IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 18, 2020, accepted March 1, 2020, date of publication March 23, 2020, date of current version April 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2982218

Smartphone Data Classification Technique for Detecting the Usage of Public or Private Transportation Modes

PINO CASTROGIOVANNI[1], EDOARDO FADDA[2,3], GUIDO PERBOLI[2,3], AND ALESSANDRO RIZZO[4], (Senior Member, IEEE)

[1]TIM Joint Open Lab, Politecnico di Torino, 10129 Torino, Italy
[2]Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy
[3]ICT for City Logistics and Enterprises Lab (ICE Lab)
[4]Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, 10129 Torino, Italy

Corresponding author: Edoardo Fadda (edoardo.fadda@polito.it)

ABSTRACT One of the main endeavors of smart cities is the organization and subsidization of public transportation. To achieve this, it is important to obtain information about the way in which people move. This once-difficult problem can now be addressed by using smartphones. This paper introduces a machine learning-based framework that is able to ascertain the usage of a public or a private transportation mode by analyzing a little amount of data sampled by a user's smartphone. The presented method exhibits a good accuracy and a limited battery consumption. A public anonymized dataset based on real measurements is also provided along with this study. To the best of our knowledge, this is the first dataset of this kind that is offered to the public.

INDEX TERMS Mobility, transportation mode detection, smart city.

I. INTRODUCTION

The use of smartphones is nowadays pervasive. In addition to communication capabilities, they are also equipped with several sensors, and are usually carried by people throughout the day. Several applications have been released to provide fundamental tools to improve life in smart cities. Innovative applications revolve around e-work ([1], [2]), opportunistic data collection from Internet of Things (IoT) sensors ([3]–[5]), and the collection of data useful for transportation ([2], [6], [7]). One of the most important issues in the latter topic is the use of smartphone data for detecting the transportation mode chosen by users. This problem is called Transportation Mode Detection (TMD) and it is often addressed through machine learning approaches [8].

In this paper, we focus on the online detection of the users' mode of transportation, namely determining whether he/she is traveling by public or private transportation means. This problem is compelling, as the knowledge of transportation habits of people can inform municipalities in planning and optimizing transportation services, devising mobility models, and improving navigation systems ([9]–[11]).

The TMD problem can be addressed by using different sets of data, such as those coming from smartphone sensors (e.g., Global Positioning System (GPS), accelerometer, gyroscope, compass, etc.) or data taken from social and environmental sources such as tweets, facebook posts, and smart cameras ([8], [12]).

The techniques using the former set of data, usually, attain good performance and have fewer privacy problems. The level of accuracy is so high, for certain transportation modes, that Android has a suite of functions installed in each smartphone (called Android Activity recognition Application Programming Interfaces) that enables detection of a user motion mode, specifically, whether he/she is moving by vehicle, by bicycle, or by foot.

In the academic field, a large branch of the literature dealing with innovative methods for detecting transportation modes with high accuracy exists [8]. Nevertheless, to the best of our knowledge all of the presented approaches have three main drawbacks, which the approach presented in this paper overcomes.

The associate editor coordinating the review of this manuscript and approving it for publication was Omid Kavehei.

The first drawback is that they use data coming from sensors that are not present in all smartphones. Hence, due to the heterogeneity of mobile phone components, we restrict the presented application to consider sensors that are present in all mobile phones, such as GPS and accelerometers.

The second drawback is that the used sensors drain energy from the smartphone in the collection of data and computation of results. Smartphones are in fact an essential component of our daily life and no one wants to install applications that drain too much energy from their device. The presented methodology achieves accurate classification results while limiting the energy power consumption in the sampling phase. It is worth noting that, despite the importance of this question for the feasibility of the method, no study has explicitly taken this problem into account.

The third drawback is the lack of benchmark instances, which do not grant repeatability of the experiments. To the best of our knowledge, this is the first work providing an open anonymized dataset of the collected measurements. It is important to note that the first two drawbacks are critical for the development of a real-life application for the detection of transportation mode.

In order to provide useful information to the municipality, TIM, the main Italian telecommunication company, the ICT for City Logistics and Enterprises Lab of Politecnico di Torino, and the startups Move Plus and Pony Zero have joined the project Open Agorá.[1] The main objective of the project is to improve sustainable mobility and its final deliverable is a mobile application that people can download and install. Once launched, it runs in the background, collecting data that are daily sent to a server. Once the server collects enough information, it proposes more sustainable travel options to the app user and possible actions to the municipality for improving its public transportation services. It is worth noting that by sustainable travel options we mean a set of alternative transportation modes more environmental friendly (such as public transportation, bicycle, etc.) that enable the user to travel from its origin to its destination without sensible time variations. Sustainability is a key target for the smart cities ([4], [13]–[15])

All the results presented in the paper have been developed in the Open Agorá project. The proposed methodology is composed by three steps. In the first step, we extract a set of characteristic features from the datasets of each travel. In the second step, we use machine learning to infer whether each travel is done by using a private transportation mode or a public one. Finally, we search a trade-off between classifier performance and data sampling rate of the sensors in order to minimize energy consumption.

The article is organized as follows: in Section II, we conduct a literature review of transportation mode detection. In Section III, we briefly describe the characteristics of smartphones, the procedures used for collecting data, while the features and classifiers used for the analysis are described in Section IV. In Section V, we discuss the performance of the proposed methodology by using real data. Finally, in Section VI, we present the conclusions of the work.

[1] http://torinolivinglab.it/portfolio/open_agora/

II. LITERATURE REVIEW

The literature on the TMD problem is wide, covering different sets of transportation modes and techniques used. The goal of presenting an exhaustive literature review about the TMD problem is out of the scope of the present paper and thus we limit our review to the most relevant papers related to our specific application. As an example of the heterogeneity of the papers in this field, [16] considered features to ascertain if a user is walking or traveling by bicycle, car, bus, tram, or subway. It extracts features from the data of the accelerometer and gyroscope. On the other hand, for the same set of transportation modes, [17] considered only acceleration data. Several studies have compared different classification algorithms, in order to find the best one. We summarize the most recent results in Table 1. To the best of our knowledge, in the present literature, the papers that present applications closest to ours are [17]–[22], and [23]. Nevertheless, none of them focuses on the classification of public and private transportation modes. For this reason, we consider the closest comparison; that is, the discrimination between the use of car and bus. In particular, [18] uses standard statistics indicators (average, standard deviation, mode, and median) of the acceleration series to discriminate between several different sets of transportation modes. The classifier chosen for their study is the binomial logistic regression. Despite the simplicity of their approach, they achieve a high goodness of fit in several comparisons, nevertheless, the discrimination between car and bus reaches an accuracy of 66.7%.

The papers [19], [20], and [21] use GPS, accelerometer data, and Geographical Information System (GIS) data, in order to discriminate between car and bus. The papers [19] and [21] obtain accuracies of 99% and 97%, respectively, by using Random Forest (RF) methods. Alternatively, [20] reaches an accuracy of 82% by using RF and a Hidden Markov Model (HMM) classification algorithm. In [22], the authors use the GPS and accelerometer to gather data, combined with GIS data and other signals, such as GSM or Wi-Fi. For the classification, they use Bayesian networks as classifiers. Their method achieves an accuracy of 75% for the discrimination between car and bus. For the same discrimination, [17], [24] and [23] use only acceleration data and reach accuracies of 85% and 82% by using a HMM and adaptive boost decision tree (DT), respectively. A paper that obtain an accuracy greater than 90% in the discrimination between car and bus is [25]; this paper considers accelerometer data, in combination with GPS data. The techniques adopted by the authors is a Bayesian Belief Network that uses data collected from ad-hoc devices, more accurate than usual smartphones.

With the same choice of initial data, [26] develops a large-scale travel survey by employing data from smartphones. A total of 266 hours of travel data are collected to design and evaluate the models. Using a set of 72 features,

the best classification results are achieved for detecting walking (92%) and bike riding (98%). The accuracy of the detection of car transportation is 75%.

Another relevant work is [27]. In this paper, the authors sample data from the accelerometer, gyroscope, and rotation vector using the highest possible frequency of the application, but they do not sample GPS data, as the usage of the GPS system can deplete the battery. To the best of our knowledge, this is the first and only paper that addresses the problem of energy consumption. However, differently from the present paper, it does not consider the optimization trade-off between the reduction of sensor sampling and machine learning performance.

For the sake of completeness, we cite [28], [29], and [30] as the most important studies that consider only GPS data to recognize different transportation modes. In the same setting, [31] uses neural networks (NN) and particle swarm optimization to achieve an accuracy of 95%. Other papers as [16] consider data from the accelerometer in combination with data from the compass and magnetometer. The performance is satisfactory, both for the accuracy and from the amount of power that the collection of data requires. Unfortunately, several models of smartphones have neither a gyroscope nor a magnetometer.

From the analysis of the literature, we conclude that RF is the classification method that, in most of the cases, achieves the best results. As previously mentioned, one of the main weakness of the TMD literature is the absence of benchmark datasets. This problem is serious as, in different cities, the habits of people are different; hence, a good methodology for a city can have poor results in another one. For this reason, it is a necessary condition for the improvement of the field to build a set of benchmarks datasets available online. For the sake of completeness, it is important to cite some available applications that classify the transportation mode in real-time. The most famous are Modalyzer,[2] Chronology[3] and the Android Activity Recognition.[4] Modalyzer can detect if the user is traveling by bus or by car, but it needs the maximum possible GPS sampling frequency. Chronology is a feature included in Google Maps. Its objective is to monitor the movement of users by storing the transportation mode used and the places visited. Its accuracy is not very high and, sometimes, it fails to register some types of travel. The Android Activity Recognition APIs identify different activities. These APIs do not use GPS. They are able to determine if the user is walking, running, traveling by bicycle, or traveling by vehicle. These APIs are not able to distinguish between public and private transportation modes.

