Applied Intelligence (2022) 52:16021–16050
https://doi.org/10.1007/s10489-022-03214-y

A survey on local transport mode detection on the edge of the network

Mahdieh Kamalian[1] · **Paulo Ferreira**[1] · **Eric Jul**[1]

Accepted: 29 December 2021 / Published online: 19 March 2022

Abstract

We present a survey of smartphone-based Transport Mode Detection (TMD). We categorize TMD solutions into local and remote; the first ones are addressed in this article. A local approach performs the following steps in the smartphone (and not in some faraway cloud servers): 1) data collection or sensing, 2) preprocessing, 3) feature extraction, and 4) classification (with a previous training phase). A local TMD approach outperforms a remote approach due to less delay, improved privacy, no need for Internet connection, better or equal accuracy and smaller data size. Therefore, we present local TMD solutions taking into account the above mentioned four steps and analyze them according to the most relevant requirements: accuracy, delay, resources consumption and generalization. To achieve the highest accuracy (100%), studies used a different combination of sensors, features and Machine Learning (ML) algorithms. The results suggest that accelerometer and GPS (Global Position System) are the most useful sensors for data collection. Discriminative ML algorithms, such as random forest, outperform the other algorithms for classification. Some solutions improved the delay of the proposed system by using a small window size and a local approach. A few studies could improve battery usage of their system by utilizing low battery-consuming sensors (e.g., accelerometer) and low sampling rate (e.g., 10Hz). CPU usage is primarily dependent on data collection, while memory usage is related to the features and complexity of the ML algorithm. Finally, the generalization requirement is met in studies that consider user, location and position independency into account.

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 and accessing to the Internet, nearly all smartphones today provide users with a varied range of built-in sensors and applications which have considerably influenced the 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. The advancements in sensing, computing and storage capabilities of smartphones have made them an effective tool for monitoring the travel behavior (i.e., used transport modes) 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 [2, 3].

Monitoring individuals' travel behavior 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 to assign 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 during a specific period of time [4].

Furthermore, monitoring patients' travel behavior may contribute to controlling the spread of a disease during a pandemic. Likewise, it is a useful way to detect abnormal activities of patients with Alzheimer's, dementia, or other

✉ Mahdieh Kamalian
mahdiehk@ifi.uio.no

Paulo Ferreira
paulofe@ifi.uio.no

Eric Jul
ericbj@ifi.uio.no

[1] Department of Informatics, University of Oslo, Gaustadalléen 23B, Oslo, 0373, Norway

mental pathologies. Hence, patients can be protected from danger and undesirable consequences [5].

In environmental epidemiology, transport mode detection (TMD) is an excellent tool for studies that are concerned with the influence of travel modes on air pollution exposure [6].

When done in real-time, detection of the transport modes can be used, e.g., as an automatic way for issuing tickets for the travelers, particularly when they are using public transport means such as train, bus, subway, etc.

Some previous studies relied on travel logs; others have relied on dedicated wearable sensors for TMD. However, such systems put the burden on research participants and users. Therefore, there has been a shift towards the use of smartphones for TMD in recent years [7]. The most common solution for smartphone-based TMD is to collect data from smartphones' sensors, train a machine learning (ML) algorithm (i.e., a classifier) based on such data, and then use the generated classifier for making future predictions [3, 8–10].

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, etc.) 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 [7].

Performing classification locally on a smartphone has some advantages over a remote approach using some (cloud-based) server(s) as follows. 1) The smaller amount of data exchanges 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 or even intermittent) network between a smartphone and a server; so, latency is much smaller and real-time (i.e., quick enough, based on the 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 (i.e., the result from the corresponding classification). In other words, a user's smartphone is not required to be connected to the Internet during the classification process; the reason is 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 remote 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. [3] proposed a local approach which its overall accuracy (i.e., 94.5%) is equal to the average accuracy (i.e., 94.5%) of a remote approach done by Chen et al. [2]; 6) The growing increase in computing, storage, and power capabilities of smartphones make them an encouraging platform for running ML algorithms and storing data locally rather than on a remote cloud server.

In short, local classification gives us the opportunity for real-time (as previously indicated) TMD by avoiding the latency that transmission to a remote server imposes. In addition, 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, low 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 [11], or to inform a user about the real-time location of relevant buses [8].

1.2 TMD requirements

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

The TMD accuracy is the fundamental requirement; it expresses the number of correct predictions (i.e., used transport modes) made by the classification model.

