we left each user free to carry and use the device at his taste during the experiments (i.e. we did not impose fixed orientations and locations of the wearable devices like in [18][21]).

To collect the data from different human beings, thus having different physical characteristics and thus probably different patterns, we run an experiment with 6 different people in order to gather the data to be processed.

In Table I we report the demographics for the samples collected. Clearly, not all the human beings could collect samples for all the different kind of transportation modes, so we report the actions collected by each individual on the last column. As it is possible to see, we performed the test with 5 males, marked with the M, and one female, marked with the F.

Name	Height (cm)	Weight (kg)	Age Age	Actions collected
M1	180	100	28	$m^{\{S,C,Bc,T,W\}}$
M2	175	75	33	$m^{\{S,Bn,T,W\}}$
M3	174	105	56	$m^{\{S,C,T,W\}}$
M4	170	68	24	$m^{\{Bk,C\}}$
M5	168	75	25	m^{Bk}
M6	166	80	22	$m^{\{C,T,W\}}$
F1	165	72	28	$m^{\{C,Bc\}}$
F2	169	57	29	$m^{\{Bk,C\}}$

Table I. Demographics for the samples collected

4.2. Feature extraction

After having collected the data, we divided each data-set D_m into consecutive non-overlapping time sequences of length T (equal to 5 seconds in our tests). From each sequence k, and sensor s, we extracted the following set of features:

- $min(s,k)$: this is the *minimum* value of sensor s over the sequence k.
- $max(s,k)$: this is the *maximum* value of sensor s over the sequence k.
- $avg(s,k)$: this is the *average* value of sensor s over the sequence k.
- $std(s,k)$: this is the *standard deviation* of sensor s over the sequence k.

In Section 5.3, we investigate the impact of the sequence length (i.e. T) on the accuracy of the classifiers. While several set of features can be used to represent a sequence, our choice is mainly motivated by the observation (supported by Figures 2(b) and 2(c)) that different transportation modes produce different time-behaviours of the sensor magnitude, in terms of mean and fluctuations between the peak values. Also, we highlight that our choice involves much lower computational costs than frequency-based feature extraction techniques [19][20].
We introduce here some notations used in the rest of the paper. We denote with F^s the set of features' values relative to sensor s (e.g. F^{Ac}), over all the data-set D_m, and for each mode m. Analogously, we denote with $F^{s_1+s_2+\dots s_h}$, the set of features' values associated to the combined utilization of sensors $s_1, s_2 \dots s_h$ (e.g. F^{Ac+Gy}).

Even though [18] states that it is better to use overlapping windows to classify data, we show in Figure 3 a comparison we run in order to justify our choice of using non overlapping windows. We show the accuracy for the different actions, and the overall accuracy, for four different window configurations, which are 1 and 5 seconds non overlapping windows, and 1 second with 0.1 seconds of overlapping windows, and 5 seconds with 1 seconds of overlapping. As it is evident to see from Figure 3, most of the gain in accuracy is given by a greater window size rather than an overlapped portion of the window. Moreover, for some specific actions (i.e. the m^C), the overlapped windows perform better than overlapped ones.

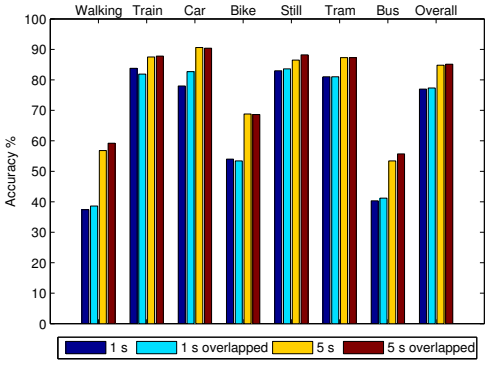


Figure 3. Comparison between overlapping and non overlapping windows.

4.3. Data classification algorithm

In its general definition, a transportation mode classification scheme l takes as input an instance x, composed by a set of features $F_k^\bullet$ for a sequence k and for a given combination of sensors data, and a training set, and produces as output a value $m \in M$, that represents the estimated transportation mode. Previous studies on transportation mode recognition [32][33][34][35][36][37] rely on the utilization of a single classification algorithm, which is often selected as the one providing the highest accuracy (e.g. Random Forest), but without taking into account the computational costs. Conversely, in our approach we attempt to reduce the energy consumption of mobile devices, while maintaining a pre-defined level of accuracy. For this reason, we combine multiple learners through cascading [40], which is a widely used machine learning technique to increase the overall accuracy while using relatively inaccurate (but simple) classification algorithms. Let $l_1, l_2, \dots l_K$ the set of learners used for the classification, ordered on the

basis on their computational costs, i.e. l_{j+1} is costlier than l_j. Let θ be the requested confidence of the transportation mode recognition scheme. In our case, θ can be defined by the end-user through the mobile application interface (Section 6). Reasonably, higher values of θ produce more accurate classifications, but might also introduce additional computational overhead. The classification scheme works through an iterative classification process over the set of K learners, as described by Algorithm 1. Let x be the new instance to be classified. At each step, learner l_j decides the detection mode m_{ij}, and computes its confidence over instance x (i.e. $w(l_j, x)$), calculated as the posterior probability of selecting mode m_{ij}, given learner l_j and instance x:

$$w(l_j, x) = max_i P(m_{ij}|l_j, x) \quad (4)$$

If the confidence $w(l_j, x)$ exceeds the requested threshold θ, then the algorithm ends, and m_{ij} is returned as the output of the classification. Otherwise, learner l_{j+1} is used, till the requested confidence is met or all the K algorithms are evaluated.

Algorithm 1: Cascading classification algorithm

```
Given: x (instance), l_1, l_2, ...l_K (learners), θ (threshold)
Set index j=0
repeat
    Set j=j+1
    Execute l_j and get m_ij // Execute the j-th learner on input x,
                                and return the classification m_ij
    Compute w(l_j, x) through Equation 4
until w(l_j, x) ≥ θ or j ≥ K
return m_ij
```

The training phase is done individually on each learner in an iterative way, as described in [41]. At the beginning, learner l_1 (i.e. the simplest one) is trained over all whole training set. At each step j, learner l_j is trained over the set of instances not learnt correctly by the learner l_{j-1}, i.e. it localizes on patterns rejected by previous algorithms, and also on the instances for which $w(l_{j-1}, x) < \theta$, i.e. on those instances for which previous algorithms were not confident. Since the learners are ordered on the basis on their increasing complexities, this approach guarantees the fact that costlier algorithms are trained (and then used) on complex patterns not recognized by the previous (simpler) algorithms [61] [62].