As some real-life experiments have shown, such as Singapore's Livable Places (see [32]), a clear understanding of transportation usage requires data collected with multiple methods (e.g., sensors, cameras, crowd-sourcing, and social networks). Thus, the collaboration of phones with other data sources (e.g., car, bus, and bike sensors or street sensors at traffic lights) improve tracking the moving habits of people. Nevertheless, such technology opportunities are not yet available in the place of our case study and will not be available in the (foreseeable, at least) future. Furthermore, in order to use these data, it is required to have such an infrastructure across the entire country that, especially in rural areas, is difficult to implement. For these reasons, we consider an easier approach that can involve, as users, the vast majority of the population.

The main contribution of this paper is to develop a framework for data collection and for the classification of transportation mode that is robust with respect to the sampling rate. This is strictly related to the problem of power consumption in data collection. This problem is of fundamental importance for building applications that users are willing to run on their smartphone. Furthermore, this paper deals with the distinction between private and public transportation modes. For these reasons, and due to the unavailability of public datasets, we cannot compare our results with the ones obtained by other papers. Finally, the proposed methodology considers as potential class of users of the application all people who own a smartphone. As top-class smartphones usually have sensors that others do not have, we only consider sensors that are present in the majority of smartphones.

III. DATA COLLECTION

In this section, we briefly describe the main characteristics of smartphone sensors and describe the data collection process. The most common smartphone sensor is the accelerometer. It measures accelerations applied to the device (in m/s^2) with respect to the three axes shown in Fig. 1. It is usually used for determining the orientation of the smartphone and rotating the screen horizontally or vertically. The acceleration due to gravity biases the measures of the acceleration.

Another sensor that is present in almost every smartphone is the GPS (Global Positioning System) sensor. It determines the coordinates (in latitude and longitude) of the smartphone, with a precision of a few meters. The localization is carried out through the reception of of a radio signal sent by satellites. The main problem of this sensor is that GPS signal is often missing indoor, and totally absent underground.

For the sake of completeness, we also describe other sensors used in similar studies: the gyroscope is a sensor that estimates the orientation of the device with high precision. It measures rotational velocity along three axes in rad/s. It is used to improve the information about the trajectory of a smartphone in space. Compared with the accelerometer, the gyroscope is not present in all smartphones. Thus, we have not used information coming from this sensor.

The magnetometer is a sensor measuring the magnetic field along the three Cartesian axes. It is used as a compass in applications implementing navigation features. This sensor is sensitive to metallic objects and other devices nearby. As with the gyroscope, it is used to improve information about the

[2]https://play.google.com/store/apps/details?id=com.modalyzer&hl=it

[3]https://support.google.com/maps/answer/6258979?co=GENIE.Platform%3DAndroid&hl=en

[4]https://developers.google.com/location-context/activity-recognition

TABLE 1. Summary of studies regarding transport mode detection (TMD).

Article	Classes Analyzed	Used sensors	Accuracy	Algorithm
[16]	walk, bicycle, car, bus, train and subway	accelerometer and gyroscope	99.96%	RF
[22]	walk, bicycle, car, bus, tram, subway, train, others and unknown	accelerometer, Wi-Fi, GPS, Cell-ID	70-80%	Bayesian probability
[31]	walk, bicycle, car, bus	GPS	95%	NN, PSO
[19]	walk, bicycle, car, tram	accelerometer, GPS	99.5%	RF
[20]	Walking, car, train, bus, tram	accelerometer, GPS	82%	RF, HMM
[21]	Walking, static, moving slowly, riding train, riding bus, running, car	accelerometer, GPS, GIS data	97%	RF merge to GIS, GPS data
[17]	Stationary, walk, bus, train, metro, tram, car	accelerometer	85%	HMM, Adaptive Boost
[23]	Stationary, walk, bus, train, metro, tram, car, motorcycle	accelerometer	82%	DT

smartphone trajectory. Nevertheless, it is not present in several models of smartphones and, so, we have disregarded it.

The microphone is a sensor that records sounds, and the extraction of data for classification from the microphone can be interesting: background noise is different between public and private transportation modes. Nevertheless, asking users to allow an application to record sounds may have a huge impact with respect to privacy. For this reason we do not consider it in our study.

Finally, the different communication standards (e.g., Wireless LAN, Code Division Multiple Access, GSM, Long Term Evolution, and Wideband Code Division Multiple Access provide other information that can be used for localization (e.g., buses may have private wireless LAN). With respect to GPS, they consume less power, but have low accuracy.

The energy consumed by sensors is not negligible and, in Table 2 ([33], [34]), we show the power consumption of smartphones during the execution of particular tasks. We add also tasks not related to data sampling, in order to give the reader an idea of the standard smartphone power consumption.

The power consumption of the GPS is tricky to analyze, for many reasons: first, GPS sensor manufacturers typically do not provide technical details; second, GPS consumption varies considerably between idle and active time intervals. The Adafruit Fona GSM module is a commercial sensor (technical data can be found in [35]); for which the manufacturer declares that the sensor consumes 20–25 mA when it is waiting for a message and up to 200 mA when it receives a signal, with a spike up to 2 A. Furthermore, the standard sampling frequency for a GPS is between 1–10 Hz. For this reason, Table 2 shows an average value for GPS power consumption.

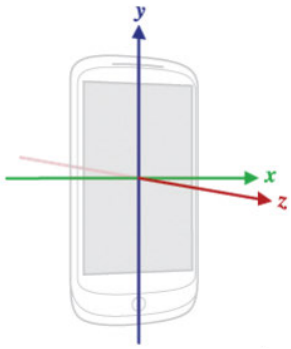

FIGURE 1. Acceleration axis.

TABLE 2. Power consumption of phone operations.

Application	Consume [mW]
Call	680
Reproducing musical contents	50
Record a video	930
Watch a video	660
Use of the accelerometer	21
Use of the magnetometer	48
Use of the gyroscope	130
Use of the microphone	105
Sampling GPS position	520
Background	140
Screen usage	470

As the reader may notice from Table 2, the accelerometer consumes only 21 mW; thus constituting the least amount of consumed power. Moreover, sampling the GPS position

consumes almost eight times the power of the accelerometer. Hence, the accelerometer is the ideal sensor to use for our analysis. Nevertheless, the information of just the accelerometer is not enough in order to distinguish between public and private transportation mode.

The first step of this study consists of data collection and the construction of a dataset containing information about the various sensors. To achieve this, an Android application called TraceMe[5] is developed, in order to collect as much data as possible (to allow a study of classifier performance for different sampling times) and to label the real travel mode. This application runs in the background and collects data from several sensors, as well as the output of the Android Activity Recognition APIs.

The app is implemented by using a free APIs available from the Android environment. The reasons for this choice are two-fold. First, it ensures that our results are repeatable; second, it does not require any extra costs for the interested reader. The data collected by TraceMe constitute the dataset used for the training and testing of the classification algorithms.

The structure of the final version of the project is composed of an application running in the background that collects the data from sensors, classify the transportation mode and sends them to a server. The server stores the data and performs the computation (such as updating the training of classifiers, if needed, and searching for sustainable alternatives to users' trips) and sends to the application possible improvements for the mobility habits of the user. At the same time, the server is responsible for sending data to the municipality suggesting possible optimization of the public transportation mode. It is worth noting that this architecture simplifies the management of the information, if the application is used by a large amount of people. As the description of the whole project is out of the scope of the present paper, we do not focus on the procedure that the server uses for computing sustainable alternatives, nor on the application running in the background. In fact, the application that we present is the one used for collecting the data for the training set of the classification algorithm and not the one to collect data running in the background. It is important to notice that we are referring to a generic server, but when the number of users increases, more complex solutions will have to be implemented ([36], [37]).

TraceMe asks its users to start recording data at the beginning of each trip, to end the registration at the end, and to communicate the transportation mode used. The main screen (Fig. 2) describes the information on the state of the application. It indicates the time when the data collection started, the transportation mode detected by the Android Activity Recognition APIs, the speed recorded in km/h, and the present activity selected by the user (among bicycle, bus, car, motorcycle, stationary, subway, train, tram, or walking). Furthermore, a button enables the user to start or stop the recording of travel (if the GPS tracker is enabled). Walking is the default transportation mode. When the start button is pressed, the device starts to record data. When the travel ends, the person can set the label and see it on a map. When the application is recording, it is possible to leave the application open or running in the background. The second screen (Fig. 3) shows the list of trips recorded by the user. It can be used by pushing the icon in the Action Bar. From this screen, it is possible to see, in chronological order, a summary of the information of each trip (starting and ending date, length, and duration).

By tapping on the travel, it is possible to see additional information (sampling frequency and synchronization status) and to modify the label of the trip.

For each recorded trip, the application creates a SQLite dataset containing the data from the accelerometer,

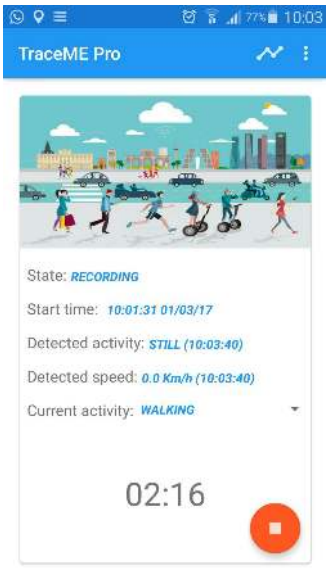

FIGURE 2. Screenshots of the application TraceMe: Home page.

