A Survey on Local Transport Mode Detection on the Edge

Mahdieh Kamalian1* and Paulo Ferreira1

Abstract

We present a survey of solutions for smartphone-based transport mode detection. These are divided into local and remote approaches being the first ones addressed in this article. A local approach performs the following steps in the smartphone (and not in some far away cloud server): 1) data collection or sensing, 2) preprocessing, 3) feature extraction, and 4) classification (with a previous machine learning based training phase). Solutions are presented taking into account the above mentioned four steps, and analyzed according to the most relevant requirements (accuracy, delay taken to detect a transport mode, resources consumption, and generalization).

Keywords: transport mode detection; accuracy; delay; resource consumption; classifier generalization; smartphone

1 Introduction

The evolution of mobile network technologies and smartphones have provided mobile users with ubiquitous access to the Internet and its value-added services on the move [1]. Besides making phone calls, nearly all smartphones today provide users with a varied range of built-in sensors and applications which have considerably influenced human lifestyle. Today, more or less everybody uses their smartphones for navigating their path, for finding the best restaurant or other entertainment facilities close by, for well-being purposes, etc. All these functionalities are integrated into one, rectangular and revolutionary technology that keeps getting smarter and smarter every day [2]. The advancements in sensing, computing and storage capabilities of smartphones have made them an effective tool for monitoring travel behavior of mobile users.

The knowledge of travelers’ behavior has a considerable impact on different research areas such as transport, urban planning, health, epidemiology, and entertainment [3, 4]. Monitoring individuals’ travel activity can help to improve urban transportation planning, design and management. Such travel information becomes even more important for traffic management when a fast response is required for the allocation of one specific mode of public transportation. For example, a football match, a national festival or a bad weather condition may change the regular public transportation demands in a specific period of time [5]. Furthermore, monitoring patients travel behaviour may contribute to control the spread of a disease during a pandemic. Likewise, it is a useful way of detecting abnormal activities of patients with Alzheimer, dementia or any other mental pathologies. Hence, patients can be protected from danger and undesirable consequences [6]. In environmental
epidemiology, transport mode detection (TMD) is a great tool for studies which are concerned with the influence of travel modes on air pollution exposure [7]. When done in real time, detection of the transport modes can be used, for example, as an automatic way for issuing tickets for the travellers, particularly when they are using public transport means such as train, bus, subway, etc.

Some of previous studies relied on travel logs and some other on dedicated wearable sensors for TMD. However, such systems put the burden on research participants and users. Therefore, there has been a shift toward using smartphones for TMD in recent years [8]. To obtain the knowledge of travelers’ behavior we need to gather some sensing information and train a machine learning (ML) algorithm which can predict transport modes.

1.1 Transport Mode Detection (TMD)
When the TMD is ML-based, it has the following main steps: 1) data collection or sensing, 2) preprocessing, 3) feature extraction, and 4) classification with a previous training phase. The advancements in sensing capabilities of smartphones and their resource improvements (i.e., faster CPU, more memory) provided us with the opportunity of collecting data-sets by using smartphones sensors (first step), and also to perform the above mentioned other three main steps (i.e., except the training which might require more computing power) locally on the smartphone. Therefore, based on the steps which are performed locally on a smartphone or on a remote server, TMD approaches can be categorized as being local or remote [8].

Performing classification locally on a smartphone has some advantages over a remote approach using some (cloud-based) server as follows. 1) The smaller amount of data exchange between a remote server and smartphones (i.e., data transmission occurs only in case that further analysis, evaluation of the classified modes, assisted feedback or long-term storage is required); 2) There is no need for transmitting raw collected data through the (possibly slow) network between a smartphone and a server; so, no latency is imposed and real time (i.e., quick enough, based on human time scale) TMD becomes possible; 3) There is no need for continuous connectivity to the Internet in order to send the raw data of each trip to a server and receive the result back for every classification. In other words, a user’s smartphone is not required to be connected to the Internet during the classification process; the reason being that in a local approach users are not dependent (for TMD purposes) on a remote server which is only accessible through the Internet; 4) Less user specific data has to be sent to a server with obvious privacy advantages; 5) The accuracy of TMD with a local approach can be the same as with a remote approach. For instance, Guvensan et al. [4] proposed a local approach which achieved 94.5% accuracy thanks to their post-processing algorithm (i.e, healing algorithm) which is equal to the average accuracy (i.e., 94.5%) of a remote approach presented in Chen’s work [3]; 6) The growing increase in computing, storage and power capabilities of smartphones makes them an encouraging platform for running ML algorithms and storing data locally rather than on a remote cloud server.

Local classification gives us the opportunity for real-time (as previously indicated) TMD by omitting the latency that transmission to a server and back imposes. Real-time TMD allows context-awareness for location-based services (LBS) to provide
customized information delivery based on a user’s needs and her/his interactions. For example, the short latency allows a TMD application to alert a user when she/he should start walking towards the bus station in order to be on time for work [9], or to inform a user about the real time location of relevant buses [10].

1.2 TMD Requirements

Different kinds of TMD approaches made an effort to meet (at least) one or a combination of four principal requirements (see Section 5 for more details): high accuracy, delay considerations, low resource consumption, and high generalization.

Accuracy is the fundamental requirement; it expresses the number of correct predictions of the classification model about transport modes of users.

Delay in TMD consists of two different parts: computing-time (i.e., the time taken for the classification done by a ML algorithm), and latency (i.e., the time taken to send the raw collected data to a server and receive the results back). To minimize latency, local classification is preferable (to remote classification) and some studies proposed a solution to reduce the dimensionality of data (i.e., simplify the data by using two dimension reduction techniques such as principal component analysis (PCA) and recursive feature elimination (RFE)) in order to minimize the computing-time and achieve near real-time detection of transport modes [11].

Resource consumption accounts for the amount of resources in terms of battery, CPU, memory (RAM), etc. that the classifier takes up while running on a smartphone. To the best of our knowledge, there is a limited number of studies which have evaluated the battery, CPU or memory usage of their solution running on smartphones [10, 12, 13].

A generalized system is one in which its accuracy is not affected by smartphone position changes, user variation (i.e., collecting data from a variety of users with different age, gender, education, job status, etc.) or the geographical location changes.

1.3 Goal

The goal of this article is to provide an overview of: i) most common sensors used in local TMD and exploring their capability regarding detecting different modes; ii) most useful metrics for evaluating the local ML-based TMD systems; iii) most common ML algorithms and the best choices for achieving high accuracy; iv) common steps of local TMD; and v) how local TMD requirements are fulfilled and affected mostly by which step. Note that we focus on local TMD (and not on remote) given its interesting capabilities regarding real-time detection, lower users’ concerns about their privacy, and the fast pace of progress in smartphones’ capabilities.

Thus, a major difference of this survey when compared to others [14, 15, 9, 16] is twofold: we mainly focus on local TMD approaches, and we introduce a novel perspective of TMD applications by considering the four main requirements of such approaches (that were mentioned above): high accuracy, delay considerations, low resource consumption, and high generalization. To the best of our knowledge, there is no review work which focus on local TMD while taking into account the above mentioned requirements.

In this survey, we use the following criteria for the selection of studies and applications which have been reviewed (see Section 7 for more details on the methodology that was used):
• They use built-in smartphone sensors for the data collection step. Those TMD solutions which use sound samples gathered by the mobile phone microphone or pictures captured by its camera (such as Miluzzo’s work [17]) are not included in this survey. The reason is that, these types of data gathering raise a lot of privacy concerns for users.

• They have implemented TMD main steps locally on smartphones. The training can still be done on a remote server or on a smartphone.

• They are able to recognize more than two types of transport modes. For example, studies or applications which are focused only on bicycles or cars are excluded.

• Although there are some human activity recognition (HAR) approaches which have implemented activity recognition locally on smartphones [18, 19], they are not part of this survey because of the following reasons: 1) they have covered activities such as walking downstairs or upstairs, standing, sitting, etc. which are not our goal classes (goal classes for most TMD applications include walking, running, car, bus, train, tram, cycling, subway), and 2) there is another study which have already evaluated them [8].

There are some other studies [3, 20] which reported that they have proposed a real-time TMD system. However, we did not find any evaluation or proof or any other details regarding a local TMD implementation. Therefore, we do not include such studies in this article.

The rest of this survey is structured as follows. We start by presenting the main reasons for having, and advantages of, local over remote TMD. Then, in Section 3 we describe the smartphone sensors which have been utilized in recent local TMD approaches (e.g., accelerometer, etc.) and their limitations. In Section 4, we describe in detail the four principal TMD steps and the different methods which can be utilized in each step. Section 5 is dedicated to the main four requirements of local TMD systems. In Section 6, different local TMD approaches are compared following the same presentation structure used in Section 4; in addition, a description is provided on how different studies have fulfilled the TMD requirements already mentioned. Section 7 describes our searching strategy. Finally, Section 8 presents the conclusion of this survey.

2 Local vs Remote Smartphone-based TMD

The variety of the built-in sensors available in smartphones, the fast growing enhancements of their processing and storage capabilities, the constant decreasing in their hardware costs, combined with their ubiquitous characteristics have matured smartphones to devices efficiently equipped for identification of an individuals’ transport mode. Hence, smartphones provide the opportunity not only to collect data-sets by the use of their built-in sensors, but also to perform the other three main steps locally (mentioned above in Section 1.1) on a smartphone (note that the training phase can be done locally or on a remote server). Thus, a local approach brings some considerable benefits over a remote approach:

1 Smaller data size. In local TMD, a smaller amount of data has to travel between the smartphone and a server. In fact, data must be sent to a server in two cases: i) when we want to send the training data-set to an ML algorithm
running in the server (which is normally performed once for most of TMD approaches), and ii) when the classification has been performed and we want to analyse the results centrally (and for long-term storage purposes); such data is obviously smaller when compared with the data that would have to be sent from a smartphone to a server regarding every single sensing information of each user (as it would happen in a remote TMD approach).

2 **Less delay.** With local TMD, near real-time detection of transport modes becomes possible. The reason lies in the fact that, in a local approach, the network latency between a smartphone and a server is omitted. In fact, there is no need to send the raw collected data to a remote server through the network and wait for the classification; data transmission occurs only in case that further analysis, evaluation of the classified modes, assisted feedback or long-term storage is required. So, with the local approach, less data needs to be transmitted to provide the users themselves, or others who may be interested in knowing users transport mode (e.g., urban city planners). Therefore, the classification step is not affected by the delay of the transmission itself. Moreover, even with no Internet access, the transport mode can still be identified with a local approach (which is not the case with a remote solution). Therefore, TMD is not delayed because of lack of Internet access.

3 **No need for Internet connectivity.** In a remote approach, Internet connectivity for each mode detection is necessary. However, in a local approach there is no limitation regarding continued connectivity to the Internet.

4 **Improved privacy.** In a local approach, smaller and less personal information of users is sent to a server. This feature alleviates the privacy concerns of TMD application users. For example, users location obtained via the Global Position System (GPS) is not required to be sent to a server for classification purposes.

5 **Better Accuracy.** The accuracy of TMD in a local approach is almost the same as with a remote approach. In fact, such accuracy mainly depends on the sensitivity of the sensors, their combination, the application internals, and on the proposed algorithm or ML algorithm being executed on a smartphone (assuming that it has been trained with sufficient and correct data).

6 **Evolving smartphones.** A fundamental reason to use smartphones to run a TMD application is the constant and fast pace of smartphones advancements mostly regarding sensors quality, computing power, storage and battery capabilities. Therefore, a classification model can be integrated into a mobile application and users’ data can be stored on their own smartphones locally (instead of a remote server). This is in line with the current trends of edge and fog computing [21].

### 3 Available Smartphone Sensors

TMD can be considered as a sub-field of HAR which has been widely studied during the last years. Most approaches relied on dedicated wearable motion sensors for activity recognition. However, there has been a shift toward smartphones for collecting sensing data given their ubiquity and always increasing capabilities. In fact, most current smartphones have various types of sensors which can measure orientation, motion and environmental conditions [12, 22].
In this section, we overview the most relevant sensors in smartphones used by different local TMD approaches. The comparison of different TMD solutions (see Section 6) suggests that accelerometer and GPS are the mostly used sensors for data collection. Despite the fact that using more sensors can provide higher accuracy, it is more common for researchers to carefully use a reduced number of sensors to restrict the energy consumption. In fact, a hybrid approach that collects the data using several sensors has achieved higher accuracy in comparison with approaches which only used one single sensor.

- An **Accelerometer** is the most commonly used sensor for TMD [11, 10, 12, 4, 13, 18, 23]. It is is an electromechanical device that is able to measure the force of acceleration caused by some movement or gravity on all three physical axis. These forces can be static (e.g., gravity), or dynamic (e.g., movements or vibrations). Accelerometers are mainly used for orientation sensing in smartphones. The ability of distinguishing between different types of transportation by detecting the changes in the acceleration and deacceleration patterns of vehicles, and its low battery consumption, has made this sensor very attractive for TMD systems. Accelerometer uses about 10 times less battery than the other motion sensors, particularly on Android [24, 25]. In spite of the wide range use of accelerometer for TMD applications, the similarity of acceleration patterns of some vehicles such as cars and buses (or subways and trains) has limited its ability for distinguishing all modes correctly [3]. For instance, single accelerometer solutions are not able to differentiate between stationary, cycling and motorized vehicles as accurately as when using the GPS [10] in combination.

Accelerometer data is very useful when it comes to the detection of changes in motions during an activity. For instance, stationary activities have a very low variance; motorized transports exhibit similar characteristics as the stationary mode in spite that, motorized transports are affected by road conditions and vehicle vibrations; walking and cycling show similar acceleration and deceleration motions in certain body areas; running has a large variance in comparison with other modes [10].