5. DATA CLASSIFICATION RESULTS

In this Section, we evaluate the performance of a transportation mode recognition system, by using the training set and the classification methodology previously described in Section 4. First, we investigate (in Section 5.1) the ability of traditional base-learners to recognize the transportation mode from different combinations of sensors data, considering the issues of accuracy, detection time, and power consumption. Then, in Section 5.2 we demonstrate the benefits of combining multiple learners through the cascading algorithm defined in Section 4.3. Finally, in Section 5.3, we analyze the impact of data acquisition methodologies and parameters used for training, such as time-sequence length and sensing rate.

5.1. Analysis of individual learners

From a machine-learning perspective, recognizing the current transportation mode from sensors data can be seen as an instance of a classification problem, and any supervised learning algorithm can be applied to solve it. However, since there is no single algorithm that induces the most accurate learner in any domain [40], it is worth to investigate which approach can fit the characteristics of our environment, like the fact that training data might exhibit noise, and that patterns relative to some transportation modes might be intrinsically more difficult to detect than others. Moreover, understanding which sensors data to utilize is a unique issue of our domain, since this choice might have a remarkable impact on the performance of the learner and of the mobile application.

Based on these considerations, we consider six base learners in our preliminary study: Random Tree (RT), Random Forest (RF) [43], Support Vector Machines (SVM) [46], Nayve Bayes (NB), Bayesian Network (BN) [47] and Decision Table (DT) [40]. These learners are also used by other works found in literature: [33] performs an evaluation among different learners, including BN, RT, and RF, eventually choosing the latter thanks to its performance. DT are insted used by [32] [33] [36] [53]. In Table II we report the overall accuracy of each learner by performing a 10-fold cross validation on our training set through the WEKA tool [48]. We repeat the experiments for every possible combinations of sensor data of the training set (i.e. F^{Ac}, F^{Gy}, F^{Gps}, F^{Ac+Gy}, etc), and then we report the average. Sensors data selection is discussed later in this Section. In Table II we also report the (*i*) building time, i.e. the time to generate the classification model on WEKA, and the (*ii*) classification time, i.e. the time to run the model and classify a new instance. Both these values are computed on a target smartphone (i.e. Google Nexus 4), and are expressed in percentage from the performance of the Random Forest algorithm. From the results contained in Table II, we can deduce the following facts: (*i*) sensors data of different transportation modes exhibit unique patterns which can be recognized by automatic learners, with reasonably good accuracy. This is also in accordance with results presented in [28][30][33][36]; (*ii*) except for SVM (polykernel discriminants are used), costlier learning models improve the accuracy of the classification process; (*iii*) techniques that partition the training set on the basis of feature values' range (such as Decision Table and Random Forest) outperform techniques which rely on a geometric partitioning of the training set (such as the SVM). Random Forest (RF) is shown to produce the highest accuracy

Wirel. Commun. Mob. Comput. 2010; **00**:1–18
DOI: 10.1002/wcm
Prepared using wcmauth.cls

Algorithm	Accuracy (%)	Time (%) (Building)	Time (%) (Classification)
Random Forest (RF)	83.94	100	100
Decision Table (DT)	82.47	26.77	62.5
Bayesian Network (BN)	77.87	16.95	37.5
Random Tree (RT)	79.72	8.5	48.5
Support Vector Machines (SVM)	61.62	216.47	48.6
Nayve Bayes (NB)	54.45	13.98	11.1

Table II. Accuracy of different algorithms

(considering the average of all possible combinations of sensors data). Again, this is also in accordance with previous studies [33]. However, the accuracy varies significantly for different transportation modes, since some patterns appear more difficult to identify correctly, as discussed later in this Section.

In Table III we report the overall accuracy of the Random Forest (RF) algorithm, when we vary the sets of sensors data used for the classification. We use the notation introduced in Section 4.2. For each row, we also report the average power consumption required to perform a single sensor sampling, or multiple samplings in case of combinations of sensors (e.g. F^{Ac+Gy}). We distinguish between Measured values (i.e. power consumption measured through the PowerTutor [60] application), and Theoretical values (i.e. as indicated by the vendors' datasheet), although no significant differences can be observed between the two columns.

Again, lots of useful information can be drawn from results in Table III. First, we can easily notice that combining multiple sensors data always guarantees a performance increment, which reaches its maximum when all the available sensors are used for the classification purpose (i.e. $F^{Ac+Gy+Gps}$). Also, Table III demonstrates that the utilization of GPS alone is not enough to distinguish transportation modes with similar speeds (see also Figure 1), thus further motivating the utilization of accelerometer/gyroscope for human activity classification purposes. At the same time, it is easy to see that the energy consumption involved by the utilization of GPS is quite high, and might easily constitute a bottleneck on a realistic deployment, due to the limited resources of smartphones, and the need to acquire several samplings before producing a classification. For sake of fairness, we must also point out that the energy consumption values of embedded sensors (accelerometer/gyroscope) might depend on the specific hardware equipments of a smartphone, but in any case it results to be much lower than the GPS [49] [65] [66].* We can conclude that the F^{Ac+Gy} configuration provides the best trade-off between classification accuracy and power consumption, and for this reason we used it for the system implementation described in Section 6.

In Figure 4 we expand results of Table II and III, by

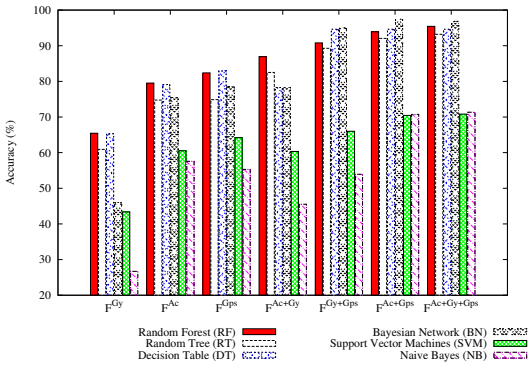


Figure 4. Average accuracy of learners on different combination of sensor data.

showing the performance of the six learners when using seven different combinations of sensors data. Again, it is possible to notice that -on the average- the Random Forest (RF) algorithm guarantees the highest accuracy, while the performance of some algorithms depends on the sensors data in use. For instance, the Bayesian Network (BN) algorithm provides the highest performance when combining multiple sensors data including the GPS (i.e. F^{Ac+Gps}, F^{Gy+Gps}, $F^{Ac+Gy+Gps}$). We can generalize the results previously shown in Table III: increasing the number of sensor data used for the classification translates into higher accuracy, independently from the learner.

We conclude the analysis by further elaborating on the ability of learners to identify specific transportation modes. Table IV provides the confusion matrix for the Random Forest (RF) algorithm, when using the F^{Ac+Gy} features, over the full set of transportation modes under analysis. Here, rows represent the real transportation modes, while the column report the classification produced by the learner. Each cell $< x, y >$ provides the percentage of samples belonging to mode x classified as mode y. For the cells forming the the diagonal (i.e. $< x, x >$), the percentage is also the accuracy of the classifier on mode x.