FIGURE 3. Screenshots of the application TraceMe: List of all the travel.

[5]The application is not publicly available, but it can be obtained by asking the authors.

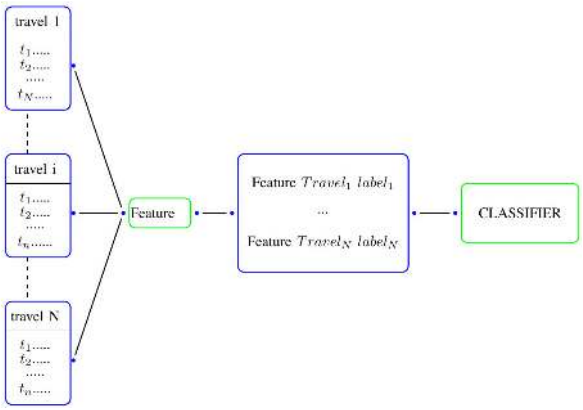

FIGURE 4. Classification model architecture.

the bluetooth, the gyroscope (if any), the GPS, the magnetometer (if any), microphone, the Wi-Fi, and the output of the Android Activity Recognition APIs. It is worth noting that despite the collection of several data, some of them are not used in the classification algorithm. We collect them in order to create a dataset as much comprehensive as possible.

Once the dataset is saved, it is compressed, sent to the server, and then it is deleted from the phone. As a more in-depth technical description of TraceMe is out of the scope of the present paper, we refer to [38] and [39] for more information.

After data collection, we obtain a database (DB) containing, for each trip, a dataset composed of a set of time-series describing accelerations, GPS positions, and the output of the Android Activity Recognition APIs. Due to the heterogeneity in the data (in terms of duration), we extract a set of features from each dataset. Hence, we obtain a unique table from the whole DB, where each row contains the values of the features of a specific trip and its respective label (public or private transportation mode). By convention, we associate the label *BUS* to the public transportation modes (bus, tram, train, or subway), and the label *CAR* to the private transportation modes (car and motorcycle). By using this strategy, we reduce the complexity of the classification method, as it does not have to deal with a huge amount of data. An alternative technique to manual feature extraction is automatic feature extraction. However, we decided not to use it, as we desire the obtained features to have a physical interpretation.

It is important to notice that the aforementioned procedure consists of an offline classification in which all the classification is performed in a PC after the travel is executed, rather than during the travel. This strategy is used only in the prototyping phase. Indeed, the classifier is fully integrated in the final version of the application.

IV. CLASSIFICATION ALGORITHM

From the set of features considered, we select the most significant to develop a light and efficient classifier and to reduce over-fitting. The final classification algorithm proposed is shown in Fig. 4.

The collected data undergo a series of processing steps. The first processing applied to the dataset is to extract the module of the acceleration. As acceleration values are biased by the action of the gravity force, we subtract the gravity acceleration from the collected data.

The obtained set of values is a time-series, which is a set of data, collecting the value of a phenomenon for several evenly spaced time instants. In the following, we consider $\mathcal{I} = \{1, \ldots, I\}$ the set of all travels recorded and $\mathcal{T}_i = \{1, \ldots, T_i\}$ the set of time steps of travel i. We call a_t^i $\forall i = 1, \ldots, I$, $t = 1, \ldots, T_i$ the value of the acceleration of travel i at time t. The auto-correlation at lag k is

$$r_k^i = \hat{Corr}[a_t^i, a_{t-k}^i] = \frac{\sum_{t=k+1}^{T_i}(a_t^i - \bar{a}^i)(a_{t-k}^i - \bar{a}^i)}{\sum_{t=k+1}^{T_i}(a_t^i - \bar{a}^i)^2}, \quad (1)$$

where $\bar{a}^i = \sum_{t=1}^{T_i} a_t^i$ is the average acceleration of travel i. It holds that the value r_k^i is normally distributed if the random variables a_t are independent and identically distributed [40].

By considering r_k^i $\forall$ k, we have the so-called Autocorrelogram Function (ACF), which is a summary of the dependence patterns over time.

Besides correlation, another important characteristic of a time-series is the frequency analysis. Features related to frequency use the Fourier transformation. In particular, given the acceleration time-series a_t^i, the spectrum of the series is defined as

$$A(\upsilon) = \sum_{t=-\infty}^{+\infty} a_t^i e^{-i2\pi\upsilon t}, \quad (2)$$

where υ is the measured frequency. The result of Eq. (2) is the description of the signal in terms of a sum of sinusoidal functions with different frequencies. We can use the information on the spectrum in order to obtain the frequency most present in the acceleration data and the corresponding amplitude.

Before to start describing the features that we use, we recall that given a vector $u = [u_0, \ldots u_N]$, the l_1 norm is $\sum_{i=0}^{N} |u_i|$, the l_2 norm is $\sqrt{\sum_{i=0}^{N} u_i^2}$ and the l_∞ norm is $\max_i u_i$.

The first features that we consider are the maximum ($\max_{t=1\ldots T_i}[a_t^i]$), the minimum ($\min_{t=1\ldots T_i}[a_t^i]$), and the average acceleration ($\bar{a}^i$) computed over each trip. Since, they are not enough to characterize the distribution of the acceleration, we also consider the number of times that the acceleration is above a threshold. We chose, as a threshold, the 70^{th} percentile of the acceleration data considering over all the trips i.e., the value $q_a^{0.7}$ such that $\mathbb{P}[a_t^i < q_a^{0.7}] = 0.7$. In this way, we can measure the number of extreme values of the acceleration distribution. This feature is particularly high for private vehicles characterized by high power. It is worth noting that the discrimination ability of the aforementioned features is reduced in high traffic conditions.

Furthermore, we consider the l_1, the l_2, and the l_∞ norms of the ACF r_k^i $\forall$ k by excluding the value $r_0^1 := 1$.

Usually, since buses, trams, trains, and metros have reserved paths, their acceleration is more correlated than the other transportation modes.

We also consider the l_∞ norm of the spectrum $A(\upsilon)$ defined in (2) and its argmax. The rationale behind this choice is that, in public transportation modes, there are more small vibrations than in private transportation modes.

Finally, we notice that, qualitatively, the time-series of the accelerations of public transportation modes are smoother than the series of private transportation modes. For this reason, we consider the variance of the values contained in a sliding window of length equal to one-tenth of the total number of observations. From this time-series, we compute the l_1, the l_2, and the l_∞ norms of its ACF (by excluding the first value that, by definition, is equal to 1).

To these features, we add the duration of the trip and its starting time. The reason for the usage of these two features is that people use transportation modes in a repetitive way.

Furthermore, we also consider the number of times that the Android APIs register tilting activities since we expect that people use the smartphone less if they are traveling by private transportation modes. The reason is that if the user is travelling by using a private transportation modes it is possible that he is driving and thus not tilting the phone (see [41] and [42] for further studies).

Finally, we also consider the output of a map-matching algorithm as a feature. Due to its complexity and importance, we explain this feature in Sec.IV-A. The final list of features is

1) the l_1 norm of the ACF;
2) the l_2 norm of the ACF;
3) the l_∞ norm of the ACF;
4) the l_1 norm of the spectrum;
5) the l_2 norm of the spectrum;
6) the l_∞ norm of the spectrum;
7) the l_1 norm of the ACF of the mobile variance;
8) the l_2 norm of the ACF of the mobile variance;
9) the l_∞ norm of the ACF of the mobile variance;
10) the minimum acceleration;
11) the maximum acceleration;
12) the average acceleration;
13) the travel duration;
14) the time of the start of the trip;
15) the number of times that the acceleration exceeded a threshold (set to the 70^{th} percentile of all the acceleration record);
16) number of registered tilting activities;
17) map-matching.

With the aforementioned set of features, we use Principal component analysis (PCA) to determine the subset that best explains the variation in the data. We recall that PCA computes the eigenvalues of the co-variance matrix of a set of measurements. Given a feature, the greater the eigenvalue, the more significant the associated features are.

Even if our classification has only two classes, we perform a tuning guided by PCA in order to use the optimal number of features that maximizes the performance of the method and mitigates over-fitting.

By using this method, we decided to classify the trips by using different sets of features, in order to see how the prediction power of the classifiers changed.

A. MAP-MATCHING

The map-matching algorithm compares GPS observations with the paths of public transportation lines. If they are close to each other, then the travel is considered to be executed with a public transportation mode; otherwise, it is considered to be executed by a private transportation mode. Algorithms of this type represent an important way to tackle the problem of TMD. For this reason, we consider this feature in the present section. The inputs of the algorithm are the points recorded by the GPS and the activities recorded by the Android APIs. The outputs are the guessed transportation mode used along with its associated probability. The first step of the algorithm tries to recognize if the travel has been travelled by subway. This classification is easy, since the GPS does not provide data when the smartphone is underground. If the test recognizes the subway, it produces this result in output with probability 1; otherwise, further tests are performed. The second test checks if the path has been travelled by train. In particular, the algorithm compares the GPS positions with the paths of the trains from OpenStreetMap. By calling π_{TRAIN} the fraction of points having a distance less than 100 meters from the railway, if $\pi_{TRAIN} > 75\%$ then the path is labeled as being executed by train with probability π_{TRAIN}; otherwise, further tests are performed.