- A **GPS** sensor is commonly used for recording a person’s location (speed can then be obtained rather easily). Built-in GPS sensors are provided by most current smartphones. They are able to locate a user’s position based on the distance of her/his device to four or more visible satellites. Then, by locating the user and calculating the time taken to move from source to destination, the user’s speed is also known. However, GPS sensor brings about some limitations in a TMD context. First, it has been shown that a GPS sensor consumes a considerable amount of battery and, as a result, smartphone battery last a few hours with continuously logging [3]. Second, a GPS sensor is unable to work properly in cases that a user is travelling underground, in a tunnel, along urban canyons, or anywhere else where the GPS signals get blocked or jammed. Specifically, accurate GPS localization requires an unobstructed view of at least four satellites and the precision increases with more visible satellites. Third, current GPS-based only TMD approaches (i.e., without the help of other sensors) can only detect fine-grained motorised transportation
modes with modest accuracy [12, 26]. Using GPS along with other sensors such as accelerometer, gyroscope and magnetometer improves the accuracy but not significantly [26].

- A **Gyroscope** is able to measure a smartphone’s rate of rotation around all three physical axes. Gyroscopes are categorised into three different groups: mechanical, optical, and micro-electro-mechanical systems (MEMS). MEMS gyroscopes are predominately used within mobile devices (which we call simply gyroscopes in the rest of this article) [27]. The gyroscope complements the built-in accelerometer to understand which way a device is orientated, adding another level of precision. An accelerometer measures the linear acceleration or directional movement of the smartphone, whereas the gyroscope sensor measures the angular velocity, tilt and lateral orientation of the smartphone. Although gyroscope sensor energy consumption is much lower than GPS, its power usage is significantly higher in comparison with the accelerometer [15]. Furthermore, some metrics such as offset error, shock and vibration sensitivity, temperature sensitivity, etc. can affect gyroscope’s quality and calibration [27]. Thus, single gyroscope solutions are not efficient for TMD.

- A **Magnetometer** sensor on a smartphone uses modern solid state technology to create a miniature Hall-effect sensor that detects the Earth’s magnetic field along the three physical axes [28]. The magnetometer is crucial for detecting the orientation of a device relative to the Earth’s magnetic north. Some approaches have exploited this feature to detect motorized vehicles in combination with other sensors (i.e., accelerometer) [3]. Comparisons of battery consumption between different sensors suggests that a magnetometer uses about twice the energy than a gyroscope sensor [15]. Another relevant aspect is that the information such a sensor can provide for differentiating between motorized transports and walking or running is not sufficient for itself.

- The **GSM** (Global System for Mobile Communications) sensor can be used to collect and detect a smartphone position measuring the signal strength of a device and the fluctuation pattern of cell identifiers. A cell is a geographic region within which mobile devices communicate with a particular GSM base station (i.e., a GSM base station define sectors of coverage for mobile phones); each cell has a unique cell identifier. The GSM signal fluctuation pattern together with its strength can provide the information on a mobile phone’s position. For instance, Sohn et al. [29] follows the changes in GSM cell tower observations (up to seven) in order to detect if the user is still, walking or in a motorized vehicle. However, single GSM sensor solutions such as developed in Sohn’s work are not useful for fine-grained classification. Also, this sensor is dependent on network endpoint density which varies based on the users’ environment [10]. Moreover, ping-pong phenomenon is another limitation of the usage of GSM sensors. Ping-pong phenomenon happens when a user is within the coverage of two or more cell towers; because of cell channel fluctuation, signal strength from the towers fluctuates. As a result of such repetitive changes, the system assumes that the user is moving but she/he is in fact stationary.

- **WiFi** is a technology that connects smartphones to a network. Based on the coverage of a WiFi access point, a smartphone’s position can be predicted.
However, the position information provided by WiFi is not sufficiently accurate and/or precise for TMD, it is unreliable outside urban areas, and it is the most battery consuming sensor (after GPS) when used to provide just location information (i.e., without communication) [14]. The ping-pong effect (already mentioned above for GSM sensors) is also typical for WiFi data. As a result, WiFi has not been widely used for TMD (see Section 6).

There are some other sensors available in the smartphones such as microphone, light sensor, barometer, Bluetooth, etc. In Wang’s work [30] the barometer along with three other sensors (i.e., accelerometer, gyroscope, magnetometer) is used to train a LSTM model which can detect bus, car, subway and train with 96.9% accuracy.

Bluetooth is also used in Chen and Bierlaire’s work [31] to provide valuable information about the smartphone’s context. For example, the Bluetooth sensor detects other nearby Bluetooth devices (more in a public transport environment than in a private mode). The number of nearby visible Bluetooth devices varies with the context. In public transports, people are more compact in the vehicle, and they are stationary relative to each other. Hence, a smartphone has a higher chance to observe more Bluetooth devices than in private transportation. Therefore, they utilize the information about nearby Bluetooth devices in differentiating public/private transport context.

In another study [32] the signal levels of Bluetooth are used for inferring the information about a user’s environment. Thus, from the number of discoverable Bluetooth networks around a user, it is possible to estimate the number of people around. Also, if other people happen to move with the user, such as when taking the bus or metro, but also when sharing a car with multiple people, then some of these networks will stay in range over the course of the journey.

In addition to the sensors, some studies have exploited the information of external resources such as route maps, public transportation timetables, and network infrastructures (e.g., road and rail maps), etc. [32, 31]. However, these types of external sources are not commonly used among local TMD systems because of the extra complexity and memory required.

To the best of our knowledge local TMD systems have not widely used neither the previously mentioned sensors (microphone, light sensor, barometer, Bluetooth) nor external resources. Therefore, in this survey we do not address them further.

4 TMD Steps
As previously mentioned, most local TMD approaches rely on some ML algorithm, and the detection follows four steps: 1) data collection or sensing, 2) preprocessing, 3) feature extraction, and 4) classification with a previous training phase [8]. A few local TMD approaches have an additional step for post-processing which aims at correcting the miss-classification results of a ML algorithm [4]. In this section we first provide a description of each one of these steps. We conclude this section with details on local vs remote classification and the most important TMD metrics.
4.1 Data Collection
As mentioned in the previous section, in most current local TMD approaches data is collected through one or a set of built-in smartphone sensors such as the accelerometer, GPS, magnetometer, gyroscope, etc. (such sensors were described in Section 3). Sensor selection, position-dependency of the sensor (where the user put his smartphone usually during data collection), number of test users, and test duration are the key factors for data collection which affect the accuracy of the classification. As discussed in Section 6, most local TMD approaches use several sensors to be able to detect fine-grained transport modes accurately.

In most of ML-based TMD studies, when building the classification model, the collected data-set is divided into 3 parts, each used for a different phase of modeling [10]: training, validation, and test set. A training data-set trains the ML model using a supervised learning method (e.g., decision tree). Then, the model that resulted from the training phase is used to predict the responses for the observations in the validation data-set. The validation data-set provides an unbiased evaluation of a model fit on the training data-set while tuning the model’s hyper-parameters. Finally, the test data-set is used to evaluate the final classification model [33]. The test data-set can also be known as a holdout data-set in case that it has never been used during the training phase.

4.2 Preprocessing
In this step, raw collected data is processed in various ways; e.g., performing data cleaning or removing user errors within the data. In all ML-based approaches a training data-set is collected with the help of some participants; these participants label the transport modes used after every trip so that a true basis is built. Such participants use a collecting application on their smartphones and sometimes, in addition to labeling, they have to record the start and the end of the trip. In these cases there is a possibility for users to (mistakenly) insert wrong labeling and recording. Therefore, it is essential to remove such trips from the data-set before training the ML algorithm. Moreover, it is possible for the data-set to have some noise or systematic errors particularly, in raw GPS data. In fact, the smartphone mobility introduces noises when it moves close to the human body or when it is placed at different positions. In order to provide an accurate classification it is pivotal to use some techniques such as data filtering (i.e., removing the data that does not represent user real positions) and smoothing (to help reducing random noise present in the data) to ensure better accuracy [34].

4.3 Feature Extraction
Raw data collected by different sensors is typically segmented into frames, using a sliding window, each frame being processed, and features extracted from such a frame. The extracted features are used for learning and classification tasks [14].

The size of the sliding window can affect the classification accuracy, response time, and memory size [15]. On one hand, a small window size decreases the classification accuracy due to certain features not being effective (e.g., accelerometer frequencies); on another hand, a large window size may introduce new source of noise in the data. Moreover, both window size and sampling rate are very important parameters which influence computation and power consumption of sensing algorithms.
Most local TMD studies rely on time and frequency domain features to transform sensing information into lower dimensional sets of features [15]:

- **Time-domain features** are basic statistical data such as mean and variance, based on a frame of samples. Time-domain features express envelope metrics of the frame through features such as minimum, maximum, mean and median of the data frame [27, 15].

- **Frequency-domain features** are based e.g., on the fast Fourier transformation (FFT) or wavelet decomposition of a frame of samples. Frequency domain features can quantify periodic patterns within the signal, which are typically generated by repetitive physical movements such as walking or cycling [27].

4.4 Classification and Training

In this section we provide an overview of the classification step along with details regarding the training phase. We address the two categories for classifiers (discriminative and generative), the different approaches for performing classification (locally or remotely) and the various useful metrics for evaluating the classifiers. Moreover, in this section, we detail the two different methods of training a TMD model: local and remote training.

Classification is a supervised learning approach in which a classification model is trained, based on a collected data-set; this learning is then used to classify a new observation. The data-set may simply be bi-class (i.e., assigning a given email to the "spam" or "non-spam") or it may be multi-class such as several possible transport modes used by a mobile user.

The main objective of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features [35]. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown.\(^1\)

The classifiers commonly used for TMD can be categorised in two different groups: generative, and discriminative. In general, a generative model explicitly models the actual distribution of each class, while a discriminative model, models the decision boundary between the classes. Both models are predicting the conditional probability \(p(y|x)\), and learn different probabilities. Generative classifiers learn a model of the joint probability, \(p(x,y)\), of the inputs \(x\) and the class label \(y\), and make their predictions by using Bayes rules to calculate \(p(y|x)\), and then pick the most likely label \(y\). Discriminative classifiers model the posterior probability \(p(y|x)\) directly, or learn a direct map from inputs \(x\) to the class labels [36].\(^2\).

\(^1\)In ML, observations are often known as instances, the explanatory variables are termed predictor features, and the possible categories to be predicted are defined with class labels; e.g., for detecting transport modes, trips are instances which based on the extracted features (e.g., average, maximum, minimum and standard deviation of acceleration) are categorized into one of the class labels (e.g., walking, running, biking, car, bus, train, metro, etc.).

\(^2\)Discriminative classifiers do not attempt to model underlying probability distributions. Instead, they estimate parameters of posterior probability directly from training data [14, 37]. Posterior probability is the probability an event will happen after all evidence has been taken into account [38].
Table 1 Common implemented classifiers on smartphones for local TMD

| Classifiers               | Category  | Relevant studies | Total studies |
|--------------------------|-----------|-----------------|---------------|
| Decision Tree (DT)       | discriminative | [10, 12, 13, 34, 24] | 5             |
| Random Forest (RF)       | discriminative | [13, 11, 4, 34, 24] | 5             |
| (K-)Nearest Neighbours (KNN) | discriminative | [10, 13, 11, 4, 24] | 5             |
| Support Vector Machines (SVM) | discriminative | [10, 34, 39, 24] | 4             |
| Naive Bayes (NB)         | generative | [10, 4, 24] | 3             |
| Adaptive Boosting (AB)   | discriminative | [12, 24] | 2             |
| Neural Networks (NN)     | discriminative | [24, 23] | 2             |
| Hidden Markov Model (HMM) | generative | [10, 12] | 2             |
| Bayesian Networks (BN)   | generative | [39, 24] | 2             |

Table 1 presents common classifiers and their corresponding categories, for local TMD. The algorithms considered are those which have been used at least for two different studies. As it is shown in the table, discriminative classifiers have been used more frequently for local TMD due to their lower computational cost (thus being suitable for running on a smartphone) [14]. Furthermore, the total studies represents the number of local TMD works which have evaluated each algorithm.

As already mentioned, based on the steps described in the previous section which can be performed either locally (on a smartphone) or remotely (on a server), TMD approaches can also be categorized as being local or remote [8]:

- **Local approach.** The TMD steps are done locally on the smartphone. These steps include sensing, preprocessing, feature extraction and classification (note that the training phase can still be done on a remote server or locally on the smartphone). Obviously, with this approach, some information regarding raw data and the automatically detected transport mode can still be sent to a server either for further analysis or for long-term storage; note that this would happen opportunistically and the local approach is not required to have network connection to work. In fact, the main four steps already mentioned are done on the smartphone. Therefore, even if the generated classifier has been trained on a server beforehand, it must be adapted to resource constrained devices such as smartphones (i.e., less resources when compared to a server) where it runs detecting the transport modes of travellers.

- **Remote approach.** The data collection is done on a smartphone and the raw data is sent to a server for preprocessing, feature extraction and classification. The preprocessing step can be partially done on the smartphone but the classification algorithms run on a server. As mentioned in Section 2, remote classification leads to longer delays when compared to a local approach, given that raw collected data travels through the (possibly slow) network from a smartphone to a remote server. Moreover, the requirement of the Internet access imposes more delay to the classification step.