From Table IV, we can see that motorized modes (e.g. m^T, m^C, m^{Bc}) can be more easily recognized than non-motorized modes (e.g. m^W or m^{Bk}). This might be explained considering the fact that -on the average- the smartphone is less subject to orientation/acceleration vector changes when the user is on a car, on a bus or on a

*In Table III we report the average energy consumption for a GPS sampling. Our result is comparable with previous experiments in [49] .

Features	Accuracy (%)	Consumption(mW) Measured	Consumption(mW) Theoretical
F^{Ac}	79.53	6.85	5.99
F^{Gy}	65.45	49.28	42.12
F^{Gps}	82.87	480	440.11
F^{Ac+Gy}	86.93	56.12	48.11
F^{Ac+Gps}	93.91	526.58	446.1
F^{Gy+Gps}	90.81	487.85	482.23
$F^{Ac+Gy+Gps}$	95.44	533.43	488.22

Table III. Accuracy by using different sensor data (Random Forest)

Real/Predicted	Still	Walk	Car	Train	Bike	CityBus	NatBus
Still	**100**	0	0	0	0	0	0
Walk	0	**81.9**	0.9	14.5	1.8	0.6	0.3
Car	0	1.1	**95.5**	3.7	0.2	0.2	0.1
Train	0	1.9	7.5	**87.1**	0.5	1.5	1.4
Bike	0	2.6	1.3	5.4	**80.9**	6.5	3.2
CityBus	0	0	0.5	2.4	5.6	**89.8**	1.6
NatBus	0	0.1	2.8	10.1	7.2	10.2	**69.4**

Table IV. Confusion Matrix (Random Forest, F^{Ac+Gy})

Real/Predicted	Still	Walk	Car	Train	Bike	CityBus	NatBus
Still	**99.6**	0.2	0	0	0	0	0.2
Walk	0.1	**91.8**	0	0	0.1	2.7	5.3
Car	0	0	**65.6**	13.1	11.8	0	9.5
Train	0	0	13.2	**66.5**	12.1	0.1	8.1
Bike	5.1	0	2.1	2.2	**89.7**	5.1	0.9
CityBus	0	0.8	0	0	3.3	**95.6**	0.3
NatBus	0.4	4.7	3.4	1.9	4	6.9	**78.7**

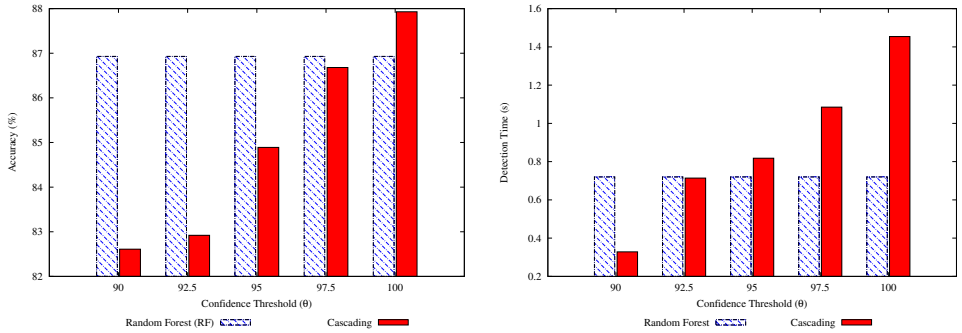
Table V. Confusion Matrix (Random Forest, F^{Gps})

train, respect when he/she is doing physical activities (like driving a bike or walking). An exception is constituted by the case of national bus (i.e. m^{Bn}), where misclassification errors occur more frequently due to similarity with the city bus. To confirm these considerations, we also report in Table V the confusion matrix when only GPS is used (i.e. F^{Gps}). Here we experience a dual situation than Table IV, since the highest accuracy is achieved for non-motorized modes (e.g. m^{W} or m^{Bk}), while a significant amount of misclassification errors occurs for motorized modes (e.g. m^{T}, m^{C}, m^{Bc}) which might exhibit overlapping intervals of speed values.

5.2. Analysis of multi-learners approaches

In this Section, we study the benefits of combining multiple learners through the cascading technique defined by Algorithm 1. To this purpose, we consider only a subset (K) of the six learners evaluated in Section 4, i.e. Random Tree (RT), Bayesian Network (BN), Decision Table (DT) and Random Forest (RF), since they provide much higher accuracy than the other two approaches (i.e. SVM and Naive Bayes). The order in which they are executed: l_1=Bayesian Network (BN), l_2=Random Tree (RT), l_3=Decision Table (DT), l_4=Random Forest (RF), is based on the computational overhead to classify a new instance, according to the results in Table II. We consider only the features extracted from to the accelerometer and gyroscope (i.e. F^{Ac+Gy}), since this configuration can provide the best trade-off between classification accuracy and energy-consumption, as previously shown in Table III. Beside cascading, several techniques have been proposed to combine learners in parallel or through multi-stages, and thus it is worth investigating which approach can be suitable for our domain. To this aim, Table VI reports the overall accuracy of five different multi-learner techniques, built over the set K of base learners, i.e.:

- *Cascading*: this refers to Algorithm 1, where the threshold θ is set to 97.5%.
- *Boosting(AdaBoost)* [50]: similar to the previous case, the learners are ordered on the basis of their complexity, and trained incrementally. However, no confidence metric is defined (Equation 4), and l_{i+1} is trained only on the wrong classifications of l_i.
- *Voting* [44]: in this case, all the learners are executed, and the final output is computed as the average of individual predictions.
- *Stacking* [51]: like the previous case, all the learners are executed, but the individual outputs are combined according to a non-linear function whose parameters are also learnt by the algorithm.



(a) Accuracy θ (b) Time over θ

Figure 5. The accuracy of Algorithm 1 as a function of the confidence threshold θ is shown in Figure 5(a). The average detection time of Algorithm 1 as a function of θ is shown in Figure 5(b).

- *Bagging* [45]: like the previous case, all the learners are executed, but they are trained on slightly different training sets, which are built through random replacements from a common set of instances.

From Table VI, we can deduce that the cascading approach provides the best performance in terms of classification accuracy, due to the fact that each learner decides only for the patterns for which it is confident. As a result, complex patterns (as those associated to the m^{Bk} mode, for instance) are classified by costlier learners, while simple patterns can be identified by quicker learners. Also, the cascading technique is highly efficient from the computational side, as indicated by the average values of model building and classification time. Results shown are normalized with respect to the cascading time (θ=100%). Not surprisingly, cascading provides the lowest average classification time, since on some instances only a subset of the K learners might be used, while the other techniques require to execute all the K base learners on each instance. The same benefits can be also appreciated in terms of model building time, since each learner is built on a subset of the training set.