The third test tries to distinguish if the travel has been travelled by using a private transportation mode or by using a public transportation mode, by comparing the GPS positions with public transportation lines. We call π_{BUS} of the fraction of points that are less distant than 100 meters from a public transport line. If $\pi_{BUS} > 50\%$, then the travel is labeled as done with a public transportation mode with a probability π_{BUS}. If, instead, $\pi_{BUS} < 50\%$, then the travel is labeled as done with a private transportation mode with a probability $1 - \pi_{BUS}$.

Finally, $\pi_{BUS} = 1$ if the first and last point of the travel are near to a bus station and if the activities recorded by the Android APIs is first walking, then in a vehicle, and then walking again.

It is worth noting that in the algorithm there is no consideration for the people that by using their cars follow the same route as a bus. This is a drawback of the map matching algorithm, but in several cases the machine learning algorithm correctly detects these situations by analyzing the statistical properties of the acceleration patterns.

The pseudo-code of the procedure is shown in Algorithm 1.

Since the algorithm uses GPS data, the device consumes a lot of power. Furthermore, a high computational burden is involved by the comparison of every GPS position with all the stops of the public transportation mode. For these reasons, it is interesting to understand how the performance of the method

Algorithm 1 Map-Matching

if first and last points are near the metro and no other point **then**
  $p = (METRO, 1)$
else
  for point in GPS_record **do**
    count_Train + = point $\sim$ Train_path
  π_{TRAIN} := count_Train / len(GPS_record)
  if $\pi_{TRAIN} > 75\%$ **then**
    $p = (TRAIN, \pi_{TRAIN})$
  else
    for point in GPS_record **do**
      count_Bus + = point $\sim$ Bus_path
    π_{BUS} := count_Train / len(GPS_record)
    if $\pi_{BUS} > 50\%$ **then**
      if first and last points are near the bus and WALKING before and after **then**
        $p = (BUS, 1)$
      else
        $p = (BUS, \pi_{BUS})$
    else
      $p = (CAR, 1 - \pi_{BUS})$
return p

degrades with respect to variation in the sampling time. As a performance indicator, we use the recall:

$$r_x = \frac{\text{number of travels traveled by X labeled correctly}}{\text{number of travels traveled by X}}, \quad (3)$$

where $X \in \{BUS, CAR\}$. It is the percentage of trips traveled by X that we have been correctly identified.

In particular, we are interested in r_{BUS} since, if the data are under-sampled, the algorithm is more likely to label the travel as 'CAR'. The variations of r_{BUS} with respect to the sampling frequency are shown in Fig. 5.

As the reader may notice, if the sampling time doubles from 1 Hz to 2 Hz, the recall does not significantly worsen; however, if it triples, the recall reaches values around 0.5. We can conclude that the map-matching algorithm is able to achieve good results only if the data are sampled with a high rate (i.e., if a lot of power is consumed). For this reason, it is not suitable to use the map-matching algorithm as the sole method for classification. it is important to underline that the outcome of this algorithm will be used as a feature for the classification approach presented in the next Section.

As aforementioned, the map-matching algorithm needs a lot of energy in order to sample the GPS positions. Furthermore, it does not use the acceleration data which sampling consumes a small amount of energy. From these considerations, it follows the need to define a new techniques that is able to reduce the number of data required by the map-matching algorithm by using the acceleration data.

In the following, we consider four classifiers: Decision Tree, Support Vector Machine (SVM), Random Forest (RF),

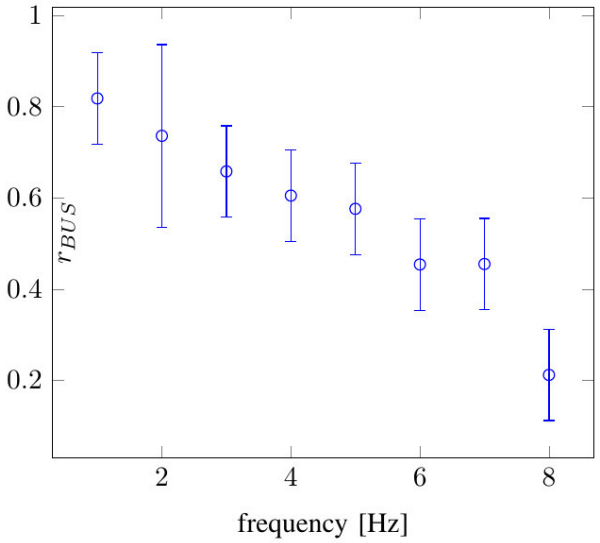

FIGURE 5. Recall of the map-matching algorithm with different sampling times.

and Naïve Bayes. We have chosen not to consider other nonlinear classifiers, such as neural networks and nonlinear support vector machines, because we already have features that are nonlinear in the original data. Moreover, we do not use data clustering techniques (such as the one used in [43], in [44] and in [45]) because in this setting, they perform far worst that the other techniques.

V. NUMERICAL RESULTS

In this Section, we consider various classifiers and compare their performance on the real data collected thought TraceMe (the aforementioned smartphone application). We develop the code using Python 3.6 leveraging the availability of packages for machine learning. In particular, as performance indicators, we consider the recall (see (3)) and precision:

$$p_x = \frac{\text{number of travels traveled by X labeled correctly}}{\text{number of travels labeled as X}}, \quad (4)$$

where $x \in \{BUS, CAR\}$. Precision is the probability that label X is correct.

The final dataset is composed of 311 trips performed by 30 different people, in different moments of the day.

For comparing the proposed methodology with other methods, we searched online publicly available dataset but unluckily we did not find any data.

Thus, we decided to freely provide our data.[6] In all experiments, we randomly divided the observations into a training set (containing 70% of the observations) and a test set (containing the remaining 30%). Every experiment is performed 100 times, in order to obtain robust results. The first experiment considered all the 17 features together. The results are shown in Table 3.

As the application consumes a lot of power in gathering GPS information, we also show the results of the

[6]http://www.orgroup.polito.it/resources.html

TABLE 3. Comparison between the classifiers by using all the 17 features.

	r_{CAR}	r_{BUS}	p_{CAR}	p_{BUS}
RF	0.88 (0.08)	0.93 (0.06)	0.93 (0.06)	0.89 (0.06)
DT	0.90 (0.10)	0.87 (0.09)	0.87 (0.09)	0.91 (0.09)
SVM	0.89 (0.09)	0.73 (0.13)	0.77 (0.11)	0.88 (0.08)
NB	0.89 (0.06)	0.90 (0.07)	0.89 (0.07)	0.89 (0.06)

TABLE 4. Comparison between the classifiers by all features except map-matching.

	r_{CAR}	r_{BUS}	p_{CAR}	p_{BUS}
RF	0.84 (0.08)	0.93 (0.07)	0.92 (0.08)	0.85 (0.07)
DT	0.83 (0.13)	0.80 (0.15)	0.81 (0.12)	0.83 (0.19)
SVM	0.74 (0.10)	0.72 (0.13)	0.72 (0.12)	0.74 (0.09)
NB	0.80 (0.01)	0.83 (0.09)	0.83 (0.09)	0.81 (0.09)

classifiers when the map-matching features are not considered in Table 4.

The best classifier is the RF, having the best precision and recall, where both are over 90%. An interesting fact is that, without the map-matching feature, the performance of the RF slightly decreased.

As the usage of 17 features may lead to over-fitting and since our aim is to reduce battery usage, we apply a feature selection procedure. In details, we consider the five features with highest coefficients in the first six principal components (which explained 95.96% of the variance). These are: the l_2 norm of the mobile variance, the result of the map-matching algorithm, the average, the l_1 norm of the spectrum, and the number of times that the series exceeded the extreme quantile. It is important to notice that while some of them are can be guessed a priori (e.g., the results of the map-matching algorithm) some others are not (e.g., the l_1 norm of the spectrum). For this reason, the usage of the PCA has been of fundamental importance.

In the following subsections, we analyze the results of the classifiers when considering these five features while reducing the sampling time. In particular, in Subsection V-.1, we consider the performance of the Decision Tree; in Subsection V-.2, we consider the SVM; in Subsection V-.3, the RF; and, finally, in Subsection V-.4, the Naïve Bayes. For each classifier we plot the p_{CAR}, p_{BUS}, r_{CAR}, r_{BUS} against the acceleration sampling rate (i.e., ν_A) for different GPS sampling rate (i.e., ν_S). In particular, for the acceleration sampling rate we consider $\nu_A \in [0.1s, 2s]$. We do not reduce more the sampling frequency, as the power consumption of the accelerometer with a sampling frequency of 2 seconds is negligible. Instead, for the GPS sampling rate we consider $\nu_G = 10s$, $\nu_=20s$, $\nu_=30s$, $\nu_=40s$ and no GPS information (i.e. $\nu_G = \infty s$).

1) DECISION TREE

A decision tree is a classifier that uses a tree-like graph to discriminate between two classes, where the leaves of the tree

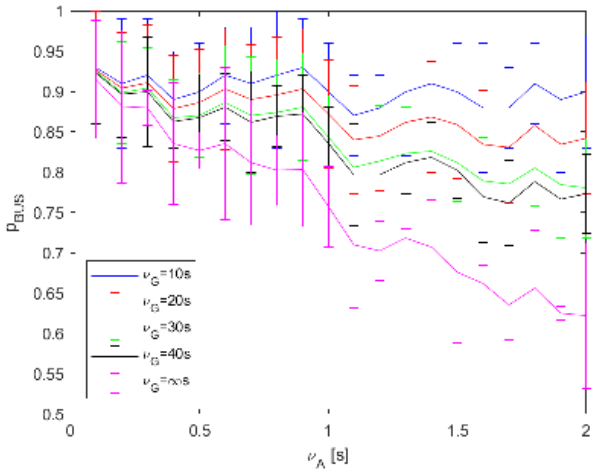

FIGURE 6. p_{BUS} of the decision tree for different values of ν_A and ν_G.