There are various classifiers that have been developed to run locally on smartphones in the last few years. The most commonly-used classifiers in local approaches (see Table 1) are decision tree (DT), random forest (RF), support vector machine (SVM), and K-nearest neighbour (KNN).

In some studies, two classifiers are combined in different ways, thereby creating a hierarchical or multi-layer classification system [3, 10]. For instance, in Reddy’s work [10], a decision tree followed by a dynamic hidden Markov model (DHMM) has been proposed as a two-staged classification system.
Most of previous work carried out an evaluation between different ML algorithms to determine the most accurate classification model (or the combination of them) for TMD. Performance evaluation of the resulting classifier is commonly measured through four different metrics: accuracy, precision, recall and F-measure (all detailed below). The most standard metric for a classifier’s evaluation is based on prediction accuracy [35]. The other metrics are useful to provide an overall insight about a classifier performance. Therefore, in this document we first introduce all the metrics, and then emphasize on the accuracy of TMD approaches for a comparison.

The classification results for a particular TMD system can be organized in a confusion matrix $M_{n \times n}$ for a classification problem with $n$ classes. This is a matrix in which the element $M_{ij}$ is the number of instances from class $i$ that were actually classified as class $j$. Thus, for a binary classification problem, the following values can be obtained from the confusion matrix:

- **True Positives (TP)** - The number of positive instances that were classified as positive.
- **True Negatives (TN)** - The number of negative instances that were classified as negative.
- **False Positives (FP)** - The number of negative instances that were classified as positive.
- **False Negatives (FN)** - The number of positive instances that were classified as negative.

The values above can be generalized for a problem with $n$ classes such as it happens with TMD. In such a case, an instance could be positive or negative according to a particular class, e.g., positives might be all instances of walk while negatives would be all instances other than walk.

As mentioned above, performance evaluation of the resulting classifier is commonly measured through four different metrics which we now present in detail:

- **Accuracy** - indicates how correct or wrong the classifier prediction is (see Equation 1). The accuracy is the most standard metric; it summarizes the overall classification performance for all classes. So, accuracy can tell us immediately whether a TMD model is being trained correctly and how it may perform generally.

$$\text{Accuracy} = \frac{TP + TN}{\text{TotalPredictions}} \tag{1}$$

- **Precision** - indicates the number of instances correctly classified as positive out of the total instances classified as positive (see Equation 2). Precision is a good measure to determine, when the costs of false positives is high. For instance, in email spam detection, a false positive means that an email that in fact is non-spam has been identified as spam (predicted spam). Therefore, the user might lose important emails if the precision is not high for the spam detection model. The precision, often referred to as positive predictive value, is the ratio of correctly classified positive instances to the total number of instances classified as positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$
Recall - indicates the number of items correctly classified as positive out of the total actual positives (see Equation 3).

Applying the same understanding, recall is a good measure to determine when the costs of false negatives is high. For instance, in sick patient detection, if a sick patient (actual positive) goes through the test and is predicted as not being sick (predicted negative). The cost associated with a false negative will be extremely high if the sickness is contagious. The recall, also called true positive rate, is the ratio of correctly classified positive instances to the total number of positive instances.

\[
Recall = \frac{TP}{TP + FN} \quad (3)
\]

F-measure - is useful when we need to seek a balance between precision and recall and there is an uneven class distribution. The F-measure combines precision and recall in a single value (see Equation 4).

\[
F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)
\]

As already mentioned, all the above metrics are defined for a binary classification but they can be easily generalized for a problem with \(n\) classes such as the TMD problem. Note that the importance of each of the above mentioned metrics depends of the TMD use case. For instance, assume that the municipality of a city uses a TMD system for measuring the total amount of CO2 emission by cars. The municipality is responsible to warn the respiratory patients to stay at home when the amount of CO2 pollution is higher than a specific threshold. In this situation, recall is a very important metric. The reason being that detecting cars mistakenly as other modes (FN) leads to measuring less CO2 emission and it can put respiratory patients in danger. Also, assume that the municipality defines a fine for car drivers in the most polluted days. In this use case, precision is more important than recall. The reason is that, fining the individuals who did not actually drive a car (FP) is against the law.

In order to classify test data into various pre-defined transport mode classes such as walk, run, bicycle, car, train, tram, metro, bus, etc., the classifier needs to be initially fitted (trained) with a training data-set. Specifically, in the context of mobile phones and with local TMD, the training phase can be done in two different ways known as local or remote:

- **Local training** means that the training of the classifier is done on the smartphone. To the best of our knowledge, this method has not been used by local TMD approaches due to the computation and battery limitations that smartphones represent.

- **Remote training** means that the training of the classifier is not done on the smartphone; instead, it is usually done on a server or on any type of computer which is capable of running the corresponding algorithm. Most studies have used the remote method since computationally it is cheaper and more feasible in the real world scenarios [13, 11].
5 TMD Requirements

In this section we introduce the main requirements of TMD systems and the related influencing factors. Different kinds of TMD approaches made an effort to meet (at least) one or a combination of the requirements already mentioned: high accuracy, delay considerations, resource consumption analysis, and generalization. In the following, we provide a detailed description of these requirements and various techniques used by TMD systems to fulfill them.

5.1 Accuracy

Achieving the highest accuracy is a fundamental requirement and goal of every TMD system. As already mentioned (see previous section) accuracy is a metric that states how many correct predictions the classification model has made (i.e., the percentage of correct predictions divided by the total number of predictions). Obviously, having the TMD with 100% accuracy is the ideal goal; however, note that there may be some systems which can still perform perfectly with a lower level of accuracy as there is always a low boundary of acceptable accuracy depending on the context. As an example, when a TMD application is used to provide a personalized scorecard for tracking environmental impacts of one’s activity, the accuracy should be more than 90% [10]. Another example is that of a company is using a TMD application to count the number of people using their bicycles instead of their own car or public transportation; in this case, the accuracy should be 100%. The assessment of previously proposed local TMD systems suggests that none of them have been able to achieve the highest accuracy (i.e., 100% accuracy) for detecting all modes (i.e., fine-grain transport mode detection) [12, 11, 4]. These systems are detailed in Section 6.

A critical factor that directly affects the accuracy in a supervised classification is the choice of the specific ML algorithm which is used to train the classification model. There are at least three techniques which are used to evaluate a classifier’s accuracy. One technique is to split the training data-set by using two-thirds for training and the other third for estimating performance. In another technique, which is known as cross-validation, the training data-set is divided into equal-sized subsets, and for each subset the classifier is trained on the union of all the other subsets. Therefore, the error rate of the classifier can be estimated based on the average of the error rate of each subset. Leave-one-out validation is a special case of cross validation; all validation subsets (i.e., on which the model is evaluated) consist of a single instance. Although this type of validation is more expensive computationally, it is useful when the most accurate estimate of a classifier’s error rate is required [35].

Another key factor that affects the accuracy of a classification model is the amount of training data. Having more data means that the ML algorithm has more information to understand the various situations and correlate them before giving the right answer. In other words, by providing the ML algorithm with a variety of data that can cover wide-ranging scenarios, it is possible to avoid biased decisions. Hence, the more the classification model is fed with various data the more its accuracy will improve.

In practice, it is always recommended to try and compare different algorithm’s accuracy, in order to choose the most appropriate solution. Such accuracy is influenced by the collected data and extracted features. Therefore, the number of sensors
and their quality (i.e., typically more recent and/or expensive smartphones provide a better quality of sensors than the older ones), number of test users, test duration, number of features and samples, window size, the different classes and whether they are linearly separable or not, are the contributing factors for more accurate TMD.

5.2 Delay

One of the main requirements of TMD systems is to provide users with real-time responsiveness. This means that TMD systems must be quick enough, at human time scale, in detecting the transport mode used. In other words, delay considerations makes the real time transport mode detection possible. Real-time TMD offers context-awareness for location-based services (LBS) making it possible to provide customized information delivery based on user’s needs and her/his interactions. Where a user expects to be provided with some assisted feedback, lower delay is a very important factor in addition to accuracy.

For example, the TMD application should alert a user when she/he should start walking towards the bus station in order to be on time for work [9], or to inform the user about the real time location of the relevant buses [10]); so, in such cases, delay in detection is intolerable, i.e., if the TMD application does not detect the transport mode used in real time, it is not able to send the user an in-time alert regarding the next bus. In order to achieve such high responsiveness (i.e., real-time capability) a TMD system requires minimizing the delay of the TMD application.

Delay in TMD systems has two components: 1) computing-time, and 2) transmission-time or (network) latency. Computing time is the time spent by the proposed ML algorithm (running locally) to classify different transport modes taking into account their pre-defined classes. It is clear that a complex ML algorithm and too many features may cause a longer delay in the prediction time. Transmission time or latency is the time required to send data (collected) from a smartphone to a server (which would host the transport mode classifier in a remote approach) and receive the results (i.e, such as "the user was in a car" or "the user was in a train") back from the server. It is clear that with a local approach, latency is minimized. The reason is that, with local classification the delay which is imposed to the application as a result of sending/receiving data through the (possibly slow) network is omitted. Moreover, it is not necessary for the application to be connected to the Internet for detecting the transport modes and present the results. However, most of the existing remote solutions for TMD, which rely on a server (e.g., on a cloud) to run a ML algorithm and then provide the results, do impose some latency when compared to a local approach.

There is another type of delay which can affect the overall time taken for TMD; we call it modal change time. Modal change time has been considered in few studies, particularly in Hemminki’s work [12] in which this type of delay is named latency. However, it has a different meaning from what we referred previously. In Hemminki’e study the latency of detecting changes in the transportation mode has been introduced [12] meaning the delay in detecting the correct modality when the transport mode has changed.
5.3 Resource Consumption

Resource consumption accounts for the amount of resources in terms of CPU, memory (RAM), and battery that the classifier takes up while running on a device (i.e., a smartphone). Such analysis must be performed in order to validate if the proposed local TMD system can be run in real-world settings, and to measure how much it impacts the performance of other applications that a mobile phone may run simultaneously.

5.3.1 CPU and Memory Usage

CPU and memory consumption is an important aspect of TMD systems. This is one of the factors that allows shifting from remote classification to local classification. The constant fast pace of advancements in smartphones capabilities regarding their computing and storage power, besides the decreasing trend in their prices, allows TMD developers to use smartphones for local classification (and local training). To the best of our knowledge, there is a limited number of studies which have evaluated the CPU and/or memory usage of their solution running on smartphones [10, 13]. However, it is a vital parameter to evaluate the TMD application performance.

5.3.2 Battery Usage

Most users expect a TMD application without a significant impact on their smartphone battery not imposing the need for a recharge every 24h. That is why one of the main challenges of a smartphone-based TMD application which uses a smartphone (along with its sensors and local classification) is the battery usage minimization.

There are different factors that influence the battery usage in a TMD process. The type and the number of sensors (which are used for the data collection step), the window size and the sampling rate, are the most important factors which can define the amount of energy that a TMD application uses. To minimize battery consumption, different parameters should be taken into account, such as the following: the number of sensors used for data collection, the type of sensors (e.g., GPS is a high power consuming sensor), amount and dimensionality of collected data, the sampling rate (i.e., the frequency at which a TMD application senses and records data), and the computing power used for classification. The sensor selection is one of the most important stages for a TMD system. Sensors should be chosen in a way that the highest accuracy is promised and the battery usage should not be that much high to influence the user experience during application usage.

5.4 Generalization

A generalized system, in the context of TMD, is a system in which its accuracy does not depend on different parameters such as the smartphone position (i.e., during data collection), user variation or the geographical location. In fact, a generalized system provides a great flexibility regarding the position of the sensing device during data collection (e.g., a smartphone carried in a backpack, in the pocket, etc.); it also shows almost the same detection accuracy regarding new users with different age, education and different travel behaviours (e.g., different walking patterns, different walking or running speed, etc.). Furthermore, such a TMD system should be generalized geographically and not only tested in a specific location and road conditions. Therefore, system generalization includes three major independency aspects: device position, user, and location.
5.4.1 Device Position Independency

As far as a user convenience is concerned, it is of great importance to design and develop a TMD system which accuracy is not dependent of the smartphone placement. Mobile phones are often carried by individuals at different positions. For instance, males often attach their mobile phones to a belt holder or carry them in their pocket; females often put their phones in their bag; and individuals during workouts often attach the phone to the arm or chest area. Instantaneous changes in the body position (i.e., hence smartphone position) may affect the classification and cause low accuracy [4]. Therefore, a TMD classifier is generalized regarding the device position when it is able to detect the transport mode accurately in spite of the smartphone position (i.e., sensor position or direction).

5.4.2 User Independency

Another factor which defines a generalized classifier for TMD is related to user independency. To validate TMD system generalization regarding user independency, there are two different aspects to consider. The first aspect actually discuss how user-friendly the TMD application is; is it easy for different users with different age, gender, education and job status to use the app or not? In fact, new users also should be able to use the proposed TMD application (and enclosed classifier) without the need for user-specific training.

The second aspect concerns how the user variation impacts on the classification accuracy itself. The proposed classifier should detect transport modes with the same accuracy for different users with different walking/running patterns, speed and travel behaviours. To accomplish this goal, the classifier must be trained with a variety of data that covers a wide-range of users. Hence, the classifier should be trained with enough various users’ walking or running patterns, speed and travel behaviour to be able to meet the generalization requirement.

5.4.3 Location Independency

A TMD application running on smartphones should work "everywhere" meaning that the enclosed classifier should be able to detect the transport mode with the same accuracy in different geographic locations with various road and public transportation conditions. To achieve this requirement, data-sets should be collected or the proposed solution should be tested with data from various locations with different road and public transportation conditions.