In the following, we demonstrate another characteristic of the Algorithm 1 which makes it suitable for a mobile application deployment, i.e. the possibility to control the load produced by the classification process on the basis of the available computational resources. To this purpose, Figure 5(a) shows the overall accuracy of the cascading algorithm when varying the threshold θ, i.e. the minimum confidence required. We do not plot confidence thresholds below 90%, since there is a high probability that the first algorithm of the cascade (i.e. the Bayesian Network) would be confident on that specific samples, thus the accuracy would be equal to the Bayesian Network algorithm alone (i.e. 77.87%). Increasing θ translates into higher accuracy, since pattern classification is performed through costlier and more accurate algorithms (i.e. the Random Forest). We can think to θ in a double way: as (*i*) a user-defined parameter, which can be tuned to limit the energy consumption involved by the mobile application; or as (*ii*) a configuration parameter, which is automatically tuned by the system on the basis of the hardware characteristics of the device. Moreover, in Figure 5(a) we also plot the accuracy of the Random Forest algorithm over the same sensors data (F^{Ac+Gy}). No confidence threshold is used by the Random Forest (RF), and thus all the bars refer to the same value. It is interesting to notice that, when requiring the maximum confidence (i.e. $\theta = 100\%$), the cascading algorithm provides higher accuracy than the best base learner (i.e. the Random Forest) , thus demonstrating the benefits of combining multiple learners with a multi-stage training technique. Also, Figure 5(a) shows that our approach can provide accuracy values comparable to the configuration used in [36] including also the GPS (i.e. F^{Ac+Gps}), but with a significant reduction of the energy consumption. In Figures 5(b) and 6 we provide further insights on the scalability of Algorithm 1. More specifically, in Figure 5(b) we report the average detection time required by the cascading algorithm, as a function of the threshold θ. While model building time does not depend on θ, detection time increases with θ, since costlier learners are more heavily involved in the classification. This is clearly shown by Figure 6, where we show the percentage of classifications performed by each learner, as a function of the threshold θ. On the same instance, multiple learners can be executed at different stages, based on the requested confidence. This explain why the classification time of cascading exceeds the performance of the Random Forest (RF) for $\theta \geq 95\%$.

For sake of completeness, we report in Table VII the confusion matrix of the cascading algorithm for a threshold θ equal to 0.975. By comparing results below with those contained in Table IV (i.e. the confusion matrix of the Random Forest algorithm), we can notice that differences among transportation modes are reduced, i.e. there is no clear distinction among motorized/non-motorized modes,

Algorithm	Accuracy (%)	Time (%) (Building)	Time (%) (Classification)
Cascading	87.94	100	100
Bagging	87.93	648	289
Boosting	87.45	587	312
Voting	85.88	173	106
Stacking	84.39	1146	103

Table VI. Accuracy of different multi-learner algorithms

Real/Predicted	Still	Walk	Car	Train	Bike	CityBus	NatBus
Still	**89.4**	0	5	5.6	0	0	0
Walk	0	**91.5**	1.1	4.1	1.9	1	0.3
Car	0.5	0.	**89.1**	9.7	0.1	0.2	0.2
Train	0.2	2.7	7.8	**87.5**	1.1	0.1	0.3
Bike	0	3.2	1.8	6.2	**80.6**	3.6	4.4
CityBus	0	0.1	1.4	3.3	5.6	**85**	3.6
NatBus	0	0.6	2.8	4.2	3.8	4.3	**84**

Table VII. Confusion Matrix (Algorithm 1, F^{Ac+Gy})

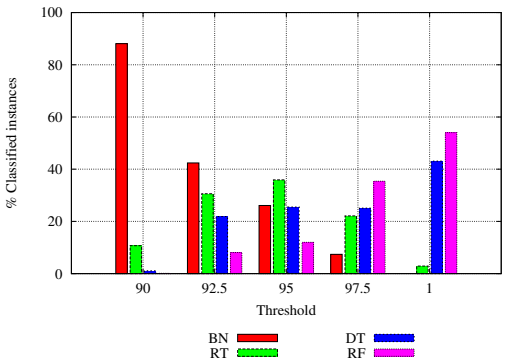


Figure 6. Percentage of instances classified by each base learner, as a function of the confidence threshold θ.

and that the accuracy is higher than 80% for all the evaluated modes.

5.3. Analysis of Methodologies and Parameters for Training

In supervised learning, how to build and populate the training set constitutes a fundamental issue affecting the system performance. In the evaluation conducted so far, a mobile data crowd-sourcing [15] approach is assumed, i.e. the training set is produced by aggregating sensors data from different sources (people/devices). This is mainly motivated by issues of generality, since we were interested in investigating the performance of the transportation mode recognition systems without assuming any specific target device and any specific habit from the end-users. However, implementing a data crowd-sourcing technique poses several problems [15], both in terms of data acquisition (i.e. how to protect the users' anonymity and privacy?) and management (a centralized infrastructure is needed). As an alternative, training can be performed locally, by using the sensors data collected by each end-user on its mobile device.

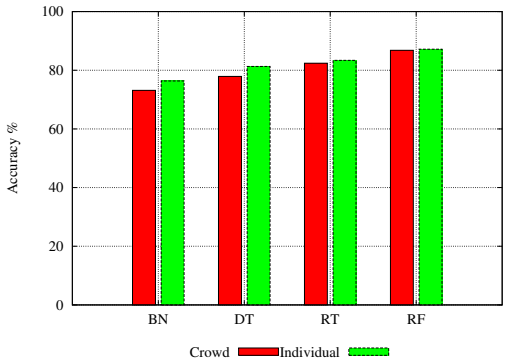


Figure 7. Data acquisition approaches: Individual vs Mobile-crowdsourcing.

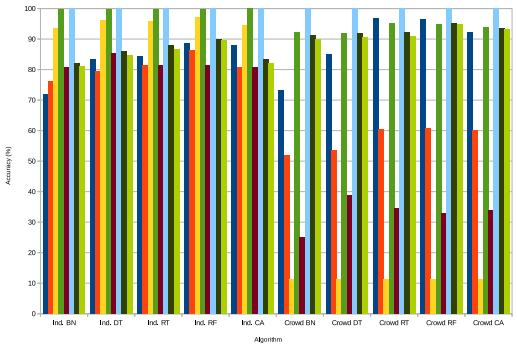


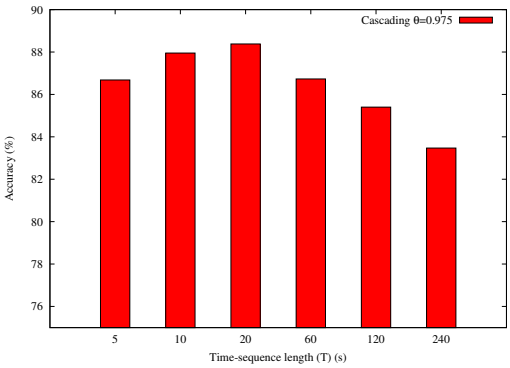
Figure 8. Data acquisition approaches: Individual vs Mobile-crowdsourcing per single user.