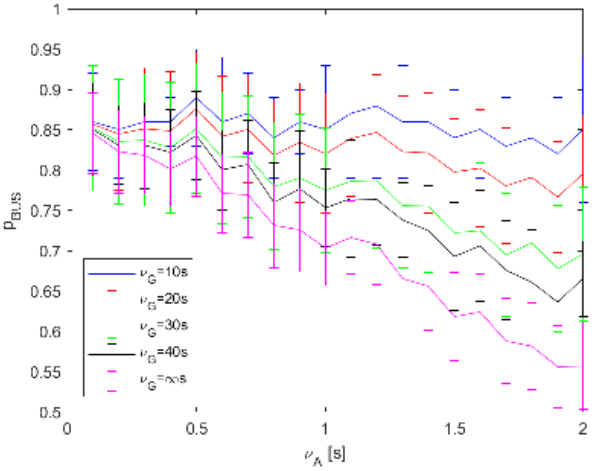

FIGURE 7. p_{CAR} of the decision tree for different values of ν_A and ν_G.

contain the selected classes (see [46] for more details). In our experiments, we used the implementation DecisionTreeClassifier, available in the package sklearn [47].

As we can see, from Figures 6, 7, 8 and 9, the classifier is robust with respect to the number of acceleration samples considered (i.e. ν_A) if $\nu_G = 10s$. In fact, all the reported values are not statistically different. However, the classifier is very sensitive with respect to the sampling time of the GPS position. This is an effect of the degradation of performance in the map-matching algorithm: since map-matching is a good parameter to discriminate, the most important rule of the decision tree considered the map-matching feature. Once this feature became unreliable, the decision tree lost an important part of its discrimination power.

2) SUPPORT VECTOR MACHINE

A support vector machine (SVM) is a classification method that tries to divide the two class of observations by means of a hyperplane (see [46] for more details). If the two classes cannot be divided, then the SVM finds the hyperplane that

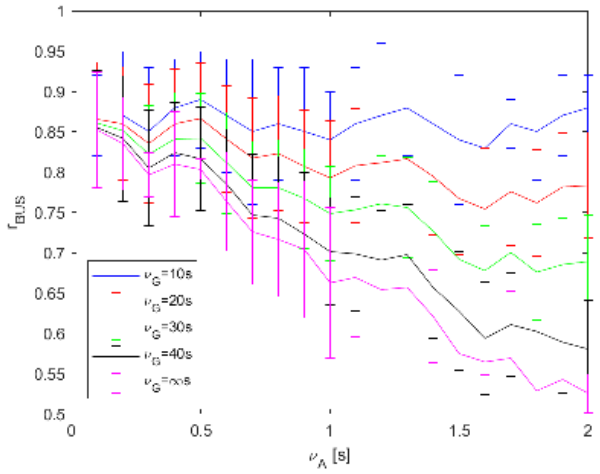

FIGURE 8. r_{BUS} of the decision tree for different values of ν_A and ν_G.

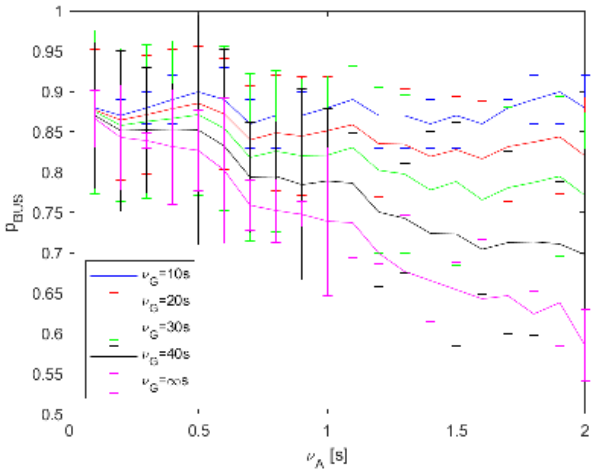

FIGURE 10. p_{BUS} of the SVM for different values of ν_A and ν_G.

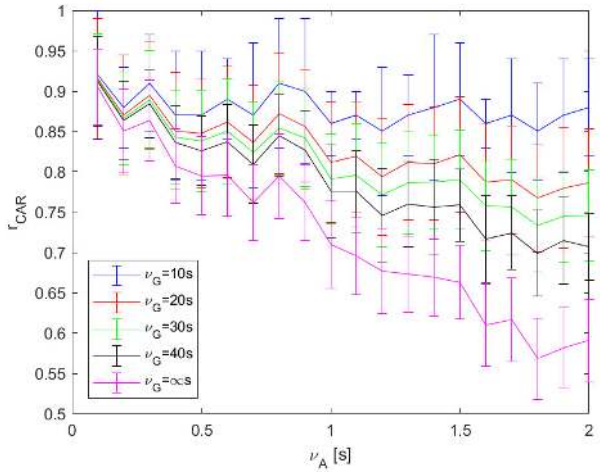

FIGURE 9. r_{CAR} of the decision tree for different values of ν_A and ν_G.

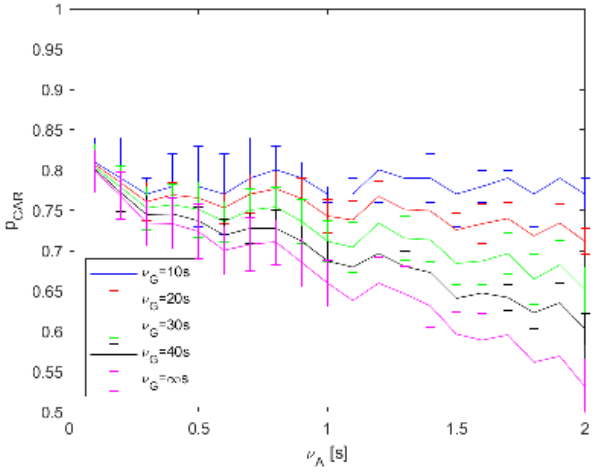

FIGURE 11. p_{CAR} of the SVM for different values of ν_A and ν_G.

minimizes the number of errors. The implementation of SVM used in our experiments is the function svm of the package sklearn [47].

The performance of the SVM for different values of sampling times are shown in Figures 10, 11, 12 and 13. As the reader can notice, the results of precision and recall are not statistically different from one setting to another (if $\nu_G = 10s$) and, in general, the standard deviations are lower than the one of the decision tree. This behavior is explained by the fact that the SVM is obtained by solving in an exact way a convex optimization problem. When we reduced the sampling time of the GPS to be less than 10 seconds, the performance degrade fast. An interesting fact is that the decrements of p_{CAR}, and r_{BUS} when ν_G passes from $40s$ to ∞s is not greater than the others. This means that the importance of the map-matching decreases as the sampling rate increases. It is worth noting that the SVM achieves the lowest results in both p_{CAR} and r_{BUS} with respect to the other classifiers. In particular, even with the smallest sampling time $p_{CAR} = 0.81$ and $r_{BUS} = 0.76$. This observation, and the low variance in the results lead the SVM to be, on average, the worst classifier that we have considered.

The reason of this behaviour can be explained in the poor discrimination power of linear functions used by the classifier. It is important to notice that, even with all the 17 features, the SVM does not achieve accurate results.

3) RANDOM FOREST

A RF is a classifier composed of a set of decision trees. It averages multiple decision trees trained on different parts of the training set. In this way, it reduces the over-fitting that a single tree can produce (see [46] for more details). In our experiments, we used the class RandomForestRegressor from the package sklearn [47].

Figures 14, 15, 16 and 17 show the performance of the RF for various acceleration sampling times. As with the SVM, the RF classifier is also robust with respect to changes in the GPS sampling times. Furthermore, the performance of the RF is also stable with respect to the different sampling times of the GPS positions (if $\nu_G = 10s$).

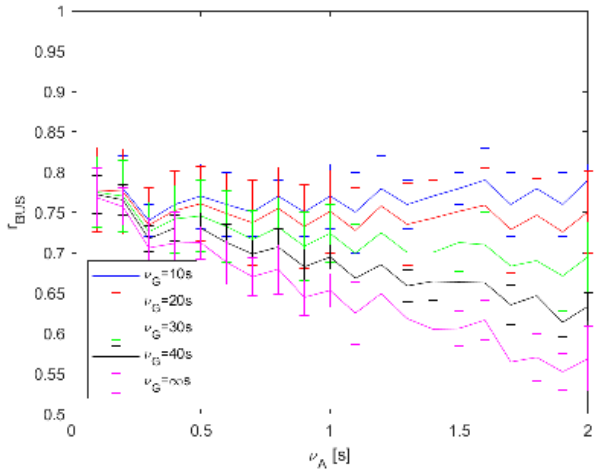

FIGURE 12. r_{BUS} of the SVM for different values of ν_A and ν_G.