For example, when the classifier is trained with collected data from Finland and then tested with the data collected from various destinations abroad (e.g., Japan, Germany, Luxembourg) the classifier generalization can be evaluated. For this purpose, the results of the tests with the generalized classifier should be almost the same as the result of the trained classifier. The reason lies in the fact that, when the classifier is trained with different data-sets from different locations it has a wider view for the new observations. Furthermore, as far as the speed limits and standards of various public transportation in different cities are concerned, assessment of the classifier generalization with different locations is necessary.
6 Existing TMD Systems

There are some surveys reviewing the TMD approaches. Nikolic et al. [14] provides a review of TMD approaches based on smartphones but not necessarily for local classification; Biancat et al. [16] reviews the solutions based on both smartphones and other devices (e.g., accelerometer and GPS loggers) without any focus on local classification; Perlipcean et al. [9] presents an analysis of the research for TMD with an emphasis on three different disciplines: location based services (LBS), Transportation Science (TSc) and Human Geography (HG); Wang et al. [15] proposed a standardized and publicly available data-set while suggesting recognition scenarios and sensor combinations. However, this study was tested on only one single type of smartphone (i.e., Huawei Mate 9), thus suffering of not investigating dependent parameters to different type of smartphones such as sensor energy consumption. Huang [40] provides a systematic review of TMD which only concentrates on mobile phone network data which is collected by telecommunication operators. This work is limited to the fact that, access to such a data is not always feasible in all areas and by all operators.

Therefore, in this section we describe the solutions that local TMD works apply to meet the requirements already described in Section 5: high accuracy, delay considerations, resource consumption analysis (i.e., battery, CPU and memory usage), and generalization. We compare local TMD works in terms of their different approaches for each step including data collection, preprocessing and feature extraction, classification and training. Thus, we follow the same structure of the four steps mentioned in Section 4: we present data collection in Section 6.1 and merge the preprocessing and feature extraction (steps 2 and 3) in Section 6.3 because of the strong relation of these two steps that we observed.

When addressing each work within each step, we also consider how each one behaves regarding the requirement that is most affected by that step. Thus, when collecting data (first step, Section 6.2), the generalization requirement is presented given that it is mostly affected by the first step. As far as resource consumption (i.e., battery, CPU and memory) is concerned, we address it in Section 6.4 after data collection, preprocessing and feature extraction steps. The reason being that all these three steps impact the amount of resources used. We have compared the accuracy of different approaches along presenting their classification solution in Section 6.5 since it is a requirement dependent to all steps. Finally, delay is mostly influenced by the last step (classification and training) which is presented in Section 6.6.

6.1 Data Collection

The sensors used, data collection duration, and the number of test users are the key parameters for data collection step of local TMD systems, which influence the classification accuracy.

Moreover, collecting data from a wide range of users with different age, gender, education or job status (referred as users’ attributes) is an important parameter regarding the generalization requirement. This requirement defines a generalized TMD system that is provided for the new users without individual user-specific training (which is also interpreted as friendliness of the TMD application).

Additionally, position independency of the sensor (where the user puts his smartphone usually during data collection) is another important factor which affects the
Table 2 Sensors usage in local TMD systems.

| Studies       | Sensors      | Data collection duration | Number of test users | Users attributes                      | Sensor positions                          |
|---------------|--------------|--------------------------|----------------------|---------------------------------------|------------------------------------------|
| Reddy [10]    | GPS, ACC     | 120h                     | 16                   | gender: 8 males and 8 females, age: 20-45 | arm, waist, chest, hand pocket, bag       |
| Hemminki [12] | ACC          | 150h                     | 16                   | NA                                    | trouser pockets, bag, jacket pockets      |
| Sonderen [13] | ACC, Gy, Mag | 3h                       | 1                    | volunteer                            | leg (same position as front pocket)       |
| Martin [11]   | GPS, ACC, Mag| 96.59h of GPS data, 98.62h of ACC data | 6                    | gender: 2 males and 4 females, age: 18-25, students | pocket, hand bag                         |
| Guvensan [4]  | ACC, Gy, Mag | 79h                      | 8                    | various gender, age: 20-45           | rear/front trouser pocket, jacket pocket  |
| Marra [34]    | GPS          | 1053 days, 4625 days[^a] | 41, 625 (validation) | 39 students, 2 co-authors             | NA                                       |
| Soares [39]   | GPS, WiFi, Cellular | NA                      | 18                   | students                              | NA                                       |
| Liang [24]    | ACC          | 14h (about 2h for 7 modes) | 4                    | users                                 | user’s preferences                       |
| Zhao [23]     | ACC, Gy      | 71.4h[^b], 18h[^c]       | 11                   | volunteers                            | waist                                    |

Note: ACC=Accelerometer; Gy=Gyroscope; Mag=Magnetometer; GPS=Global Positioning System; GSM=Global System for Mobile communication; NA=Not Available.

[^a]: Validation data-set
[^b]: Training set
[^c]: Testing set

A TMD system. Table 2 provides a brief comparison of the sensors used, duration of data collection, the number of test users, users attributes, and sensor positions (during the data collection step) for each local TMD work that we present next.

6.1.1 Reddy

Reddy et al. [10] proposed a classification system that uses both the GPS receiver and the accelerometer to detect transportation modes. The data-set used for training and testing was obtained from 16 individuals, 8 male and 8 female between ages 20-45. The total amount of data (for training and testing purposes) was collected during 120 hours and from 16 users, comprising 1.25 hours of data (15 minutes of data for each of five transportation modes including still, walk, run, bicycle, motorized) per position (six) per individual (sixteen). During the collection, all the users were volunteers who performed the activities in an urban area with six phones attached to different positions of their bodies (i.e., arm, waist, chest, hand, pocket, and in a bag with preferred orientation). As a result, the proposed system in Reddy’s work is position independent.

To validate the user independency requirement, two distinct experiments were performed: 1) user-specific mode - where only a particular user’s data is used for training and testing purposes with 10-fold cross validation (i.e., this test is repeated
for each one of the 16 users individually), and 2) leave-one-user-out mode - where the classifier is trained with all but one user (fifteen out of sixteen) and tested with the user out of the training set.

With the user-specific mode experiment (when training and testing is done on an individual user basis) the accuracy increased only 2.2% compared to the generalized classifier (which is trained and tested on all individuals). With leave-one-user-out mode experiment, an average accuracy of 93.6% and a minimum accuracy of 88.2% was achieved. Overall, the results of these two experiments suggests that the proposed system in Reddy’s work can fulfill the user independency requirement. The reason being that, the increase in the accuracy achieved by the two experiments is minimal in comparison with the generalized classifier (which is trained and tested on all individuals). In addition, the classifier generalization in challenging urban environments is also analyzed in this work. However, given that the data-set is not collected or tested in different locations, Reddy’s solution does not meet the location independency requirement.

6.1.2 Hemminki

Hemminki et al. [12] presented a novel accelerometer-based technique for fine-grained TMD on smartphones. One of the contributions of this work is estimating the gravity component of accelerometer measurements which provides more accurate and robust estimation during motorized transportation.\[3\] To develop and evaluate the proposed approach, over 150 hours of transportation data is collected by 16 individuals from 4 different countries (both for the training phase and for the testing when evaluating it). Three smartphone models (Samsung Nexus S, Galaxy S2 and S3) were used for data collection with 60Hz and 100Hz accelerometer sampling rates.

To ensure the position independency requirement the data-set was collected from different sensor placements. The three most common placements for a smartphone were considered: trouser pockets, bag and jacket pockets. The result of an evaluation using leave-one-placement-out cross-validation indicates the robustness of this study against variations in device positioning. When using leave-one-user-out cross-validation, the small variance of the results demonstrate the robustness of their system across user variations, as well. Finally, generalization capability of the Hemminki’s work regarding location independency is also assessed. The data is collected from various destinations abroad including Japan, Germany, Helsinki, and Finland and the results demonstrate that their approach generalizes well across different geographic locations.

6.1.3 Sonderen

In another work Sonderen et al. [13], a data-set obtained from readings of the accelerometer, the magnetometer, and the gyroscope. The data-set is gathered by the use of an application running on a Samsung Galaxy S4 mini. At first, the

\[3\] A fundamental challenge in accelerometer-based TMD is to distinguish information relevant to movement from other factors that can affect the accelerometer signals. In particular, gravity, user interactions, and other sources of noise can mask the relevant information [12].
data of the accelerometer is used and data from the two other sensors is added in case of insufficiency. To minimize the strain on the phone’s processor and battery, the following items are taken into account: 1) not using the GPS sensor, 2) using the lowest “useful” sampling rate (i.e., 140 Hz for the accelerometer) and 3) using the algorithms with the lowest time and space complexity such as random forest, decision tree, and k-nearest neighbors (see Section 6.5).

The phone is strapped to the leg of a volunteer in the same position as it would be when inside the front pocket. Due to the technical limitations of their application and the phone, the data could not be collected inside the front pocket. Thereby, in Sonderen’s work, device position independency requirement is not met. Moreover, their system is not generalized regarding user independency. The reason is that only 3 hours of data has been collected from a single volunteer. Sonderen also gathered data from a second volunteer in the same way to determine the loss of accuracy when the data of two people is mixed. The results show the clear accuracy loss when more than one person is targeted. They expressed that the accuracy will likely decrease further as more people are added and even the accuracy will drop faster if data is gathered from people with different age, walking pattern and weight.

Using the data of a single person and not considering user variation not only violates the user independency requirement but also introduces an approach which is not applicable in the real world. Moreover, no information regarding user attributes (e.g., age, gender, etc.) of the volunteer is known. Another limitation of this work is that the location in which the data is gathered is not mentioned. So, we do not know whether the classifier is trained and tested in different locations, which results into a non generalized classifier regarding the location independency requirement.

6.1.4 Martin

Martin et al. [11] proposed and compared three different techniques (detailed in Section 6.5) for local classification from the smartphone GPS and accelerometer sensing data. The data is collected by six students (i.e., 2 male and 4 female) aging from 18-25 participating in an undergraduate summer project. This is one of the drawbacks of this work, i.e., they have limited their users to a single-type small group of people (only students). This feature limits the classifier generalization regarding user independency.

The six students collected 96.59 hours of GPS data and 98.62 hours of accelerometer data. The smartphones were carried in ordinary positions such as pockets and handbags, and the linear magnitude of the acceleration is computed to ensure that the classifier is generalized regarding device position changes. The authors tried to meet the location requirement by not feeding their algorithm with mapping information such as bus routes. Hence, their algorithm would be more generalizable by removing the dependence on location-specific data. Although, such an action is helpful for location independency, based on our definition, Martin’s work is unable to fulfill the location independency requirement. The reason lies in the fact that, there is no clear information about the evaluation or test of such a requirement with data coming from a broader variety of locations.
6.1.5 Guvensan

Guvensan et al. [4] introduced a novel multi-tiered architecture which relies on the accelerometer, gyroscope and magnetometer to collect a total of 79 hours data involving 8 participants with various gender and age (from 20 to 45 years old). During the tests, participants carried the phone always on the body (i.e., rear/front trouser pocket and jacket pocket) in different positions. The participants also made their journeys in different modes (such as on foot, sitting, etc.) and in multiple directions including forward, backward and sideways. The variety of users and the different positions in which phones are carried during the tests, results in a generalized classifier in relation to the user and position independency requirements. The results of real life tests demonstrate the robustness of the system against the weight of the vehicle, the driving style, the role in a vehicle (driver or passenger) and the road conditions. Moreover, the final system was also tested with the public data-set provided by HTC [41] which collected transportation mode data in Taiwan. The results of such a test suggests that the system proposed by Guvensan respects location independency requirement.

6.1.6 Marra

In Marra’s work [34] a passive GPS tracking application is proposed; it is claimed that it consumes very little battery power by reducing the GPS sampling rate. However, there is no evaluation of such a claim in Marra’s article. The ML algorithms were designed to use this lower quality of location data to understand all user trips over a period of several weeks. Past travel information from users are also used to identify the missing transfers between two vehicles (i.e., where the algorithm is unable to identify the transfer point and the two vehicles from/to); thus improving the TMD accuracy. This feature (i.e., using individuals’ past travel information) may raise users privacy concerns and is a limitation for the mode detection system when such information is not available.

The smartphone application and algorithms were tested in Zurich. The resulting data-set, referred as Zurich data-set, consists of travel diaries of 39 students and 2 co-authors. In total, 1053 days of travel diaries were collected. Another data-set was collected, referred as validation data-set, with a different smartphone app from 625 users with an average of 7.4 days (4625 days in total) of travel each in the city of Basel; this was used to validate the proposed system. The use of such a data-set for validation, ensures the user independency requirement. Whereas the algorithms are dependent on the public transport operational data (mentioned in Section 6.5) and testing is performed without any data from outside Zurich and Basel, the location independency requirement is not respected; the reason being the fact that, in many cities such information is not available. Finally, in this work there is no information to justify the position independency requirement.

6.1.7 Soares

Soares et al. [39] proposed a real-time TMD application based on location traces using a data mining technique. Their proposed system uses GPS, WiFi and cellular networks data to collect the locations of 18 smartphone users (i.e., 18 students) in the metropolitan area of Rio de Janeiro. A total of 1338 chunks (i.e., set of
traces) and 120176 locations were collected. Collecting data from a limited number of students (a single type of users) makes Soares work a non generalized system regarding user independency. Moreover, no information about the sensor positions while collecting or testing the data is available. Finally, traces for this study were collected mainly in the city of Rio de Janeiro. Traces were also collected in the Duque de Caxias, Nilopolis, and Nova Iguacu regions and in the route between these cities and the downtown of Rio de Janeiro. However, this data does not suffice to respect the location independency requirement.