In Figure 7, we compare these two alternatives of training set creation, when we vary the learner used for

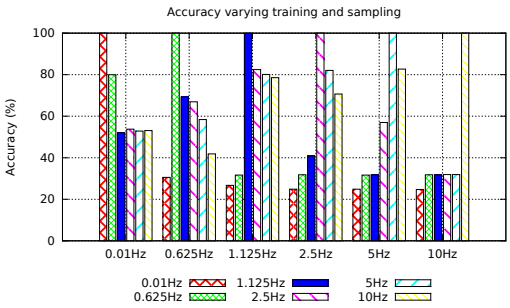
Wirel. Commun. Mob. Comput. 2010; **00**:1–18 © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm
Prepared using wcmauth.cls

the classification. Figure 8 presents instead the same data, but sketched for every user rather than averaged. For the individual training, we consider the accuracy experienced by each of the eight participants to our experiments, and we plot the average. Surprisingly, the individual training provides performance equal or slight better than a crowd-sourcing approach, mainly due to the ability to track the user habits, such as the way to carry the smartphone while he is driving or is on the train. We highlight the relevance of this result, since it justifies the possibility to implement a transportation mode recognition system as a stand-alone mobile application, solving the data anonymity challenge mentioned before, and greatly simplifying the system deployment. However, we also note that it would be possible to distribute the application with anonymous data coming from different sources, thus requiring no specific training in the beginning. Individual training would then by an option that each user would decide whether to perform or not, based on his/her satisfaction with the classification of the transportation modes.

Other two important parameters influencing the training phase are the time-sequence length (T) and the sampling rate r, introduced in Section 4. In Figure 9(a) we show the overall accuracy as a function of the time-sequence length T (on the x-axis) used to collect the data. On the x axis we plot the training frequency, on the y axis the accuracy, and the different bars represent different sampling frequencies. From this result, it is easy to notice that this parameter can affect overall accuracy since: (*i*) too short sampling windows do not allow to capture the distinctive features of a transportation mode and (*ii*) too long sampling windows can produce classification errors among transportation modes with similar features. In Figure 9(b) we show the impact of the sensing sampling rate r on the overall detection accuracy. More specifically in Figure 9(b) we plot the accuracy when classifying samples at a given rate r, using a varying training data-set, performed at different sampling frequencies. The results in Figure 9(b) show that: (*i*) optimal accuracy can be achieved when training and predicting rates are the same, (*ii*) on average, the detection accuracy tends to increase in *under-sampling* conditions (i.e. prediction rate lower than the training rate), and to decrease in *over-sampling* conditions (i.e. prediction rate higher than the training rate) and (*iii*) high frequency sampling does not provide significant performance improvements, i.e. the accuracy of the classifier is almost independent by the training rate, at the conditions of using the same prediction rate.



(a) Impact of T values on accuracy



(b) Impact of r values on accuracy

Figure 9. The accuracy of Algorithm 1 computed over different sequence lengths (T) and for different sampling rates (r) is shown in Figures 9(a) and 9(b), respectively.

6. IMPLEMENTATION

After having investigated the performance of a transportation mode recognition system based on a multi-learner approach, we implemented it into a mobile application for the Android [†] platform. Figure 10 shows the architecture of our framework, called WAID (What Am I Doing).

The WAID framework can run in foreground or in background as an Android Service, and supports two different modes: *Training* and *Predicting*. In the *Training* mode, the application asks the user to indicate the current transportation mode, and starts collecting samples from the accelerometer/gyroscope at a fixed rate r (see the screenshot in Figure 12(a)). Every T seconds, the Feature Extractor module computes the features defined in Section 4.2 from each sensors data (i.e. $min(s,k)$, $max(s,k)$, $avg(s,k)$, $std(s,k)$), and stores them on a Training Set repository, implemented as an SQLite database. Both the sampling rate r and the observation window length T (set to 10 Hz and 10 seconds by default) can be tuned by the user through the graphical interface. Once the user stops the data acquisition, the Model Builder executes WEKA [48] to generate the models of the four base learners (i.e. RF, RT, DT, BN) from the Training Set, based on the training algorithm defined in Section 4.3. The model of Cascading is then generated according to Algorithm 1. In the *Predicting* Mode, the

[†] Target version is Android 2.3.3 or later

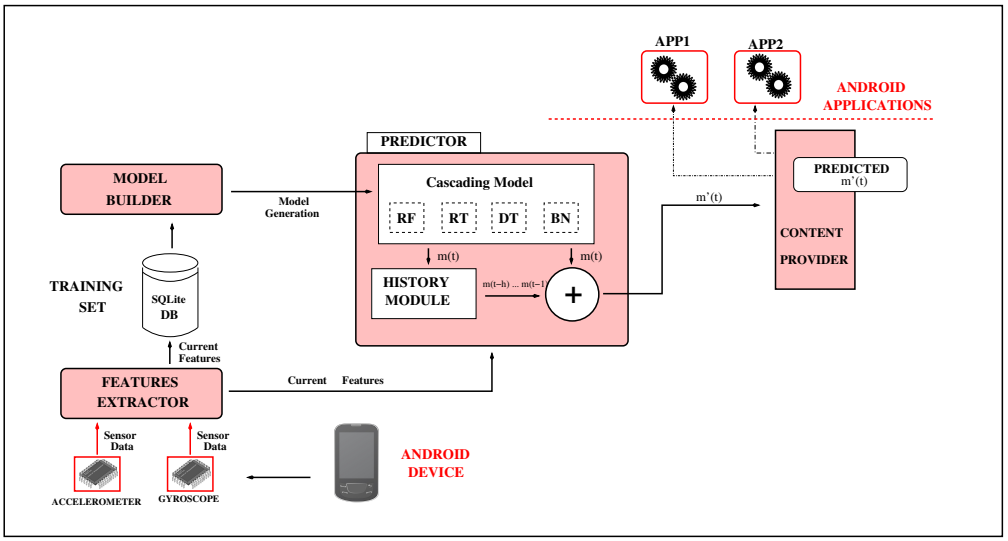


Figure 10. The WAID framework architecture.

application recalls the Feature Extractor modules (as in Training mode) to extract the features from sensors data relative to the current time sequence t, and then executes the Predictor module which produces in output the estimated transportation mode at time t, i.e. $m'(t)$. The Predictor Module is based on the Cascading algorithm, but also includes an additional mechanism (i.e. the History module, explained in Section 6.1, with h set to 5 in our implementation) to reduce the occurrence of misclassification errors on time-series. Figure 12(b) shows a screenshot of the WAID user interface in Predicting mode, where the $m'(t)$ values are plotted to the screen. Moreover, these values are also exported into a Content Provider, which represents an Android facility to share data among different processes/applications. As a result of this choice, other Android applications can be built on top of the WAID framework and can leverage the information about the current transportation mode for enhanced context-aware experiences. Since the data sharing process is implemented through a standard interface of Content Providers, the external applications do not need to know the implementation details of WAID in order to access and consume the $m'(t)$ data. In Section 6.2 we provide details of a sample application (i.e. the Device Adapter) that has been built on top of the WAID framework.