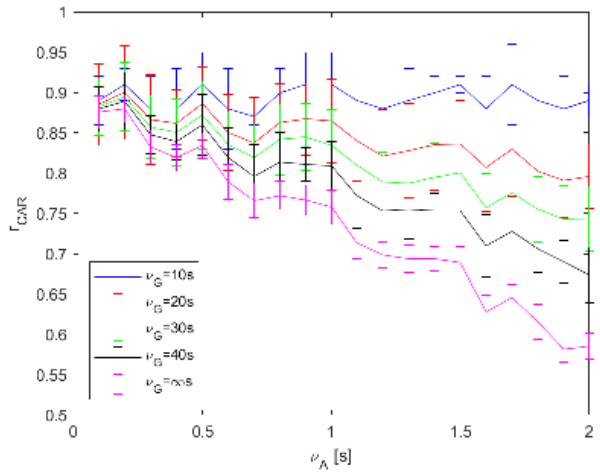

FIGURE 13. r_{CAR} of the SVM for different values of ν_A and ν_G.

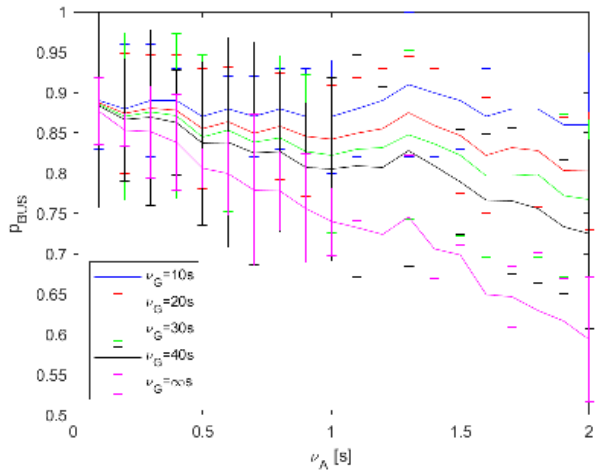

FIGURE 14. p_{BUS} of the RF for different values of ν_A and ν_G.

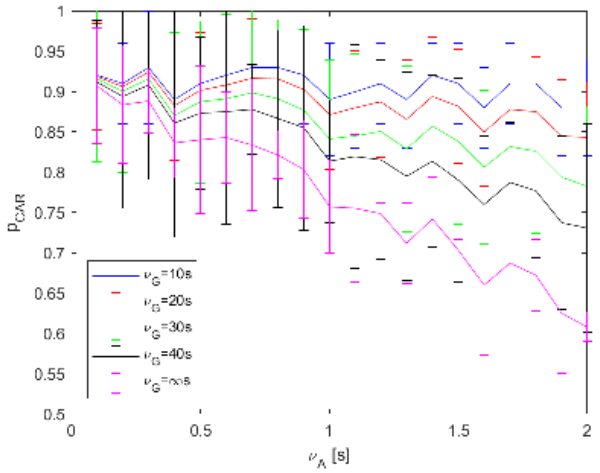

FIGURE 15. p_{CAR} of the RF for different values of ν_A and ν_G.

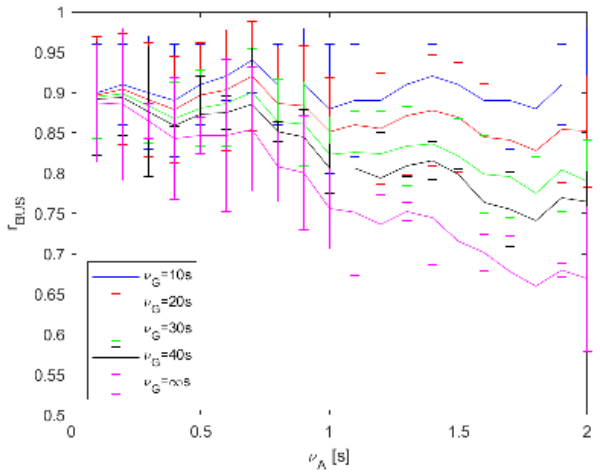

FIGURE 16. r_{BUS} of the RF for different values of ν_A and ν_G.

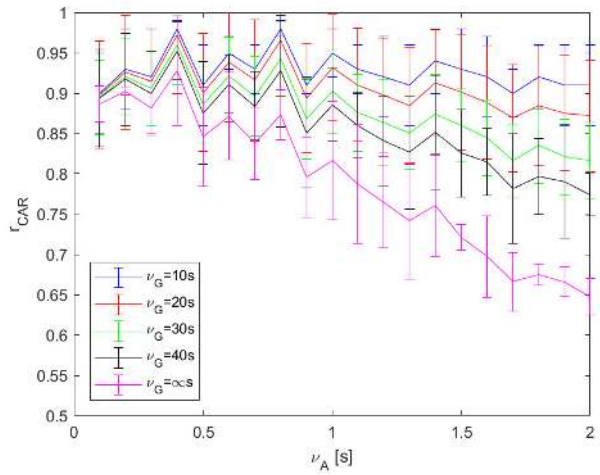

FIGURE 17. r_{CAR} of the RF for different values of ν_A and ν_G.

Finally, as with the above classifiers, its performance degrade if no GPS information is given and the sampling frequency of the acceleration is low, but it is important to notice that the RF classifier is the one having the least performance decrements without GPS information ($\nu_G = \infty s$). As the reader can notice, the reduction in the performance if ν_G

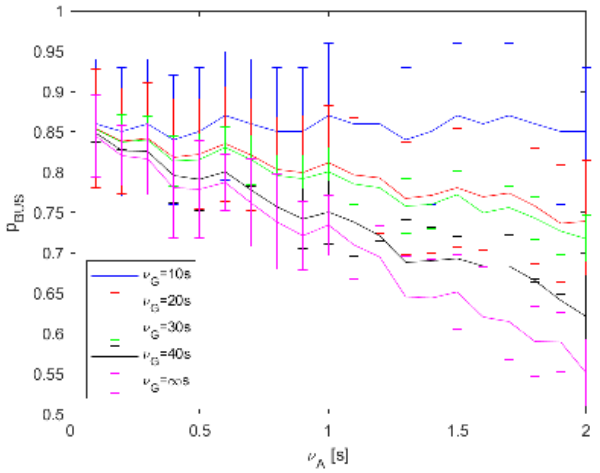

FIGURE 18. p_{BUS} of the Naïve Bayes classifier for different values of v_A and v_G.

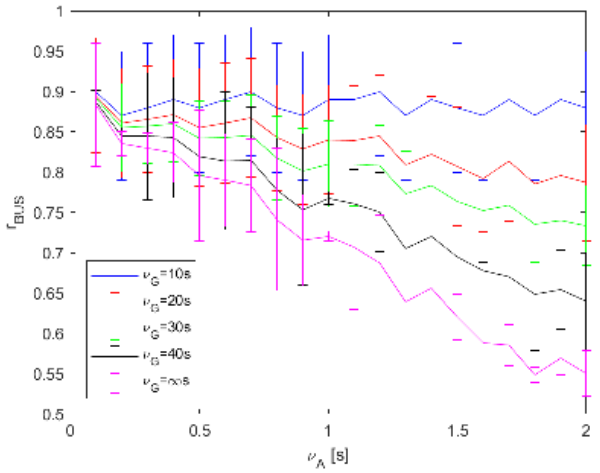

FIGURE 20. r_{BUS} of the Naïve Bayes classifier for different values of v_A and v_G.

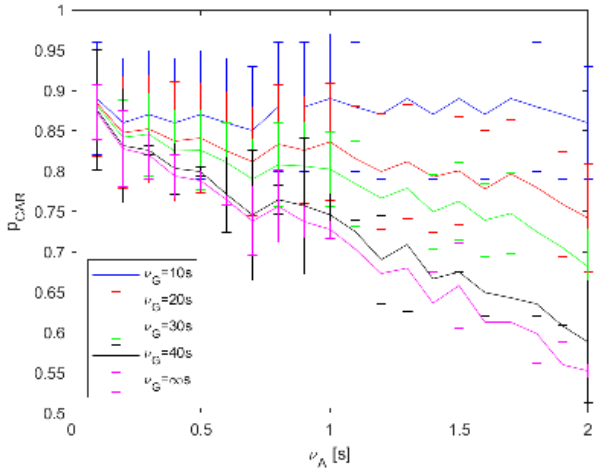

FIGURE 19. p_{CAR} of the Naïve Bayes classifier for different values of v_A and v_G.

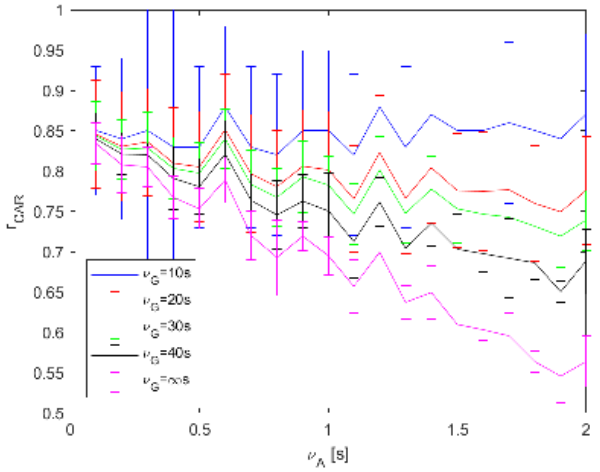

FIGURE 21. r_{CAR} of the Naïve Bayes classifier for different values of v_A and v_G.

passes from 40s to ∞s is greater than the other reduction (i.e., form 10s to 20s, from 20s to 30s and from 30s to 40s). This means that even with $v_G = 40s$ the map-matching feature is still really important for this classifier. Furthermore, the performance of the RF are better than the ones of the other classifiers in almost all settings. This confirms that the RF is the best classifier for the considered TMD problem.