6.1.8 Liang

Liang work [24] proposes a light-weighted TMD system using only accelerometer sensors of smartphones. This study is not completely clear about being a local TMD approach, but we assume it local based on the information in the conclusion part. In total, 14 hours of data (about 2 hours of data for 7 different modes: stationary, walk, bicycle, car, bus, train, subway) is collected by 4 users. Users can hold their phone freely in any orientation to their preference which makes this system generalized regarding position independency requirement. To minimize the influence of position and rotation, the magnitude value of accelerometer is calculated. However, the small number of participants in this study limits its capability regarding user independency. Users take trains from New Jersey to Washington D.C. to collect data for train mode; collect subway data in New York city; drive cars in local roads and highways in NJIT campus and collect walk and stationary data in campus. Collecting data for each mode in different locations introduces a non generalized approach regarding location independency.

6.1.9 Zhao

Zhao et al. [23] used built-in accelerometer and gyroscope sensors of smartphones for data collection. The data-sets were collected using Huawei P9 and Xiaomi smartphones. The data is collected by 11 volunteers carrying their smartphones on their waists, with the smartphone monitor faced out. Therefore, the Zhao system is not generalized when it comes to position independency. The data of 8 subjects is used for training and the 3 others for testing.

In total, 16755 × 128 × 6 of data is used for training and 4239 × 128 × 6 of data is used for testing. The authors sampled data with 50 Hz and have divided it to windows (i.e., frames) of 128 values; so, each frame is 2.56s long (128÷50). Having 16755 frames for 6 different modes (i.e, stationary, walk, run, bicycle, bus and subway) results into 71.4h (i.e., 16755 × 2.56s × 6 ÷ 3600) of data for training. Likewise 4239 × 6 frames results into 18h (i.e., 4239 × 2.56s × 6 ÷ 3600) of data for testing. In this study no information regarding the users' attributes and location, during data collection, is provided. Therefore, the assessment of this system regarding user and location independency is not possible.

6.2 Generalization

In Table 3 we provide a comparison between different local TMD works regarding their independence with respect to sensor position, user variation, and location. In some works, position independency has been considered, and such TMD systems
Table 3 Generalization requirements fulfilled by local TMD approaches.

| Studies          | Position independency | User independency | Location independency |
|------------------|-----------------------|-------------------|-----------------------|
| Reddy [10]       | Yes                   | Yes               | No                    |
| Heminki [12]     | Yes                   | Yes               | Yes                   |
| Sonderen [13]    | No                    | No                | NA                    |
| Martin [11]      | Yes                   | No                | No                    |
| Guvensan [4]     | Yes                   | Yes               | Yes                   |
| Marra [34]       | NA                    | Yes               | No                    |
| Soares [39]      | NA                    | No                | No                    |
| Liang [24]       | Yes                   | No                | No                    |
| Zhao [23]        | No                    | NA                | NA                    |

Note: NA=Not Available.

have used different ways to show their classifier generalization. For example, some solutions trained the classifier with the data collected from different smartphone positions such as arm, bag, hand, rear/front pocket, chest or waist [10, 4, 11, 12, 24] to validate that the classifier generalization is fulfilled; some other solutions used different approaches to limit the side effects of sensor position changes during data collection [12, 24]. Moreover, in some works, user and location independency are examined; for user independency, data from different users with a variety of attributes is tested, and for location independency some approaches have investigated their system generalization by training and testing classifiers with the data coming from different geographical locations.

6.3 Preprocessing and Feature Extraction

Different local TMD approaches use different techniques to filter the collected data and transforming the filtered data to a computationally efficient set of features. Features are extracted from the whole trip (trip-based extraction), from a segment (segment-based extraction), or from a frame (frame-based extraction). Therefore, in this section we provide a comparison of local TMD approaches regarding the preprocessing and feature extraction steps.

The sensors, sampling rate, window size, preprocessing technique, and feature selection technique influence the accuracy and the resource consumption of TMD systems. Therefore, in Table 4 we provide these parameters for the preprocessing step along with the achieved accuracy.

6.3.1 Reddy

Reddy et al. [10] proposed a noise filtering step in which invalid GPS points are discarded; such invalid points occur when the phone is significantly shielded or if the user is in a covered area. The filtering process also analyses accelerometer data. If too few samples are received from the accelerometer sensor to calculate the frequencies of interest (i.e., frequencies between 1-10Hz are chosen for the accelerometer), this data is discarded as well.

A window of one second is used for the period of classification. Variance along the Discrete Fourier Transform (DFT) energy coefficients between 1-3 Hz from the accelerometer and the speed from the GPS receiver were selected as the feature set.

[4] It is assumed that within each segment the transportation modality remains unchanged.
### Table 4 preprocessing and Feature extraction performed by local TMD approaches.

| Studies | Sensors | Sampling rate (Hz) | Window size (seconds) | Pre-processing technique | Feature selection technique | Overall accuracy |
|---------|---------|--------------------|-----------------------|--------------------------|----------------------------|------------------|
| Reddy [10] | ACC, GPS | ACC:1-10 GPS: Variable [4] | 1 | filtering[a] | CFS frame-based | 93.6% |
| Hemminki [12] | ACC | 60, 100 | 1.2 | low pass filtering | frame-based, segment-based peak-based | 80.79% |
| Sonderen [13] | ACC, Gyr, Mag | 1 [6], 10 15, 30, 60 | 1, 5[c], 10, 20, 30 | NA | frame-based | 98% |
| Martin [11] | ACC, GPS | ACC:5 GPS:1 | 30, 60, 90, 120 | filtering | PCA, RFE segment-based (Movelets) | 96.8% |
| Guvensan [4] | ACC, Gyr, Mag | 5, 10, 15, 50, 100[c] | 5, 10, 20, 40, 60[c], 80 | NA | frame-based | 94.57% |
| Marra [34] | GPS | every 38 seconds | whole trip | filtering, smoothing (Kalman filter) | trip-based | 88.75% |
| Soares [39] | GPS, WiFi, Cellular | between 1, 2 seconds | 90 | filtering | frame-based | 69.8% |
| Liang [24] | ACC | 50 | 2.56, 5.12 10.24 | smoothing, filtering (gravity) | frame-based | 94.48% |
| Zhao [23] | ACC, Gyr | 50 | 2.56 | filtering | frame-based | 92.8% |

Note: ACC=Accelerometer; GPS=Global Positioning System; Mag=Magnetometer; Gyr=Gyroscope; PCA=Principal Component Analysis ; RFE=Recursive Feature Elimination; CFS=Correlation-based Feature Selection; NA=Not Available.

[a] Data filtering removes the data that do not represent the correct information [34].
[b] GPS is not uniformly sampled instead the GPS sampling is started when the user is outdoor
[c] Best choice

This feature set is selected by the use of correlation based feature selection (CFS) method. [5] Accelerometer variance can be used to infer if the user is running and the DFT coefficients help in differentiating between foot-based modes. Moreover, GPS speed can help to determine if the user is still or in a motorized transport mode.

### 6.3.2 Hemminki

Hemminki [12] uses a low-pass filter to remove the jitter from the accelerometer measurements. Then, the measurements are aggregated using a sliding window with a 50% overlap and a duration of 1.2 seconds each. After preprocessing, both horizontal and vertical gravity effects from the accelerometer measurements are removed. Then, features on three different level of granularity including frame-based, peak-based, and segment-based are extracted. Peak and segment-based features describe the movement patterns of vehicles, instead of movements of the user, making these features robust against different device positioning (i.e., it helps to meet position independency requirement).

[5] Correlation-based feature selection (CFS) is a feature subset selector that eliminates irrelevant and redundant attributes.
Frame-based features capture characteristics of high-frequency motions caused by a user during pedestrian activity or motorized periods (i.e., from vehicles engine and contact between its wheel and the surface). For movements with lower frequencies such as acceleration and breaking periods of motorized vehicles, peak-based features are proposed. Segment-based features characterize patterns of acceleration and deceleration periods over the observed segment (i.e., during a period of stationary or motorized movement).

6.3.3 Sonderen
In Sonderen's work [13] there are 25 data-sets each with a different combination of sampling rate and time frame. The sampling rates to be tested are (in Hertz) 60, 30, 15, 10, and 1. As mentioned in Section 6.1, the sampling rate of the raw data is 140 Hz. To change this into 60 Hz, 30 Hz, 15 Hz, 10 Hz, and 1 Hz, rows of data are filtered out. Resulting from this process, 5 new data-sets are made available. Then, for each sampling rate, several different time frames are extracted. The chosen time frames are 1, 5, 10, 20 and 30 seconds long. The time frames are created by splitting the data-sets in parts which results into 25 data-sets per transport mode in total. Each data-set is split in two parts: 70% of the data is used for training the algorithms, and the remaining 30% is used for testing and calculating the accuracy. From the data in each frame, several features such as average, median, minimum, maximum, and interquartile range are extracted (using Matlab [42]). Sonderen suggests that the best candidate data-set sample is 1 Hz with a time frame of 5 seconds; in fact, this choice not only gives a better accuracy but also it consumes less resources.

6.3.4 Martin
In Martin's work [11] GPS and accelerometer data were recorded at a rate of 1 Hz and 5 Hz, respectively. In the preprocessing step, data before and after a transport mode change (within 10 seconds) is removed because mode transitions may yield intrinsic properties uncharacteristic of any one of the five modes of interest (i.e., walk, bicycle, car, bus, rail). Moreover, short trips that did not contain at least 120 seconds of data were removed; they are too short to create the desired time series features, and they are uncharacteristic for representing the correct transport mode.

There are three different relevant techniques introduced in Martin’s article. The first one (described further in the classification step, Section 6.5) uses the raw GPS and the accelerometer time series to identify unique signatures of modes of transportation. The last two techniques summarize extracted features of the time series and build classification models based on these. The extracted features are the results of time windows with 30 second intervals (i.e., 30 s, 60 s, 90 s, 120 s). Features such as mean, median, variance, minimum, maximum, etc. are calculated on speed, acceleration and their iterated differences. Two dimension reduction strategies, named principal component analysis (PCA) and recursive feature elimination (RFE), were used to summarize the feature sets and thereby reduce the dimensionality of the data.

6.3.5 Guvensan
In Guvensan work [4], data from the accelerometer, gyroscope, and magnetometer is gathered at a frequency of 100 Hz. In fact, based on their observation during
data collection, each transportation mode includes at least one minute walking and stationary actions; therefore, the acquired data is evaluated within a 60 seconds window using 40% overlap in order to not miss-classify actions, especially a transition between two activities. Other sampling rates including 5, 10, 20, 50, and 100 Hz, with window sizes of 5, 10, 20, 40, 60, and 80 seconds, are also evaluated in this work to examine their influence on the recall metric (see Section 4). The results suggest that the sampling rate does not seem to have much effect on recall; with window size greater than 60 seconds, the recall starts to decline. In total, 29 time domain features are extracted consisting of 17 common features, and 12 new features. Common features are those that have a proven success in the area of activity and TMD [43]. The new set of features are suggested in order to have a better understanding of the movement of a vehicle (e.g., the total amount of times that a vehicle accelerates, decelerates and remains at a constant speed). To achieve this, the author explores statistics regarding the distribution of the data within a determined range.

6.3.6 Marra

The application proposed in Marra’s work [34] collects users location information approximately every 38 seconds. To reach a balanced trade-off between data quality and battery consumption the proposed app sends requests with different priorities at different intervals: a low priority request every 30 seconds, and a high priority request every 3 minutes; this led to an average sampling rate of 38 seconds in the collected data-set.

In the preprocessing step, the data cleaning process is executed; it filters and smooths the collected data. Data filtering removes the data that does not represent a user’s real position; the smoothing process reduces the random noise present in the data. Two main features are used to filter erroneous GPS points: speed and angle between points. Points with a speed equal to zero and over 150 km/h were removed; moreover, all points with an angle less than 15 degrees and a distance greater than 60m from the previous point were also removed. After completing the data filtering, the Kalman filter [44] is applied to smooth the latitude and longitude of the GPS data. The extracted features from this data-set include maximum, average, and median speed; maximum, average, and median acceleration, median angle, etc.

6.3.7 Soares

In Soares et al. work [39], user location traces (containing altitude, latitude, longitude, precision, and timestamps of the measurement) are captured for 90 seconds. During the tests, location traces of volunteers were captured with an average accuracy of 25.1 meters and frequency between one and two seconds. Traces with an horizontal error margin bigger than 200 meters or timestamps difference lower than one second are discarded. At the end of each 90 seconds of collection, the summarization attributes of the set of traces (called a chunk) are extracted. The summarization attributes extracted are maximum speed, maximum acceleration, and number of direction changes.
Table 5 Local TMD studies with resource consumption analysis.

| Studies          | CPU usage (percentage) | Memory usage (MBs) | Battery usage (watts) |
|------------------|------------------------|--------------------|-----------------------|
| Reddy [10]       | 6.91                   | 29.64              | 0.425                 |
| Hemminki [12]    | NA                     | NA                 | 0.085                 |
| Sonderen [13]    | DT: 44[^a]             | DT: 5.48           | DT: 19.18[^a]         |
|                  | RF: 89                 | RF: 4.90           | RF: 51.14             |
|                  | KNN: 98                | KNN: 6.31          | KNN: 161.43           |

Note: ACC-based=Accelerometer-based solution; DT=Decision Tree; RF=Random Forest; KNN=K-Nearest Neighbors; NA=Not Available.