6.1. Predictor Module

In Predicting mode, the classification algorithm used by WAID relies on the output of Algorithm 1, with a threshold set to $\theta = 100$. At each time sequence t, features are extracted from sensors data (i.e. F^{Ac+Gy}), and the algorithm is executed to produce a new classification m_t. However, treating each classification response in isolation without considering possible time correlations might lead to bizarre sequences like: m_t^C,m_{t+1}^C, m_{t+2}^T, m_{t+3}^C, m_{t+4}^C, which is clearly wrong, since the user cannot intermittently change its transportation mode in a short observation window. We adopted a similar approach to [35], i.e. we considered the time-correlation among consecutive classifications through the History Module of Figure 10. This component keeps track of the latest h classifications performed by Algorithm 1 and stores them into an history-set H, i.e.:

$$H = \{m_{t-h-1}, m_{t-h+1}, ...m_{t-1}\} \quad (5)$$

Once a new classification $m(t)$ is performed by the Cascading algorithm, it is inserted into the history-set H and the m_{t-h-1} value is removed, so that the sliding window is adjusted properly. At the same time, a reference answer $m'(t)$ is computed as the transportation mode having more occurrences in H. Then, $m'(t)$ is returned to the user and placed into the Content Provider. We highlight here that $m'(t)$ might be different than $m(t)$, i.e. the History Module might correct the current classification on the basis on the past values. As a result, fluctuating sequences like the one previously described are more unlikely to occur.

An important parameter of the History Module is h, i.e. the history size. Indeed, short values of h might still induce classification errors due to random fluctuations, while long values of h might impact the reactivity of the application in tracking effective transportation mode changes. This trade-off is captured by Figure 11, where we show the accuracy of Algorithm 1 for different values of h. We consider a realistic scenario in which the user dynamically changes its transportation mode during the experiment (i.e. from m^W to m^C) and we depict the average detection accuracy of Algorithm 1 over the whole length of the experiment,

Wirel. Commun. Mob. Comput. 2010; **00**:1–18 © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm
Prepared using wcmauth.cls

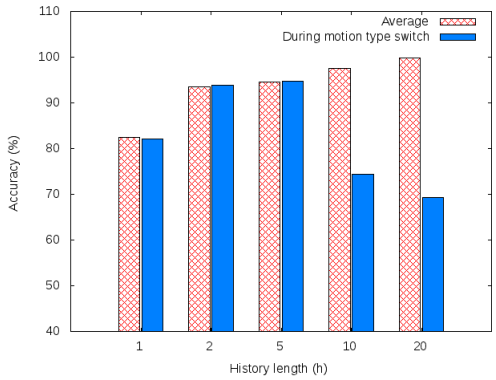


Figure 11. Impact of history length (h).

and the average accuracy during the mode switch [‡]. As expected, Figure 11 demonstrates that the configuration with the longest value of h minimizes the occurrences of classification errors (on average). At the same time, it is easy to see that shorter values of h provide higher accuracy in proximity of the transportation mode switch.

We highlight here that there exist a trade-off to detect transitions. If one wants to be more responsive to transitions from one action to another, the he/she would set a low h values, to give more importance to the new values over the old ones. However, this might induce classification errors, as also shown in Figure 11. Inversely, if the user wants to be more confident on the classification, at the expense of a slower reaction to transitions, then he/she would set a higher h value. This would need a higher number of classified action after the transition to actually detect it, but clearly generates less classification errors, as Figure 11 shows. We note that no magical value do exist, since the h value of the history might be different according to the user wanted application behavior.

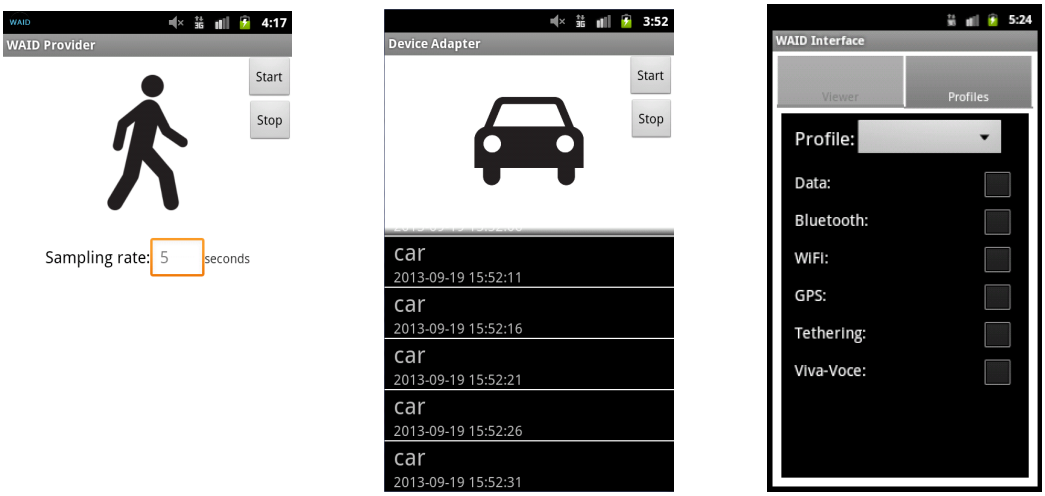
6.2. Transportation Mode-aware Applications

Transportation mode information provided by the WAID framework can enhance the context-awareness of mobile applications in lots of possible domains, as discussed in Section 2.1. Since the information is exported through a Content Provider, new applications can be deployed on top of WAID framework. To provide a proof-of-concept, we deployed another Android application, called Device Adapter, through which the user can customize the configuration of its smartphone based on the detected transportation mode. The application comprises a classification model based on anonymous data from the crowd, which can be further specialized by each user. More specifically, this application allows the user to define a profile associated to each transportation mode. A profile is constituted by a list of preferences about the ring-tone state (e.g. volumes of ring-tones, vibration on/off) and/or about the network interfaces' state (e.g. Wifi on/off). An example of profile associated to m^T (train mode) might be: 3G data network: off, vibration: on, ring-tones: off. Figure 12(c) shows the screenshot of the application where the user is associating a profile to a mode. Once such mode is detected by WAID, the Device Adapter is notified through the Content Provider, and the device configuration is adjusted according to the profile.

[‡] In practice, we computed the average accuracy over the next 5 predictions following the mode switch.