Delay in TMD consists of three different parts: i) computing-time, which is the time taken for the classification to be done by an ML algorithm, and ii) latency which is the time taken to send the raw collected data to a server and receive the results back. iii) modal change time, which is the time taken to correctly detect the change from a modality to one another.

Resource consumption accounts for the number of resources in terms of battery, CPU, memory (RAM), etc. that the classifier consumes 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 solutions running on smartphones [8, 9].

A generalized system is one in which its accuracy is not affected by smartphone position variability, user variation (i.e., collecting data from a variety of users of different ages, gender, education, job status, etc.) or 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; 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 (and not on remote) TMD given its interesting capabilities (mentioned above) 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 [11–14] is twofold: i) we mainly focus on local TMD approaches, and ii) we introduce a novel perspective of TMD applications by considering the four main requirements of such approaches (that were mentioned above): high accuracy, minimum delay, low resource consumption, and high generalization. To the best of our knowledge, there is no survey that focuses on local TMD while taking into account the requirements mentioned above.

In this survey, we use the following criteria for the selection of studies to be reviewed (for more details on the methodology, see Section 7):

- They use built-in smartphone sensors for the data collection step. Those TMD solutions which use sound samples gathered by the mobile phone's microphone or pictures captured by its camera (such as Miluzzo's work [15]) are not included in this survey. The reason is that these types of data gathering raise many user privacy concerns.
- They have implemented TMD classification locally on smartphones. The training (to generate the classifier) can still be done on a remote server or a smartphone.
- They can 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 [16], they are not part of this survey because of the following reasons: 1) they have covered general activities such as walking downstairs or upstairs, standing, sitting, etc. which are not our goal classes (goal classes for most TMD applications include those used for transportation such as car, bus, train, etc.), and 2) there is another study that has already evaluated them [7].

Some other studies [2, 17] claimed that they proposed a real-time TMD system. However, we did not find any evaluation, proof, or 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 in Section 2. 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 main TMD steps and the different methods which can be utilized in each one. We have dedicated Section 5 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, we describe how different studies have fulfilled the TMD requirements already mentioned. Section 7 describes our searching strategy and Section 8 presents a summary of the main findings. Finally, Section 9 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, their hardware price reduction, combined with their ubiquitous characteristics have made them 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 remote server. In fact, data must be sent to a server only in two cases: i) when we want to send the training data-set to an ML algorithm running in a server (which is normally performed once for most TMD approaches), and ii) when the classification has been locally performed, and we want to analyze the results centrally (and for long-term storage purposes); in both cases, such data is smaller when compared to the data that would have to be sent from a smartphone to a remote 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 is that in a local approach, the network latency between a smartphone and a server is much smaller (when

compared to a remote approach). In fact, there is no need to send the raw collected data to a remote server through the network and wait for the classification; in a local approach, data transmission occurs only in the cases mentioned above. 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 the lack of Internet access.

3. **No need for Internet connectivity**. In a remote approach, for each mode classification, the smartphone must be connected to the Internet. However, in a local approach, there is no such a need (i.e., it is not necessary to have continuous connectivity to the Internet).
4. **Improved privacy**. In a local approach, smaller and less personal information of a user is sent to a server. This feature alleviates the privacy concerns of TMD applications for their users. For example, a user location obtained via the Global Position System (GPS) is not required to be sent to a server and over the Internet for classification purposes.
5. **Better accuracy**. The accuracy of TMD in a local approach is almost the same as with a remote approach. 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 for the use of 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 a users' data can be stored on her/his own smartphone locally (instead of a remote server). This is in line with the current trends of edge and fog computing [18].

3 Available smartphone sensors

TMD can be considered a sub-field of HAR that 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 [9, 19].

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 the accelerometer and the GPS are the most commonly used sensors for data collection. Although using more sensors can provide higher accuracy, it is more common for researchers to carefully use an engineered number of sensors to restrict the energy consumption. Such a hybrid approach (that collects the data using several sensors) can achieve higher accuracy than approaches that only use one sensor.

- An **Accelerometer** is the most commonly used sensor for TMD [3, 8, 9, 16, 20, 21]. It is an electromechanical device that can measure the force of acceleration caused by some movement or gravity on each of the three physical axes. 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 to distinguish 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. In fact, the accelerometer uses about ten times less battery than the other motion sensors, particularly on Android [22, 23].