[^a]The percentage of CPU usage only used for the time it took each classifier to classify a data point once every 5 seconds.
[^b]Note that to make the comparison possible we have converted the author battery measurements in mAh to watts by assuming that the testing smartphone (i.e., HTC One X) voltage is 3.7 Volt.

6.3.8 Liang
Liang et al. [24] performs data preprocessing with Matlab [42] in two steps. In the first step, the gravity component of acceleration measurements, which is generated by earth, is removed. It is important that the gravity component to be removed from the collected acceleration data. The reason being the fact that, gravity component influences all the travellers, thus it does not help to differentiate different modes. The acceleration data after removing the gravity component is called linear acceleration.

In the second step, the data is smoothed to reduce the influence of large fluctuations. These fluctuations result from user sudden movements (such as picking up the phone while driving). The data smoothing is performed with the help of the central moving average algorithm [45]. After preprocessing, the magnitude of acceleration is calculated and the resulting data is divided into small windows to enable real-time TMD. The window sizes considered in this study are 2.56, 5.12 and 10.24 seconds with 128, 256 and 512 values, respectively. The sampling rate of 50 Hz is chosen in order to balance the battery usage and data precision.

6.3.9 Zhao
In Zhao et al. work [23], the data which is sampled at the rate of 50 Hz is filtered to remove the random noise. Then, the data is divided into windows of same size (i.e., 2.56s) and 50% overlap size which later features are extracted from them. In this study, bidirectional long short-term memory (Bi-LSTM) is used to extract the features. Hence, the output of the current moment is related both to the previous and the next state. Bi-LSTM, in contrast to a classical RNN, is not limited to one-way transmission of the state (i.e., from front to back), thus increasing available information.

6.4 Resource Consumption Analysis
In this section different local approaches are compared with regards to their battery, CPU and memory usage.

6.4.1 Battery Usage
Different works have proposed distinct solutions to achieve minimum energy spending. Some used less power consuming sensors (when compared to others) such as the accelerometer [12, 24, 23] for the data collection, or used a low sampling rate [34].
Others, using both the accelerometer and the GPS for data collection, proposed a solution to limit the battery consumption with an algorithm that turns on the classifier and starts GPS logging only when a user status changes to an outdoor setting [10]. The classifier is turned off once the GPS locks are lost for a predefined period of time (e.g., when the user status is indoors). The algorithm proposed in Reddy’s work [10] relies on changes in the primary GSM cell tower as a trigger to check the start of outdoor trips. Thereby, instead of uniformly sampling the GPS receiver (as a highly energy consuming sensor), the classifier turns it on and GPS sampling starts when the user outdoor status is determined. Even so, in rural areas, where cell towers have less density, a large portion of the trip (when it starts) might not be recorded. Furthermore, continuously turning on and off the GPS sensor spends energy because of the so-called tail power state (i.e., many smartphone components such as the GPS keep using energy for a period of time after the end of its activity) [46].

Hemminki et al. [12] performs a coarse-grained evaluation of power consumption (0.085 watts as shown in Table 5). In addition, not using the GPS sensor has helped to outweigh the associated energy costs.

Others, e.g., Martin [11], proposed a method that reduces the dimensionality of the data as much as possible to reduce the substantial burden on battery life.

Another work (i.e., Sonderen’s [13]) avoided GPS recording; instead, it relied on low battery consuming sensors such as the gyroscope, the magnetometer, and the accelerometer to collect data. However, the values presented for battery consumption in Sonderen’s work are highly inaccurate because the application used for measuring the power usage is based on the model of another smartphone (i.e., the developed application for data collection ran on a Samsung Galaxy S4 mini while the resource consumption tests were performed on a HTC One X). The measured values for battery consumption are 1.44 mAh for decision tree, 3.84 mAh for random forest and 12.12 mAh for K-nearest neighbors. To make the comparison possible in Table 5 we converted these values to watts.

Marra et al. [34] claimed that their proposed approach consumes very little battery power by reducing the GPS sampling rate. However, there is no evaluation of such a claim in the article. Instead, at the end of the study, 35 respondents have provided feedback on the app by completing a survey. Their feedback states that battery consumption was acceptable.

Similarly, Soares et al. [39] claimed that users provided a positive feedback about battery consumption while using their application. However, no evaluation and measuring have been done in this work regarding battery consumption.

Liang et al. [39] have a claim over a light-weighted and energy efficient TMD system by using only the accelerometer sensor which consume less energy than the other motion sensors [25]. However, the authors did not provide any evaluation for their claim.

6.4.2 CPU and Memory Usage

Reddy’s work [10], using the Nokia Energy Profiler, performed trials, 20 minutes each, that compared the CPU and memory resources used by the classifier to other activities normally running on a phone such as game playing, music playing, etc.
### Table 6: Classifiers implemented locally.

| Studies      | Algorithms                          | Phone         | Platform | Training approach |
|--------------|-------------------------------------|---------------|----------|-------------------|
| Reddy [10]   | DT, KMC, NB, NN SVM, CHMM, DT+DHMM | Nokia N95    | Symbian  | NA                |
| Hemminki [12]| AdaBoost+DHMM                       | Samsung Nexus 5 | Android | NA                |
| Sonderen [13]| DT, RF, KNN                         | Samsung Galaxy S4 mini | Android | remote            |
| Martin [11]  | KNN, RF                             | NA            | NA       | remote            |
| Guvensan [4] | LR, SVM, DT, RF                     | Samsung Galaxy S4, LG G3 | Android | NA                |
| Marra [34]   | SVM, BN, Dtab, MLP                  | NA            | Android  | NA                |
| Soares [39]  | RB, SVM, DT, RF                     | Samsung Galaxy S3 Mini | Android | NA                |
| Liang [24]   | NB, BN, DT, KNN, RNN, SVM, CNN LSTM, AB | Google Nexus 5X, Google Nexus 6 | Android | remote            |
| Zhao [23]    | Bi-LSTM NN, RNN, LSTM, Multi-LSTM, Bi-LSTM | Huawei P9, Xiaomi | Android | remote            |

Note: DT=Decision Tree; KMC=K-Means-Clustering; NB=Naive Bayes; SVM=Support Vector Machine; KNN=[K]-Nearest Neighbors; HMM=Hidden Markov Model; CHMM=Continuous Hidden Markov Model; RF=Random Forest; LR=Logistic Regression; BN = Bayesian networks; MLP=Multilayer Perceptron; Dtab=Decision Table; LSTM=Long-Short Term Memory; RNN=Recurrent Neural Network; CNN=Convolutional Neural Network; AB=Adaptive Boosting; Bi-LSTM NN=Bidirectional Long Short-Term Memory Neural Network; NA=Not Available.

Sonderen [13] also suggested Java mission control to determine the resource consumption. In this study, the authors attempt to use algorithms with the lowest time and space complexity such as a decision tree.

Note that only a few studies investigated resource consumption of their solution [10, 13, 12]. The results of their measurements regarding battery, CPU and memory usage is summarized in Table 5.

### 6.5 Classification and Training

The choice of which specific learning algorithm a system should use is a critical step for TMD. Most recent works have developed different classifiers on smartphones during the last years. The most widely used classifiers for local TMD approaches are decision tree (DT), random forest (RF), support vector machine (SVM), nearest neighbor (NN), and naive Bayes (NB) [47].

In some studies, two classifiers are combined in different ways, thereby creating a multi-layer or hierarchical classification. For example, decision tree and dynamic hidden Markov model (DHMM) are used in combination in Reddy’s work [10].

As explained in Section 5, having a TMD with 100% accuracy is the ideal goal. Nevertheless, none of the systems reviewed in this survey could achieve such a requirement. In Table 6 we present the smartphones model and their platform on which the ML algorithms have been implemented. Moreover, in this table the training phase approach (i.e., local or remote) is also presented; this table does not mention any work with a local training approach because we did not find any.

Table 7 summarizes the overall accuracy of each mentioned proposed solution along with the walk/stationary accuracy and motorized accuracy (i.e., the average accuracy of motorized modes). The reason for such a separation is that most TMD solutions can detect walk and stationary modes with a higher accuracy which will

---

Kamalian and Ferreira

Page 30 of 41
| Existing systems | Classes | Algorithms | Overall accuracy | Walk/ stationary accuracy | Motorized accuracy |
|------------------|---------|------------|------------------|--------------------------|-------------------|
| Reddy [10]       | stationary, walk bicycle, run, motorized | DT, KMC, NB, NN, SVM, CHMM, DHMM + DT | 93.6% | 90.8% / 97.08% | 94.5% |
| Hemminki [12]    | stationary, walk, bus, train, car, tram, metro | DHMM + AdaBoost | 80.79% | 98.18% / 93.92% | 92.36% |
| Sonderen [13]    | walk, run, bicycle, car | DT, RF, KNN | 98% | NA | NA |
| Martin [11]      | walk, bicycle, car, bus, rail | KNN, RF | 96.8% | 99% | 95% |
| Guvensan [4]     | stationary, walk, car, bus, tram, train, metro, ferry | KNN, RF, NB, J48 | 94.57% | 98.75% / 98.64% | 95.6% |
| Marra [34]       | walk, bus, train, tram, private modes (bicycle and car) | LR, SVM, DT, RF, SVM, CNN, LSTM, AB | 86.75% | 95.83% | 77.08% |
| Soares [39]      | walk, bicycle, bus | SVM, BN, Dtab, MLP | 69.8% | 69.7% | 66.6% |
| Liang [24]       | stationary, bicycle, car bus, subway, train, walk | NB, BN, DT, KNN, RF, RNN, SVM, CNN, LSTM, AB | 94.48% | 93% / 99.7% | 92.47% |
| Zhao [23]        | stationary, walk, run, bicycle, bus, subway | Bi-LSTM NN, RNN, LSTM, Multi-LSTM, Bi-LSTM | 92.8% | 99% / 95% | 96.59% |

Note: DT=Decision Tree; KMC=K-Means-Clustering; NB=Naive Bayes; SVM=Support Vector Machine; KNN=[K]-Nearest Neighbors; HMM=Hidden Markov Model; CHMM=Continuous Hidden Markov Model; DHMM=Discrete Hidden Markov Model; RF=Random Forest; LR=Logistic Regression; BN=Bayesian networks; MLP=Multilayer Perceptron; Dtab=Decision Table; LSTM=Long-Short Term Memory; RNN=Recurrent Neural Network; CNN=Convolutional Neural Network; AB=Adaptive Boosting; Bi-LSTM NN=Bidirectional Long Short-Term Memory Neural Network; NA=Not Available.

[a]Best solution

Table 7 Local TMD approaches modes, classifier and accuracy

affect the overall accuracy. However, it is of great importance to assess different approaches ability regarding fine-grained motorized TMD. Note that in this table, all three reported accuracy metrics (overall, walk, motorised) are the evaluation results of the best proposed solution for each work. Furthermore, note that for Hemminki’s, Guvensan’s and Soares’s works we have calculated the accuracy metrics based on the provided confusion matrix (provided in their article) and Equation 1.

6.5.1 Reddy

Reddy et al. [10] compared different ML algorithms to determine the most accurate classifier. Three distinct metrics (accuracy, precision, and recall) were employed. The final results suggested that a classification system that consists of a decision tree (DT) followed by a first-order discrete hidden Markov model (DHMM) is the best solution with an overall accuracy of 93.6%.

Reddy’s system is implemented on a Nokia N95 smartphone with the Symbian platform. Although both the smartphone and the underlying operating system are
The results of preliminary tests with alternative platforms indicate that the classification is robust against such changes.

The transport modes identified by this system include stationary (i.e., still mode), walk, run, bicycle and motorized. Its inability to differentiate fine-grained motorized modes (e.g., car, bus, train, etc.) is one of its main limitations. Furthermore, it is not reasonable to compare the achieved overall accuracy with other local TMD solutions which have proposed a system with the ability to differentiate between motorized modes (e.g., Liang’s solution).

### 6.5.2 Hemminki

A three stage hierarchical classification system is suggested in Hemminki’s work [12]. At the root, there is a kinematic motion classifier which performs a coarse-grained distinction between pedestrian and other transport modes; it uses a combination of an instance-based classifier with a discrete hidden Markov model (DHMM). The accuracy of the kinematic motion classifier is over 99%. If the kinematic classifier fails to detect a substantial physical movement, the process progresses to a stationary classifier which determines whether the user is stationary or in a motorized transport. When motorized transportation is detected, the classification proceeds to a motorized classifier which is then responsible to classify the current activity into five modalities: bus, train, metro, tram, and car. Each one of the aforementioned three classifiers is considered a variant of AdaBoost as an instance-based classifier.

Adaptive boosting or AdaBoost (introduced by Freund [48]) extends the idea of boosting by tuning the weight of samples which are miss-classified by previous classifiers. The basic idea in boosting is to iteratively train weak classifiers which focus on different subsets of training data and to combine these classifiers into one strong classifier. The AdaBoosting algorithm tries to build a strong classifier from the mistakes of several weaker classifiers. Thus, reducing the bias error which arises when weak classifiers are not able to identify relevant trends in the data.

In Hemminki’s study decision trees with depth of one or two are used as the weak classifiers. Compared to the system suggested by Reddy et al., this approach suggests over 10% higher precision and recall. In addition, not using the GPS sensor has helped to decrease the battery usage when compared to Reddy’s system. Whereas modal change delay (see more details in Section 5), especially for distinguishing between different motorized modes, has considerably increased.