7. CONCLUSIONS

In this paper, we have addressed the problem of how to automatically recognize the current transportation mode of an user from its smartphone sensors data. We have discussed possible use-cases of a recognition system, and the challenges for its practical deployment on a mobile device. Through participatory measurements we have built a training set which has been used to evaluate different classification algorithms and features' sets. From the evaluation results, we have found that a combined multi-stage learner using the embedded sensors (accelerometer/gyroscope) data can provide reasonably high accuracy for different class of motorized and non-motorized transportation modes, and can prolong the energy lifetime of the device when compared to a GPS-based classifier. Finally, we have proposed the implementation of our recognition system into an Android framework, and we have detailed the software architecture and the possible integrations with other third-part Android applications. Although our system guarantees reasonably accuracy in detecting several classes of transportation modes (seven classes tested in our analysis), there are several research issues that might be further explored, and that we are currently investigating. One of this issue is intrinsically related to the utilization of supervised machine-learning techniques, i.e. the need of an extensive and accurate training phase. In our case, training is fundamental, since the accuracy of the classification depends on the amount of data available for each transportation mode, and by the noise affecting the registrations (for instance, an user can stop several times while walking, or might change the orientation/position of the devices during the experiments). Since training requires an explicit feedback from the end-user, it might constitute a practical limitation to the usage of the WAID framework. For these reasons, we are considering the possibility to equip the WAID application with a pre-defined training set, which might eventually be extended by each end-user through the *Training* mode of the WAID application. We are currently evaluating the accuracy of this solution on a real scenario. Another potential issue (pointed out also in Section 5.1) is that the proposed system might fail in distinguishing transportation modes where human activity is prevalent (i.e. walking or biking), while the combined utilization of GPS traces might help in



(a) Training mode (b) Predicting mode (c) Device Adapter

Figure 12. The screenshots of the WAID framework in Training Mode and Predicting Mode are shown in Figure 12(a) and 12(b), respectively. The Device Adapter interface is shown in Figure 12(c).

correctly classifying these cases. For this reason, we are currently studying duty-cycle techniques to dynamically turn on/off the GPS receiver, which -combined with rate-adaptive sampling algorithms for the embedded sensors- might guarantee high classification accuracy over all the transportation modes considered so far, while producing a limited impact on the energy consumption of the device.

ACKNOWLEDGMENTS

The authors would like to thank Andrea Jonus for his contribution to the mobile application deployment.

REFERENCES

1. Graham D., Simmons G., Nguyen D. T., Zhou G. A Software-Based Sonar Ranging Sensor for Smart Phones *IEEE Internet of Things Journal* vol. 2, no. 6, pp. 479-489, Dec. 2015.
2. Han T., Ansari N. Offloading Mobile Traffic via Green Content Broker *IEEE Internet of Things Journal* vol. 1, no. 2, pp. 161-170, April 2014.
3. Liu, J., Ansari, N. Identifying website communities in mobile internet based on affinity measurement *Computer Communications*
4. Liu, J., Fang, C., Ansari, N. Request Dependency Graph: A Model for Web Usage Mining in Large-scale Web of Things *Internet of Things Journal*
5. Jie Yang, Lun Yuan, Chao Dong, Gang Cheng, Ansari, N., Kato, N. On Characterizing Peer-to-Peer Streaming Traffic *Selected Areas in Communications, IEEE Journal on , vol.31, no.9, pp.175-188, September 2013*
6. Jun Liu, Feng Liu, Ansari, N. Monitoring and analyzing big traffic data of a large-scale cellular network with Hadoop *Network, IEEE , vol.28, no.4, pp.32-39, July-August 2014*
7. MIT Technology Review. http://www.technologyreview.com/photoessay/511791/smartphones-are-eating-the-world/.
8. A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo and R. A. Peterson. People-centric urban sensing. *Proc. of ACM WICON*, Boston, USA, 2006.
9. A. Ghosh and S. Das Coverage and connectivity issues in wireless sensor networks: A survey . *Pervasive and Mobile Computer (Elsevier)*, 4(3), pp. 303-334, 2008.
10. P. Mohan, V. N. Padmanabhan, R. Ramjee Nericell: rich monitoring of road and traffic conditions using mobile smartphones. *Proc. of ACM SENSYS*, Raleigh, USA, 2008.
11. R. Bhoraskar, N. Vankadhara, B. Raman, and P. Kulkarni. Wolverine: Traffic and road condition estimation using smartphone sensors. *Proc. of IEEE COMSNETS*, Bangalore, India, 2012.
12. N. Brouwers and M. Woehrle Dwelling in the canyons: dwelling detection in urban environments using GPS, Wi-Fi and geolocation. *Pervasive and Mobile Computer (Elsevier)*, article in press, 2012.
13. L. Ferrari and M. Mamei Identifying and understanding urban spot areas using Nokia Sports Tracker. *Pervasive and Mobile Computer (Elsevier)*, article in press, 2012.
14. N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury and A. T. Campbell. A survey of mobile phone sensing. *IEEE Communication Magazine*, (48)9, pp. 140-150, 2010.
15. R. K. Ganti, F. Ye and H. Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communication Magazine*, (49)11, pp. 32-39, 2011.
16. X. Sheng, X. Xiao, J. Tang and G. Xue. Sensing as a service: A cloud computing system for mobile phone sensing. *IEEE Sensors*, (49)11, Taipei, Taiwan, 2012.
17. C. Randell, and H. Muller. Context awareness by analysing accelerometer data. *Proc. of IEEE ISWC*, Atlanta, 2000.
18. L. Bao and S. Intille. Activity Recognition from User-Annotated Acceleration Data. *Pervasive Computing*: 3001(1), 1-17, 2004.
19. Z.-Y. He, and L.-W. Jin. Activity recognition from acceleration data using AR model representation and SVM. *Proc. of ICMLA*, pp. 2245-2250, San Diego, USA, 2008.