4) NAÏVE BAYES

Naïve Bayes is a classifier based on the application of Bayes' theorem with strong (naïve) independence assumptions between the features. This approach has proved to be useful in several settings, such as in [48] (see [46] for more details). Given a set of discrete features $f_1, \ldots, f_n$, and by calling x the label, the Naïve Bayes classifier infers $P(x|f_1)P(x|f_2)\ldots P(x|f_n)$ by fitting these probabilities to the training set. In our experiments, we used the class BernoulliNB from the package sklearn [47].

Similar to the other classifiers, the performance of the Naïve Bayes are shown in Figures 18, 19, 20 and 21. The performance are not influenced by different sampling times of the acceleration.

Similar to the other classifiers, the performance of the Naïve Bayes also degrade if GPS positions are not considered and the acceleration is sampled with low frequency. This behavior is due to the fact that, if some feature had degraded, the classifiers could still use the others to achieve a accurate performance. Nevertheless, if the quality of all the features degraded, then the performance of the algorithm also degraded. As the reader can notice, on average the standard deviations of the performances produced by this classifiers are higher than the ones of the other classifiers.

In conclusion, all the classifiers behaved properly when the sampling time is high. For this reason, the best classifier to choose to implement this strategy should be one that performs

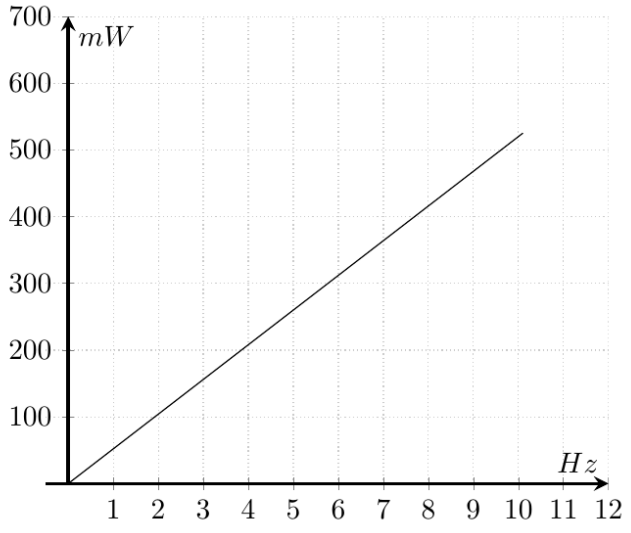

FIGURE 22. Linear interpolation of GPS power consumption.

well with low sampling time and, in particular, even without information from the GPS sensor.

The best results, from the point of view of energy consumption and performance are the one achieved for $\nu_A = 0.7s$ and $\nu_S = \infty s$ (i.e. without GPS information) by the RF classifier. It is able to reach $p_{CAR} = 0.80$, $p_{BUS} = 0.78$, $r_{CAR} = 0.83$ and $r_{BUS} = 0.82$. Nevertheless, since the GPS positions are interesting to be sampled for analysing people flow along the city and optimizing public transportation, the setting used in the final application is $\nu_A = 0.7s$ and $\nu_S = 40\ s$.

We recall that the use of GPS leads also to comparisons between paths, thereby requiring a huge amount of computational power. Concerning the classifiers, the best results are obtained by the RF methods. It is capable of achieving 90% accuracy by using a sampling time of 200 ms for the acceleration and a sampling time of 40 s for the GPS. These sampling times significantly reduce the power consumption of the application. In particular, if we consider a linear model for the power consumption of sensors versus the sampling frequency (see the one shown in Figure 22), then the proposed approach consumes 250 mW, which is a reasonable amount, since it is about half of the screen usage consumption (see Table 2). In order to measure in the real setting the energy consumption when considering maximum $40s$ and minimum $10s$ sampling time, we consider the percentage difference between the battery duration. We execute 10 observations of the time a mobile phone (an Huawei P9 lite) running only TraceMe takes to reach the 10% of the charge by starting fully charged. From these observations, we can estimate that by using the maximum sampling time ($40\ s$), on average the life of the battery of the mobile phone increases by the 25.4% with respect to the standard sampling setting. This is a satisfactory result, since it enables the application to decrease the impact it has on the duration of the battery and to the usability of the phone. For all the aforementioned reasons, TIM experts adopted this solution.

VI. CONCLUSION

In conclusion, we have proposed an innovative way for dealing with the problem of detecting whether a travel is executed by using a public or private transportation mode using data collected by a smartphone, while addressing the issue of power consumption. By doing so, we have developed a classification framework which is robust with respect to the sampling time. It is important to notice that, the introduction of the problem of power consumption in this branch of the literature is fundamental for its real applicability but, so far, it has been largely overlooked. The analysis of the performance of the classifiers with respect to under-sampled data is an important topic that has received little attention by the scientific community, despite its importance. In future work we will deepen this analysis in more general settings.

Furthermore, we have confirmed the result of several previous papers that consider Random Forrest to be the best classifier for the transportation mode detection problem. Finally, a benefit of the present work is to provide, for the first time, publicly available data that can be used as a benchmark or to reproduce or improve our results. Future development of the methodology will consider ad-hoc bootstrap techniques in order to reduce even more the number of samples used for the training of the classifiers, thus reducing the energy consumption.

REFERENCES

[1] M. A. de Almeida, J. F. Pereira, and J. A. Porfirio, "E-work at a global scale and its impact on employment and management practices," in *Proc. 6th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2011, pp. 1–4.

[2] R. Tadei, E. Fadda, L. Gobbato, G. Perboli, and M. Rosano, "An ICT-based reference model for E-grocery in smart cities," in *Proc. 1st Int. Conf. Smart Cities*, vol. 9704. Berlin, Germany: Springer-Verlag, 2016, pp. 22–31, doi: 10.1007/978-3-319-39595-1_3.

[3] E. Fadda, D. Mana, G. Perboli, and R. Tadei, "Multi period assignment problem for social engagement and opportunistic IoT," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2017, pp. 760–765.

[4] E. Fadda, G. Perboli, and R. Tadei, "Customized multi-period stochastic assignment problem for social engagement and opportunistic IoT," *Comput. Oper. Res.*, vol. 93, pp. 41–50, May 2018.

[5] E. Fadda, G. Perboli, and R. Tadei, "A progressive hedging method for the optimization of social engagement and opportunistic IoT problems," *Eur. J. Oper. Res.*, vol. 277, no. 2, pp. 643–652, Sep. 2019.

[6] T. Oransirikul, R. Nishide, I. Piumarta, and H. Takada, "Measuring bus passenger load by monitoring Wi-Fi transmissions from mobile devices," *Procedia Technol.*, vol. 18, pp. 120–125, Dec. 2014.

[7] G. Perboli, R. Tadei, and E. Fadda, "New valid inequalities for the two-echelon capacitated vehicle routing problem," *Electron. Notes Discrete Math.*, vol. 64, pp. 75–84, Feb. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S157106531830009X

[8] A. Prelipcean, G. Gidófalvi, and Y. Susilo, "Transportation mode detection—An in-depth review of applicability and reliability," *Transp. Rev.*, vol. 37, pp. 1–23, Oct. 2016.

[9] G. Solmaz and D. Turgut, "A survey of human mobility models," *IEEE Access*, vol. 7, pp. 125711–125731, 2019.

[10] E. Fadda, G. Perboli, V. Vallesio, and D. Mana, "Sustainable mobility and user preferences by crowdsourcing data: The open Agora project," in *Proc. IEEE 14th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2018, pp. 1243–1248.

[11] X. Wan, H. Ghazzai, and Y. Massoud, "Mobile crowdsourcing for intelligent transportation systems: Real-time navigation in urban areas," *IEEE Access*, vol. 7, pp. 136995–137009, 2019.