### 6.5.3 Sonderen

Sonderen’s work [13] compares three different ML algorithms: decision tree, random forest, and k-nearest neighbors. The transportation modes are limited to walk, run, riding a bicycle, and driving a car. The algorithms run on a PC for training purposes. Then, an Android application tested all the three algorithms previously mentioned. For testing, 25 data-sets are used; these were made available after the feature extraction step (see Section 6.3). If none of the algorithms can achieve an accuracy above 80% using any of the data-sets, data of another sensor is added and the experiment is repeated. This process continues until an accuracy of 80% is achieved.

A weak classifier is one that performs better than random guessing, but still performs poorly at classification.
reached or there are no more sensors to add. These tests have also been conducted using data of the first volunteer mixed with the data from a second volunteer (i.e., both data-sets collected with 1 Hz frequency and a 5 seconds time frame). The results of this test shows a drop of accuracy for all three ML algorithms when the data of the first volunteer is mixed with the second.

Finally, the best solution is based on the decision tree algorithm; it achieves an accuracy of 98% while using the least resources when compared to the other algorithms (i.e., random forest and k-nearest neighbors). However, as mentioned before, the accuracy decreases when the data of more people is added (i.e., 91% weighted average accuracy of mixed data from two people) resulting into an inefficient approach for a real world setting.

6.5.4 Martin
Martin’s work [11] compares three techniques for predicting several transport modes (walk, bicycle, car, bus and rail). The first technique is an extension of the so-called movelets approach which is introduced by Bai et al. [49]. Movelets are a dictionary-based ML technique based on matching time series vectors; this technique is used for predicting position changes (standing, sitting, lying down, etc.) on the basis of accelerometer readings. It involves partitioning accelerometer time series data into segments (movelets) and clustering the segments known to be from the same mode (e.g., standing) to define a set of characteristic signatures for that mode. For new data, the mode is detected by determining which mode contains movelets that closely matches the new observed time series. The movelets approach is extended to handle the two parallel time series defined by GPS and accelerometer measurements in Martin’s work. The experiments in this work show that movelets performed poorly in predicting the mode of travel. For evaluation, the data-set is split into 60% for training, 20% for validation, and 20% for testing.

The second and third techniques are specialised versions of k-nearest neighbors (KNN) and random forest (RF), incorporated with two dimension reduction strategies referred as PCA and RFE (mentioned in Section 6) to reduce the burden on smartphones battery. Overall, using random forest with RFE introduces the best and computationally efficient solution with an overall accuracy of 96.8%. Furthermore, 10-fold cross validation is used to train each of the RF models (i.e., RFE-RF and PCA-RF models). Leave-one-out cross-validation is used on the training data to determine the optimal value of k for KNN.

Some limitations of this work are: 1) not differentiating between different rail-based transport modes, and 2) proposing a method that only classifies trips that were known to be of a single mode of transportation (which diminishes the system’s ability to predict transitions between modes).

6.5.5 Guvensan
The multi-tiered architecture proposed by Guvensan [4] consists on performing the TMD by employing a vehicular activity detector and a vehicular activity classifier. The vehicular activity detector determines whether a vehicular, stationary or walking activity occurred in the current window. Next, if no stationary or walking states were detected, vehicular activity classification commences. The vehicular activity classifier decides on the type of vehicle used for transportation.
Authors evaluated four different ML algorithms for classification: K-nearest neighbors, naive Bayes, random forest, and J48 (J48 is an open source Java implementation of the C4.5 algorithm).\[7\]

Finally, a segment-based post-processing algorithm, named a healing algorithm, aims to correct the miss-classification results of the previous ML-based solutions. For this purpose, the classified data stream are partitioned into segments using walking activity as a separator. The healing algorithm determines that most activities occur between two walking events and labels the whole segments with the corresponding activity.

The vehicular activity classification performance is evaluated by real time tests conducted with a mobile application running on Android based smartphones: Samsung Galaxy S4 and LG G3. The detected transport modes include stationary (i.e., still), walk, car, bus, tram, train, metro, and ferry with an overall accuracy of 94.57%. In order to tune and examine the effects of the system parameters, 3-fold cross validation has been applied to 70% of the collected data and the system performance is evaluated with the remaining 30% of the data-set (mentioned in Section 6.1).

6.5.6 Marra

Marra et al. [34] suggests an application with different algorithms: 1) activity and trip identification (i.e., dividing the users’ records into activities and trips), 2) trip segmentation (i.e., grouping trips into walking or other-stages), and 3) transport mode detection (i.e., identifying the transport means).

An activity is defined when there are at least 2 successive points within 250 meters radius for at least 10 minutes. In turn, a trip is identified as a movement between two activities. For the trip segmentation, a walk occurs when the user is walking or is waiting for transport in a single place. Other stage occurs when the user travels using some transport means (i.e., car, bus, train or other vehicle). TMD detection consists of assigning a specific mode of transport to the stages that are identified as other stages in the trip segmentation.

A specific mode detection algorithm is developed to use low sampling rate GPS data. The proposed mode detection algorithm is unsupervised and does not rely on ground truth of modes or any statistical inference model. Instead, it uses actual public transport operational data. The actual public transport operational data consists of planned and actual arrival/departure times for all vehicles and all stops in Zurich. The mode detection algorithm uses this operational data to label other stages as being carried out by bus/tram, train, or otherwise a private mode vehicle (i.e., car, bicycle).\[8\]

To determine which vehicle out of a set of possible vehicles best matches the user’s other stage, a likelihood function is used. This function computes the degrees of similarity between a user’s path and the paths of the vehicle to determine the

\[7\] C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan mentioned earlier. C4.5 is an extension of Quinlan’s earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification [50].

\[8\] Note that in this work, bus and tram modes are considered as one mode (bus/tram) since the validation data-set contains only one label for bus and tram.
corresponding vehicle. After assigning modes to other stages, the mode detection algorithm tries to detect missing transfers based on the user’s visited places map (i.e., a personalized map of the places visited by each user from their travel history).

To determine private modes (i.e., bicycle and car) an additional module is integrated to the proposed system. The private mode detection uses machine learning to identify modes. This required a ground truth; therefore, the validation data-set was used to train and evaluate the private mode detection model. The private mode detection module used 70% of the validation data-set as training set, and 30% for the test set.

Several ML algorithms such as logistic regression, support vector machine, decision tree and random forest are tested for private mode detection. The best solution examined was random forest. The results suggest that private mode detection algorithm achieved an overall accuracy of 86.75%.

6.5.7 Soares

Soares [39] proposes a real-time TMD application based on location traces using a data mining technique. These traces are preprocessed, grouped in motion segments, and classified by supervised ML algorithms. The application is made available with training chunks collected by a Samsung Galaxy S3 Mini device running Android version 4.3. The set of traces are classified into motorized and non-motorized by a support vector machine (SVM) and into walk, bicycle, bus, car, and motorcycle by a Multilayer Perceptron.[9] Those chunks classified as non-motorized by the SVM are then classified into walk or bicycle using Bayesian network and decision table. The other chunks which are classified as motorized by the SVM are grouped in bus, car or motorcycle by a decision table. Therefore, this work suggests a hierarchical classification which uses SVM, MLP, decision table and Bayesian network.

The mean accuracy observed by the Multilayer Perceptron inference was approximately 69.8%. Due to the absence of training data for the motorcycle and car classes, these transport modes were not considered in the calculations and evaluation. Given that, the actual transport modes that can be detected by Soares solution are limited to walk, bicycle, and bus. For the SVM inference, the mean accuracy observed was approximately 65.5%; the SVM algorithm classified correctly 632 chunks (out of the 1338 chunks) as motorized. The inference of the motorized travel transport mode done by the decision table algorithm obtained a mean accuracy of 87.5%. As for non-motorized chunks, the SVM algorithm was able to identify 244 chunks correctly. From these chunks, approximately 43.8% were correctly classified by a Bayesian network and 42.2% correctly by a decision table.

6.5.8 Liang

In Liang’s work [24] a deep learning model using convolutional neural network (CNN) is suggested. In this study, a CNN is built on the one-dimension acceleration data to detect stationary, walk, bicycle, car, bus, subway and train in every time

---

[9] A Multilayer Perceptron (MLP) is a deep, artificial neural network. An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. An MLP is often applied to supervised learning problems [51].
window. The proposed CNN model is compared with the other traditional ML algorithms such as random forest (RF), decision tree (DT), K-nearest neighbor (KNN), Adaptive boosting (AB), etc. with different window sizes (see Section 6.3). This comparison suggests that the proposed CNN model in this work outperforms all the other compared ML solutions with 94.48% overall accuracy. They also concluded that among the traditional ML algorithms, RF is the most accurate one. Moreover, this study shows that accuracy improves with larger window sizes.

6.5.9 Zhao
In Zhao’s work [23] a deep Bi-LSTM neural network model is trained to detect 6 modes: bus, bicycle, run, subway, and walk. Bi-LSTM neural network is compared with RNN and other variants of RNN including LSTM, Multi-LSTM, Bi-LSTM. The results suggest that Bi-LSTM neural network outperforms the aforementioned algorithms with the overall accuracy of 92.8%. In order to test the accuracy of the proposed model in a real world setting, it is transferred to an Android smartphone. The results of experiments show the most difficult modes to differentiate by deep Bi-LSTM neural network are subway and stationary.

6.6 Delay Considerations
As explained in Section 2, performing local classification diminishes the delay (i.e., network latency) of a TMD system. Therefore, in this survey, all of the works reviewed have implicitly improved the latency of their solution (in comparison with remote classification). However, only one of them provided measurements of its system delay [13]. In Sonderen’s work [13], the time (in seconds) it took each classifier to process one data point is the following: with DT, 0.384; with RF, 1.175; with KNN, 53.712 seconds. Liang [24] also claimed that by dividing the time series data into small windows, near real time detection with 1 second delay is achieved.

Note that in Hemminki’s work [12], a different definition of latency is presented as it expresses the delay between modal changes (see Section 5). One can conclude that in addition to the classification approach (i.e., remote or local), the window size and the complexity of the classifier’s algorithm affect the delay of a TMD system.

7 Methodology
While making this survey, we were concerned about minimizing bias and ensuring reproducibility of the results regarding the previous work that was considered. Thus, to provide reliable findings from which conclusions can be drawn, we first looked for all TMD approaches which used built-in sensors available in smartphones. Then, we looked for the studies that clearly stated that their TMD application or system runs locally on a smartphone. To ensure the reproducibility of the current research study, we now describe the overall methodology in detail.

To explore the state of the art, we searched four databases using the following keywords: transportation mode detection, mode of travel, travel mode detection, and travel behavior. The databases searched were ScienceDirect (www.sciencedirect.com), ACM digital library (dl.acm.org), MDPI (www.mdpi.com), and IEEE Xplore (ieeexplore.ieee.org).

To find local TMD studies among almost 40 studies that were found with the previous query, we carefully investigated their design, implementation, and evaluation
to find how the proposed solution worked. Some of them, only used a smartphone for the data collection step; these were in fact based on a remote approach (thus being out of the scope). Others clearly mentioned that the implementation of the whole system was done locally on the smartphone. From the most clear studies we found some references to others with some probability of offering a local TMD (using Google Scholar - https://scholar.google.com). Among these, studies such as Reddy’s and Hemminki’s are included in our review, despite being old, given that their solution is not only relevant but they are referenced or compared by many other recent local TMD systems [24, 4, 34].

We excluded those studies and applications which used camera and microphone during data collection (such as Miluzzo’s work [17]), or those that used cameras for annotating the ground truth during the data collection step (e.g., Mun’s work [52]). The reason being that using cameras or microphones raise a lot of concerns regarding users privacy. As a result, such concerns may lead users to not use the application in all their trips and situations, and produce incomplete (and therefore uninteresting) results. Moreover, studies which defined class labels such as sitting, standing, walking down stairs or upstairs are excluded because such class labels are out of the scope (i.e., they are not our goal classes).

8 Conclusion
In this survey we reviewed the local TMD solutions regarding their steps and requirements. We described the most common steps of all TMD systems: 1) data collection, 2) preprocessing, 3) feature extraction, and 4) classification with a previous training phase. A local TMD is clearly differentiated from a remote TMD given that it performs all the above mentioned steps on the smartphone (note that training can still be done on a server remotely).

Local TMD approaches exploit some advantages over remote approaches given that the classification step is completely performed on the smartphone locally: smaller data size, less delay, no need for Internet connectivity, improved users privacy, better or same accuracy, and can take advantage of evolving smartphones.

Each local TMD system made an effort to fulfill (at least) one or a combination of the four main requirements presented: high accuracy, delay considerations, resource consumption analysis, and generalization. To the best of our knowledge none of the existing local TMD systems are able to detect fine-grained transport modes with 100% accuracy. So, all local TMD approaches used different sensors, features and ML algorithms to define the best combination to achieve the highest accuracy.

As far as delay is concerned, all the local TMD approaches have implicitly met this requirement by performing classification locally. However, none of the works reviewed in this survey (with just one exception [13]) measured the computation time and latency of their approach. Thus, we could not make a reasonable comparison regarding the delay measurements of the works reviewed in this survey.

Regarding resource consumption, most local TMD studies proposed a solution to limit the battery consumption; however, only a small number of them provided the evaluation for their claim. Moreover, only two studies measured the CPU and memory usage of the proposed system [10, 13]. The overall observation (e.g., in Reddy’s work) is that most of the CPU usage of the proposed systems is related
to the data collection step (i.e., data sampling). In fact, such CPU usage is mostly due to the sampling of the sensors. For instance, in Reddy's work accelerometer is sampled 32 times in a second, while the feature extraction and classification may get processed every second. Therefore, when compared to the data collection step, the CPU usage for the feature extraction and classification steps is not significant. The amount of RAM used by a local TMD system is related to the ML algorithm complexity, the number of extracted features, and the size of sliding window; thus, it mostly depends on the feature extraction and classification steps.