20. Z.-Y. He, and L.-W. Jin. Activity recognition from acceleration data based on discrete consine transform and SVM. *Proc. of IEEE SMC*, pp. 5041-5044, San Antonio, USA, 2009.
21. N. Ravi, N., Dandekar, P. Mysore, and M. L. P Littman. Activity Recognition from Accelerometer Data. *Neural Networks*: 20(3), pp. 1541-1546, 2005.
22. P. Casale, O. Pujol and P. Radeva. Human Activity Recognition from Accelerometer Data Using a Wearable Device, *Proc. of IbPRIA*, pp. 289-296, Gran Canaria, Spain, 2011.
23. A. Bujari, B. Licar, and C.E. Palazzi. Movement Pattern Recognition through Smartphone's Accelerometer. *Proc of IEEE CCNC*, Las Vegas, USA, 2012.
24. A. Henpraserttae, S. Thiemjarus and S. Marukatat. Accurate activity recognition using a mobile phone regardless of device orientation and location. *Proc. of IEEE BSN*, Dallas, USA, 2011.
25. A. Akl, C. Feng and S. Valaee. A novel accelerometer-based gesture recognition system. *IEEE Transactions on Signal Processing*, 59(12), pp. 6179-6205, 2011.
26. S. L. Lau and K. David. Movement recognition using the accelerometer in smartphones. *Proc. of IEEE FNMS*, Florence, Italy, 2010.
27. S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas and D. J. Cook Simple and convex activity recognition through smart phones. *Proc. of IEEE IE*, Guanajuato, Mexico, 2012.
28. M. A. Ayu, T. Mantoro, A. F. A. Matin and S. S. O. Basamh Recognizing user activity based on accelerometer data from a mobile phone. *Proc. of IEEE ISCI*, Kuala Lumpur, Malaysia, 2011.
29. L. Bedogni, M. Di Felice, L. Bononi. By train or by car? Detecting the user's motion type through Smartphone sensors data. *Proc. of IEEE/IFIP Wireless Days*, Dublin, Ireland, 2012.
30. C. Xu, M. Ji, W. Chen and Z. Zhang. Identifying travel mode from GPS trajectories through fuzzy pattern recognition. *Proc. of IEEE FSKD*, Yantai, Chinas, 2010.
31. M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, N, Hansen, E. Howard, R. West and P. Boda. PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. *Proc. of ACM Mobysys*, Krakow, Poland, 2009.
32. Y. Zheng, Y. Chen, Q. Li, X. Xie and W.-Y. Ma Understanding transportation modes based on GPS data for Web applications. *ACM Transactions on The Web*, 4(1), pp. 1-36, 2010.
33. L. Stenneth, O. Wolfson, P. S. Yu and B. Xu. Transportation mode detection using mobile phones and GIS information. *Proc. of ACM GIS*, Chicago, USA, 2011.
34. Y. Xiao, D. Low, T. Bandara, P. Pathak, H. B. Lim, D. Goyal, J. Santos, C. Cottrill, F. Pereira, C. Zegras and M. Ben-Akiva. Transportation activity analysis using smartphones. *Proc. of IEEE CCNC*, Las Vegas, USA, 2012.
35. P. Widhalm, P. Nitsche and N. Brandle. Transport mode detection with realistic smartphone sensor data. *Proc. of IEEE ICPR*, Tsukuba, Japan, 2012.
36. 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*, 6(2), pp. 1-27, 2010.
37. S. Wang, C. Chen and J. Ma. Accelerometer based transportation mode recognition on mobile phones. *Proc. of IEEE APWCS*, 4(1), Shenzhen, China, 2010
38. P. Troped, M. S. Oliveira, C. E. Matthews, E. K. Cromley, S. J. Melly and B. A. Craig. Prediction of activity mode with global positioning system and accelerometer data. *Medicine & Science in Sports & Exercise*, pp. 972-978, 2008.
39. D. A. Rodriguez, A. L. Brown and P. J. Troped. Portable global positioning units to complement accelerometery-based physical monitors. *Medicine & Science in Sports & Exercise*, (35)11, pp. 572-581, 2005.
40. E. Alpaydin. Introduction to Machine Learning. *MIT Press*, 2009.
41. C. Kaynak and E. Alpaydin. Multistage cascading of multiple classifiers: one man's noise is another man's data. *Proc. of ICML*, Stanford, USA, 2000.
42. If this do that (IFTTT) Platform: https://ifttt.com/
43. L. Breiman. Random forest. *Machine Learning*, 45(1), pp. 5-32, 2001.
44. Dietterich, Thomas G. Multiple Classifier Systems
45. Breiman, Leo Machine Learning Bagging predictors
46. C. Cortes, V. Vapnik, Support-vector networks. *Machine Learning*, 20(3), pp. 273-297, 1995.
47. G. H. John, P. Langsley. Estimating continuous distributions in Bayesian classifiers. *Proc. of UAI*, pp. 338-345, Quebec, 1995.
48. M. Hall, E. Frank, G. Holmes, B. Pfahringer, R. Reutemann, I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations*, 11(1), pp. 10-18, 2009.
49. J. Paek, J. Kim, R. Govindan. Energy-efficient rate-adaptive GPS-based positioning for smartphones. *Proc. of ACM Mobisys*, San Francisco, USA, 2010.
50. Y. Freund and R. E. Schapire . Experiments with a new boosting algorithms. *Proc. of ICML*, San Mateo, USA, 1996.
51. D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(1), pp. 241-259, 1992.
52. Prashanth Mohan, Venkata N. Padmanabhan, Ramachandran Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones. *Proc. of the 6th ACM Conference on Embedded Network Sensor Systems*.
53. Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, Andrew T. Campbell. Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application. *Proc. of the 6th ACM Conference on Embedded Network Sensor Systems*.
54. Choudhury T., Consolvo S., Harrison B., Hightower J., LaMarca A., Legrand L., Rahimi A., Rea A., Bordello G., Hemingway B., Klasnja P., Koscher K., Landay J.A., Lester J., Wyatt D., Haehnel D.. The Mobile Sensing Platform: An Embedded Activity Recognition System. *Pervasive Computing, IEEE , vol.7, no.2, pp.32,41, April-June 2008*
55. Zhixian Yan, Dipanjan Chakraborty, Sumit Mittal, Archan Misra, Karl Aberer. An Exploration with Online Complex Activity Recognition Using Cellphone Accelerometer. *Proc. of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*
56. Yi He, Ye Li. Physical Activity Recognition Utilizing the Built-In Kinematic Sensors of a Smartphone. International Journal of Distributed Sensor Networks, vol. 2013, Article ID 481580, 10 pages, 2013
57. Sanches Joao, Micò Luisa, Cardoso Jaime. Physical Activity Recognition from Smartphone Embedded Sensors. *Book Chapter in Pattern Recognition and Image Analysis*
58. Samuli Hemminki, Petteri Nurmi, Sasu Tarkoma. Accelerometer-based transportation mode detection on smartphones. *Proc. of ACM SenSys 2013*
59. Google Inc.. Google Location API - https://developer.android.com/google/play-services/location.html
60. Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, Lei Yang. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones. *Proc. of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*

61. Swagath Venkataramani, Anand Raghunathan, Jie Liu, Mohammed Shoaib Scalable-effort classifiers for energy-efficient machine learning *Proceedings of the 52nd Annual Design Automation Conference (DAC '15).*
62. P. Viola, M. Jones Fast and robust classification using asymmetric adaboost and a detector cascade Advances in Neural Information Processing System, 2001
63. H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury and A. T. Campbell The Jigsaw continuous sensing engine for mobile phone applications *Proc. of ACM SenSys, Zurich, Switzerland, 2010*
64. N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell and F. Zhao Enabling large-scale human activity inference on smartphones using community similarity networks (csn) *Proc. of ACM Ubicomp, Bejing, China, 2011*
65. A. Carroll, G Heiser An Analysis of Power Consumption in a Smartphone in *USENIX annual technical conference, 2010*
66. Oshin, T.O., Poslad, S., Ma, A. Improving the Energy-Efficiency of GPS Based Location Sensing Smartphone Applications in *Trust, Security and Privacy in Computing and Communications (IEEE TrustCom), 2012*

DOI: 10.1002/wcm
Prepared using wcmauth.cls