[12] C. Carpineti, V. Lomonaco, L. Bedogni, M. D. Felice, and L. Bononi, "Custom dual transportation mode detection by smartphone devices exploiting sensor diversity," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2018, pp. 367–372.
[13] E. Fadda, L. Gobbato, G. Perboli, M. Rosano, and R. Tadei, "Waste collection in urban areas: A case study," *Interfaces*, vol. 48, no. 4, pp. 307–322, Aug. 2018.
[14] E. Fadda, G. Perboli, and G. Squillero, "Adaptive batteries exploiting on-line steady-state evolution strategy," in *Applications of Evolutionary Computation*, G. Squillero and K. Sim, Eds. Cham, Switzerland: Springer, 2017, pp. 329–341.
[15] E. Fadda, D. Manerba, G. Cabodi, P. Camurati, and R. Tadei, "KPIs for optimal location of charging stations for electric vehicles: The Biella case-study," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, in Annals of Computer Science and Information Systems, vol. 18, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds. Warsaw, Poland: Polish Information Processing Society, 2019, pp. 123–126, doi: 10.15439/2019F171.
[16] M. Shafique and E. Hato, "Travel mode detection with varying smartphone data collection frequencies," *Sensors*, vol. 16, no. 5, p. 716, 2016.
[17] H. Bello-Salau, A. Aibinu, E. Onwuka, J. Dukiya, M. Bima, A. Onumanyi, and T. Folorunso, "A new measure for analysing accelerometer data towards developing efficient road defect profiling systems," *J. Sci. Res. Rep.*, vol. 7, no. 2, pp. 108–116, Jan. 2015.
[18] M. A. Shafique and E. Hato, "Modelling of accelerometer data for travel mode detection by hierarchical application of binomial logistic regression," *Transp. Res. Procedia*, vol. 10, pp. 236–244, Jan. 2015.
[19] M. A. Shafique and E. Hato, "Use of acceleration data for transportation mode prediction," *Transportation*, vol. 42, no. 1, pp. 163–188, Jan. 2015.
[20] D. Shin, D. Aliaga, B. Tuncer, S. M. Arisona, S. Kim, D. Zund, and G. Schmitt, "Urban sensing: Using smartphones for transportation mode classification," *Comput., Environ. Urban Syst.*, vol. 53, pp. 76–86, Sep. 2015.
[21] R. Guinness, "Beyond where to how: A machine learning approach for sensing mobility contexts using smartphone sensors," *Sensors*, vol. 15, no. 5, pp. 9962–9985, May 2015.
[22] K. T. Geurs, T. Thomas, M. Bijlsma, and S. Douhou, "Automatic trip and mode detection with move smarter: First results from the Dutch mobile mobility panel," *Transp. Res. Procedia*, vol. 11, pp. 247–262, Jan. 2015.
[23] V. Manzoni, D. Maniloff, K. Kloeckl, and C. Ratti, "Transportation mode identification and real-time CO_2 emission estimation using smartphones how CO_2GO works," Tech. Rep., 2011.
[24] O. Lorintiu and A. Vassilev, "Transportation mode recognition based on smartphone embedded sensors for carbon footprint estimation," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 1976–1981.
[25] T. Feng and H. J. P. Timmermans, "Transportation mode recognition using GPS and accelerometer data," *Transp. Res. C, Emerg. Technol.*, vol. 37, pp. 118–130, Dec. 2013.
[26] P. Nitsche, P. Widhalm, S. Breuss, and P. Maurer, "A strategy on how to utilize smartphones for automatically reconstructing trips in travel surveys," *Procedia-Social Behav. Sci.*, vol. 48, pp. 1033–1046, Jan. 2012.
[27] H. I. Ashqar, M. H. Almannaa, M. Elhenawy, H. A. Rakha, and L. House, "Smartphone transportation mode recognition using a hierarchical machine learning classifier and pooled features from time and frequency domains," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 244–252, Jan. 2019.
[28] G. Xiao, Z. Juan, and C. Zhang, "Travel mode detection based on GPS track data and Bayesian networks," *Comput., Environ. Urban Syst.*, vol. 54, pp. 14–22, Nov. 2015.
[29] H. Gong, C. Chen, E. Bialostozky, and C. T. Lawson, "A GPS/GIS method for travel mode detection in New York City," *Comput., Environ. Urban Syst.*, vol. 36, no. 2, pp. 131–139, Mar. 2012.
[30] M. Bierlaire, J. Chen, and J. Newman, "A probabilistic map matching method for smartphone GPS data," *Transp. Res. C, Emerg. Technol.*, vol. 26, pp. 78–98, Jan. 2013.
[31] G. Xiao, Z. Juan, and J. Gao, "Travel mode detection based on neural networks and particle swarm optimization," *Information*, vol. 6, no. 3, pp. 522–535, 2015.
[32] T. Abdulazim, H. Abdelgawad, K. M. N. Habib, and B. Abdulhai, "Using smartphones and sensor technologies to automate collection of travel data," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2383, no. 1, pp. 44–52, Jan. 2013.
[33] P. K. D. Pramanik, N. Sinhababu, B. Mukherjee, S. Padmanaban, A. Maity, B. K. Upadhyaya, J. B. Holm-Nielsen, and P. Choudhury, "Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage," *IEEE Access*, vol. 7, pp. 182113–182172, 2019.
[34] C. Vasile, C. Pattinson, and A.-L. Kor, *Mobile Phones and Energy Consumption*. New York, NY, USA: Springer-Verlag, 2019, pp. 243–271.
[35] Adafruit. (2019). *Adafruit Fona*. Accessed: Apr. 25, 2019. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-fona-mini-gsm-gprs-cellular-phone-module.pdf
[36] E. Fadda, P. Plebani, and M. Vitali, "Optimizing monitorability of multi-cloud applications," in *Advanced Information Systems Engineering* (Lecture Notes in Computer Science), vol. 9694. New York, NY, USA: Springer-Verlag, 2016, pp. 411–426.
[37] E. Fadda, P. Plebani, and M. Vitali, "Monitoring-aware optimal deployment for applications based on microservices," *IEEE Trans. Services Comput.*, early access, Apr. 11, 2019, doi: 10.1109/TSC.2019.2910069.
[38] L. Ruffino, "Open data e algoritmi di map matching per la classificazione degli spostamenti in ambito urbano," M.S. thesis, Politecnico di Torino, Turin, Italy, 2017.
[39] G. Maiorino, "Data analysis e machine learning per la classificazione dell'uso dei mezzi di trasporto in ambito urbano," M.S. thesis, Politecnico di Torino, Turin, Italy, 2017.
[40] R. Tsay, *Analysis of Financial Time Series*. Hoboken, NJ, USA: Wiley, 2002.
[41] D. E. Haigney, R. G. Taylor, and S. J. Westerman, "Concurrent mobile (cellular) phone use and driving performance: Task demand characteristics and compensatory processes," *Transp. Res. F, Traffic Psychol. Behav.*, vol. 3, no. 3, pp. 113–121, Sep. 2000.
[42] J. S. Hafetz, L. S. Jacobsohn, J. F. García-España, A. E. Curry, and F. K. Winston, "Adolescent drivers' perceptions of the advantages and disadvantages of abstention from in-vehicle cell phone use," *Accident Anal. Prevention*, vol. 42, no. 6, pp. 1570–1576, Nov. 2010.
[43] A. Cuzzocrea, M. M. Gaber, E. Fadda, and G. M. Grasso, "An innovative framework for supporting big atmospheric data analytics via clustering-based spatio-temporal analysis," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 9, pp. 3383–3398, Sep. 2019, doi: 10.1007/s12652-018-0966-1.
[44] S. Proto, E. Di Corso, F. Ventura, and T. Cerquitelli, "Useful ToPIC: Self-tuning strategies to enhance latent Dirichlet allocation," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, Jul. 2018, pp. 33–40.
[45] E. Di Corso, F. Ventura, and T. Cerquitelli, "All in a Twitter: Self-tuning strategies for a deeper understanding of a crisis tweet collection," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 3722–3726.
[46] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer-Verlag, 2001.
[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[48] A. G. C. de Sá, A. C. M. Pereira, and G. L. Pappa, "A customized classification algorithm for credit card fraud detection," *Eng. Appl. Artif. Intell.*, vol. 72, pp. 21–29, Jun. 2018.

PINO CASTROGIOVANNI is currently a Software Engineer and a Researcher, where he has been working with the Telecom Italia Group, since 1998. He has been involved in several Research and Development projects in the ICT field, including different European funded projects. He also works at the TIM Artificial Intelligence Joint Open Lab, where he is responsible for the coordination of research activities in collaborations with both academic and industrial partners. His latest research interests are focused on artificial intelligence and machine learning in the context of the Internet of Things applications and Telco network and service management systems.

EDOARDO FADDA received the Laurea degree *(summa cum laude)* in mathematical engineering and the Ph.D. degree in information technology and system engineering from the Politecnico di Torino, Torino, Italy, in 2014 and 2018, respectively. Since 2017, he has been an Optimization Engineer and a Data Scientist with moltosenso. He is currently a Postdoctoral Researcher with the Dipartimento di Automatica e Informatica, Politecnico di Torino, and a Researcher with ICE Lab. His main interests are optimization under uncertainty, statistics, and their applications.

GUIDO PERBOLI is currently a Professor of strategic management and operations research with the Politecnico di Torino, and an Associate Member of the Centre Interuniversitaire de Recherche sur les Réseaux d'Enterprise, la Logistique et les Transport (CIRRELT), Québec City, QC, Canada. In 2016, he founded the ICT for City Logistics and Enterprises (ICE) Center, Politecnico di Torino, a Research Center focused on two of the main activities supporting the urban growth: logistics and enterprises. He is also the Director of the ICE Lab. He is also a member of scientific boards and awards, including the Scientific Board of SOS Log, the main Italian association of Sustainable Logistics, and the Amazon Innovation Award, the International Award of Amazon on Last Mile, and Logistics and Emerging Business Models.

ALESSANDRO RIZZO (Senior Member, IEEE) received the Laurea degree *(summa cum laude)* in computer engineering and the Ph.D. degree in automation and electronics engineering from the University of Catania, Italy, in 1996, and 2000, respectively. In 1998, he was a EURATOM Research Fellow with JET Joint Undertaking, Abingdon, U.K., researching on sensor validation and fault diagnosis for nuclear fusion experiments. In 2000 and 2001, he was a Research Consultant to ST Microelectronics, Catania Site, Italy, and as an Industry Professor of robotics with the University of Messina, Italy. From 2002 to 2015, he was a tenured Assistant Professor with the Politecnico di Bari, Italy. In November 2015, he joined Politecnico di Torino, Italy, as a tenured Associate Professor. Since 2012, he has also been a Visiting Professor with the New York University Tandon School of Engineering, Brooklyn, NY, USA, where he taught several graduate and undergraduate courses and carries out research on underwater robotics and complex networks and systems. He is involved in conducting and supervising research on cooperative robotics, complex networks and systems, and modeling and control of nonlinear systems. He is the author of two books, two international patents, and more than 150 articles on international journals and conference proceedings. Prof. Rizzo was a recipient of the award for the best application paper at the IFAC World Triennial Conference in 2002 and of the award for the most read papers in *Mathematics and Computers in Simulation* (Elsevier) in 2009. He is also a Distinguished Lecturer of the IEEE Nuclear and Plasma Science Society.