A generalized classifier is defined as being one that maintains the same accuracy independently of the smartphone position, user variation, and the geographical location (where and when the classification is done). When considering the reviewed local TMD systems, only two of them [12, 4] meet all the generalization requirements. Developing a generalized classifier implies collecting data from various users and locations with different smartphones position.

The accelerometer and the GPS are the two most common sensors used for data collection. The results show that using accelerometer single solutions [12, 24] combined with a suitable classifier and feature extraction can achieve a good result. Although, to differentiate between motorized transportation modes, adding other sensors (such as GPS) can be helpful. GPS only solutions also work well for coarse-grained transportation mode classification, but perform poorly for detecting transport modes with similar speed and acceleration profiles [10].

The overall accuracy achieved by different approaches is quite different; the reason being the differences in their applied solutions for TMD. Furthermore, since studies detect different types of modes, some can differentiate only between motorized and non-motorized, and some consider all rail modes transportation as a single group [11]. For example, in Reddy’s work [10] motorised modes are not differentiated, while Guvensan [4] provides a fine-grained mode detection system. Thus, it is of great importance to compare the different local TMD solutions regarding their walk/stationary accuracy and fine-grained motorized accuracy separately. One can conclude that the more the TMD is fine-grained, the more difficult it is to achieve high accuracy. The comparison of proposed approaches suggests that discriminative classifiers are more common for local TMD (compared to generative classifiers) due to their lower computational cost.

Some approaches examined other algorithms rather than ML based; however, their final assessment suggests that their approach did not necessarily achieved better results. For example, using movelets in Martin’s work [11], or introducing algorithms matching user’s path with the available operational data of the public transports and detecting transitions using user’s travel history in Marra’s work [34].

Furthermore, the comparison of the different proposed ML-based approaches suggests that tree based ML algorithms (i.e., random forest and decision tree) can achieve the best classification results no matter if the different sensors and sampling rates are used.

Acknowledgements
The authors would like to thank the anonymous referees for many valuable comments that helped improving this survey.
Funding
Not applicable

Abbreviations
TMD: Transport Mode Detection; HAR=Human Activity Recognition; ACC: Accelerometer; GPS: Global Positioning System; Mag: Magnetometer; Gyr: Gyroscope; GSM: Global system for Mobile Communication; PCA: Principal Component Analysis; RFE: Recursive Feature Elimination; CFS: Correlation-based Feature Selection; DFT:Discrete Fourier Transform; DT=Decision Tree; KMC=K-Means-Clustering; NB=Naive Bayes; SVM=Support Vector Machine; [K]NN=[K]-Nearest Neighbors; HMM= Hidden Markov Model; CHMM=Continuous Hidden Markov Model; DHMM=Discrete Hidden Markov Model; RF=Random Forest; LR=Logistic Regression; BN=Bayesian networks; MLP=Multilayer Perceptron; Dtab=Decision Table; LSTM= Long-Short Term Memory; RNN=Recurrent Neural Network; CNN= convolutional Neural Network; AB=Adaptive Boosting; Bi-LSTM NN=Bidirectional Long Short-Term Memory Neural Network.

Availability of data and materials
Not applicable

Ethics approval and consent to participate
Not applicable

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable

Authors’ contributions
M.K. and P.F. conceived all the work underlying this article.

Authors’ information
Author details
1 Department of Informatics, University of Oslo, Oslo, Norway. 2 Department of Informatics, University of Oslo, Oslo, Norway.

References
1. Xu, H., Gupta, S., Rosson, M.B., Carroll, J.M.: Measuring mobile users’ concerns for information privacy (2012)
2. Han, Q., Cho, D.: Characterizing the technological evolution of smartphones: Insights from performance benchmarks. In: Proceedings of the 18th Annual International Conference on Electronic Commerce: E-Commerce in Smart Connected World. ICEC ’16. Association for Computing Machinery, New York, NY, USA (2016). doi:10.1145/2971603.2971635. https://doi.org/10.1145/2971603.2971635
3. Chen, K.-Y., Shah, R.C., Huang, J., Nachman, L.: Mago: Mode of transport inference using the hall-effect magnetic sensor and accelerometer. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 1(2), 8 (2017)
4. Guvensan, M.A., Dusun, B., Can, B., Turkmen, H.: A novel segment-based approach for improving classification performance of transport mode detection. Sensors 18(1), 87 (2018)
5. de S. Soares, E.F., Campos, C.A.V., de Lucena, S.C.: Online travel mode detection method using automated machine learning and feature engineering. Future Generation Computer Systems 101, 1201–1212 (2019). doi:10.1016/j.future.2019.07.056
6. Lara, O.D., Labrador, M.A.: A survey on human activity recognition using wearable sensors. IEEE Communications Surveys Tutorials 15(3), 1192–1209 (2013). doi:10.1109/SURV.2012.110112.00192
7. Zhou, X., Yu, W., Sullivan, W.C.: Making pervasive sensing possible: Effective travel mode sensing based on smartphones. Computers, Environment and Urban Systems 58, 52–59 (2016)
8. Shaob, M., Bosch, S., Incel, O.D., Scholten, H., Havinga, P.J.M.: A survey of online activity recognition using mobile phones. Sensors 15(1), 2059–2085 (2015). doi:10.3390/s150102059
9. Prelipcean, A.C., Giddafalvi, G., Susilo, Y.O.: Transportation mode detection—an in-depth review of applicability and reliability. Transport reviews 37(4), 442-464 (2017)
10. Reddy, S., Mun, M., Burke, J., Estrin, D., Hansen, M., Srivastava, M.: Using mobile phones to determine transportation modes. ACM Trans. Sen. Netw. 6(2), 13–11327 (2010). doi:10.1145/1689239.1689243
11. Martin, B.D., Addona, V., Wolfson, J., Adomavicius, G., Fan, Y.: Methods for real-time prediction of the mode of travel using smartphone-based GPS and accelerometer data. Sensors 17(9) (2017). doi:10.3390/s17092058
12. Hemminki, S., Nurmi, P., Tarkoma, S.: Accelerometer-based transportation mode detection on smartphones. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems. SenSys ’13, pp. 13–11314. ACM, New York, NY, USA (2013). doi:10.1145/2517351.2517367. http://doi.acm.org/10.1145/2517351.2517367
13. Sonderen, T.: Detection of transportation mode solely using smartphones. (2016)
14. Nikolic, M., Bierlaire, M.: Review of transportation mode detection approaches based on smartphone data. In: Swiss Transport Research Conference, Ascona, Switzerland (2017). http://infoscience.epfl.ch/record/229181
15. Wang, L., Gjoreski, H., Ciliberto, M., Mekki, S., Valentin, S., Roggen, D.: Enabling reproducible research in sensor-based transportation mode recognition with the sussex-huawei dataset. IEEE Access 7, 10870–10891 (2019)
16. Biancat, J., Brighenti, C., Brighenti, A.: Review of transportation mode detection techniques. ICST Trans. Ambient Systems 1(4), 7 (2014)
17. Miluzzo, E., Lane, N.D., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S.B., Zheng, X., Campbell, A.T.: Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, pp. 337–350 (2008). ACM

18. Liang, Y., Zhou, X., Yu, Z., Guo, B., Yang, Y.: Energy efficient activity recognition based on low resolution accelerometer in smart phones. In: International Conference on Grid and Pervasive Computing, pp. 122–136 (2012). Springer

19. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: Energy efficient smartphone-based activity recognition using fixed-point arithmetic. J. UCS 19(9), 1295–1314 (2013)

20. Byon, Y.-J., Liang, S.: Real-time transportation mode detection using smartphones and artificial neural networks: Performance comparisons between smartphones and conventional global positioning system sensors. Journal of Intelligent Transportation Systems 18(3), 264–272 (2014)

21. Bellavista, P., Berrocal, J., Corradi, A., Das, S.K., Foschini, L., Zanni, A.: A survey on fog computing for the internet of things. Pervasive and Mobile Computing 52, 71–99 (2019). doi:10.1016/j.pmcj.2018.12.007

22. Mitchell, T.M.: Mining our reality. Science 326(5960), 1644–1645 (2009). doi:10.1126/science.1174459. https://science.sciencemag.org/content/326/5960/1644.full.pdf

23. Zhao, H., Hou, C., Alrobassy, H., Zeng, X.: Recognition of Transportation State by Smartphone Sensors Using Deep Bi-LSTM neural network. Journal of Computer and Communications 2019 (2019)

24. Liang, X., Zhang, Y., Wang, G., Xu, S.: A deep learning model for transportation mode detection based on smartphone sensing data. IEEE Transactions on Intelligent Transportation Systems (2019)

25. Google-Android: Motion Sensors. https://developer.android.com/guide/topics/sensors/sensors

26. Mitchell, T.M.: Mining our reality. Science 326(5960), 1644–1645 (2009). doi:10.1126/science.1174459. https://science.sciencemag.org/content/326/5960/1644.full.pdf

27. Hemminki, S.: Advances in motion sensing on mobile devices (2019)

28. Lowry, S.: What is magnetometer sensor?. [online] gearbest.com. available at: https://www.gearbest.com/blog/how-to/what-is-magnetometer-sensor-2866 [accessed 16 sep. 2018]. (2018)

29. Sohn, T., Varshavsky, A., LaMarca, A., Chen, M., Choudhury, T., Smith, I., Consolvo, S., Hightower, J., Griswold, W., de Lara, E.: Mobility detection using everyday gsm traces, pp. 212–224 (2006). doi:10.1007/11853565_13

30. Wang, H., Luo, H., Zhao, F., Qin, Y., Zhao, Z., Chen, Y.: Detecting transportation modes with low-power-consumption sensors using recurrent neural network. In: 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Cloud, Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 1096–1105 (2018). IEEE

31. Chen, J., Bierlaire, M.: Probabilistic multimodal map matching with rich smartphone data. Journal of Intelligent Transportation Systems 19(2), 134–148 (2015)

32. Montoya, D., Abiteboul, S., Senellart, P.: Hup-me: inferring and reconciling a timeline of user activity from rich smartphone data. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 1–4 (2015)

33. Al-Masri, A.: What Are Training, Validation and Test Data Sets in Machine Learning? https://medium.com/datadriveninvestor/what-are-training-validation-and-test-data-sets-in-machine-learning-d1dd1ab09b8e

34. Marra, A.D., Becker, H., Axhausen, K.W., Corman, F.: Developing a passive GPS tracking system to study long-term travel behavior. Transportation Research Part C: Emerging Technologies 104, 348–368 (2019)

35. Kotsiantis, S.B., Zaharakis, I., Pintelas, P.: Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering 160, 3–24 (2007)

36. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In: Advances in Neural Information Processing Systems, pp. 841–848 (2002)

37. Joshi, P.M.: Generative VS Discriminative Models. https://medium.com/@mlengineer/generative-and-discriminative-models-a5637a66a3

38. Joshi, P.M.: Joint Probability Vs Conditional Probability. https://medium.com/@mlengineer/joint-probability-vs-conditional-probability-fa2d47d95c4a

39. Soares, E.F.d.S., de MS Quintella, C.A., Campos, C.A.V.: Towards an application for real-time travel mode detection in urban centers. In: 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), pp. 1–5 (2017). IEEE

40. Huang, H., Cheng, Y., Weisel, R.: Transport mode detection based on mobile phone network data: A systematic review. Transportation Research Part C: Emerging Technologies 101, 297–312 (2019)

41. Yu, M.-C., Yu, T., Wang, S.-C., Wang, Y.-M.: Fine-grained power modeling for smartphones using system call tracing. In: Proceedings of the Sixth Conference on Computer Systems, pp. 153–168 (2011)

42. Version, M.: 9.0. 0 (r2016a). Computer Software, The MathWorks Inc. (2016)

43. Yu, M.-C., Yu, T., Wang, S.-C., Lin, C.-J., Chang, E.Y.: Big data small footprint: the design of a low-power consumption sensors using tree-based ensembles. Journal of Big Data Analytics in Transportation 1(1), 57–69 (2019)

44. Welch, G., Bishop, G., et al.: An introduction to the kalman filter (1995)

45. Savitzky, A., Golay, M.J.J.: Smoothing and differentiation of data by simplified least squares procedures. Analytical chemistry 36(8), 1627–1639 (1964)

46. Pathak, A., Hu, Y.C., Zhang, M., Bahl, P., Wang, Y.-M.: Fine-grained power modeling for smartphones using system call tracing. In: Proceedings of the Sixth Conference on Computer Systems, pp. 153–168 (2011)

47. Bonaccorso, G.: Machine Learning Algorithms: A Reference Guide to Popular Algorithms for Data Science and Machine Learning. Packt Publishing, ??? (2017)

48. Freund, Y., Schapire, R.: “a decision-theoretic generalization of on-line learning and an application to
boosting’, journal of computer and system sciences, vol. 55 (1999)

49. Bai, J., Goldsmith, J., Caffo, B., Glass, T.A., Crainiceanu, C.M.: Movelets: A dictionary of movement. Electronic journal of statistics 6, 559 (2012)

50. Salzberg, S.L.: C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. Kluwer Academic Publishers (1994)

51. Leonel, J.: Multilayer Perceptron. https://medium.com/@jorgesleonel/multilayer-perceptron-6c5db6a8dfa3

52. Mun, M., Estrin, D., Burke, J., Hansen, M.: Parsimonious mobility classification using gsm and wifi traces. In: Proceedings of the Fifth Workshop on Embedded Networked Sensors (HotEmNets) (2008). Citeseer