  Despite the wide range use of the accelerometer for TMD applications, the similarity of acceleration patterns of some vehicles such as cars and buses (or subways and trains) have limited its ability to distinguish all modes correctly [2]. For instance, single accelerometer solutions are not able to differentiate between stationary, cycling and motorized vehicles as accurately as in solutions where the accelerometer is combined with the GPS [8].

  Accelerometer data is beneficial when it comes to the detection of changes in motions during an activity. For example, stationary activities have a very low acceleration 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, while running has a large acceleration variance in comparison with other modes [8].
- A **GPS** sensor is commonly used for recording a person's location (speed can then be obtained rather easily). Most current smartphones provide built-in GPS sensors. They are able to locate a user's position based on the distance of her/his device to four or more visible satellites. Then, the user's speed is also known by locating the user and calculating the time taken to move from source to destination. However, a GPS sensor has 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 lasts only for a few hours with continuously logging [2]. Second, a GPS sensor is unable to work properly in cases that a user is traveling 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 motorized transportation modes with modest accuracy [9, 24]. Using GPS along with other sensors such as the accelerometer, gyroscope and magnetometer improve the accuracy but not significantly [24].

- A **Gyroscope** is able to measure a smartphone's rate of rotation around each one of the three physical axes. Gyroscopes are categorized 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) [25]. The gyroscope complements the built-in accelerometer to understand which way a device is orientated, adding another level of precision. An accelerometer measures the smartphone's linear acceleration or directional movement, whereas the gyroscope sensor measures the angular velocity, tilt and lateral orientation of the smartphone. However, some metrics such as offset error, shock and vibration sensitivity, temperature sensitivity, etc., can affect the gyroscope's quality and calibration [25]. 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 with each one of the three physical axes [26]. 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., the accelerometer) [2]. Comparisons of battery consumption between different sensors suggest that a magnetometer uses about twice the energy than a gyroscope sensor [13]. 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.
- A **GSM** (Global System for Mobile Communications) sensor can be used to collect and detect a smartphone position measuring the signal strength of a device with respect to a base station (implemented as a cell tower) 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 information on a mobile phone's position. For instance, Sohn et al. [27] follows the changes in GSM cell tower observations (up to seven) to detect if a user is stationary, walking, or in a motorized vehicle. However, single GSM sensor solutions similar to those developed in Sohn's work are not useful for fine-grained classification. Also, this sensor is dependent on cells density which varies based on the users' environment [8]. The Ping-pong effect is another limitation of the usage of GSM sensors. The ping-pong effect occurs due to the frequent movement of mobile devices between two or more cell towers or high signal fluctuation at the common boundary of the cell towers. As a result of such changes in signal strength, the system assumes that the user is moving, but actually, he/she is 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 reliable, specifically outside the urban areas. Moreover, the WiFi sensor is (after GPS) the most battery-consuming sensor when used to provide just location information (i.e., without communication) [12]. The ping-pong effect (already mentioned above for GSM sensors) is also typical for WiFi. As a result, WiFi has not been widely used for TMD (see Section 6).

Some other sensors, such as a microphone, light sensor, barometer, Bluetooth, etc., are available in smartphones. In Wang's work [28] the barometer along with three other sensors (i.e., accelerometer, gyroscope, magnetometer) are used to train a long short-term memory (LSTM) model, which can detect bus, car, subway, and train with 96.9% accuracy.

Bluetooth is also used in Chen and Bierlaire's work [29] to provide valuable information about a smartphone's context. For example, the Bluetooth sensor detects other nearby Bluetooth devices. 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, such TMD solutions utilize the information about nearby Bluetooth devices to differentiate a public/private transport context.

In another study [30] the signal levels of Bluetooth are used to infer information about a user's environment. Thus, it is possible to estimate the number of people around a user from the number of discovered Bluetooth networks around a user. Also, if other people happen to move with the user (e.g., when taking the same bus or metro, sharing the same car) these networks will stay in range throughout 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. [29, 30]. 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, we do not address them further in this survey.

4 TMD steps

As previously mentioned, most local TMD approaches rely on some ML algorithms and the detection follows four steps: 1) data collection or sensing, 2) preprocessing, 3) feature extraction, and 4) classification with a previous training phase. A few local TMD approaches have an additional step for post-processing which aims at correcting the miss-classification results of an ML algorithm [3]. In this section, we first describe 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 the 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 her/his smartphone usually during data collection), number of test users and test duration are the key factors for data collection affecting 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 ML-based TMD studies, when building the classification model, the collected data-set is divided into three parts, each used for a different phase of modeling [8]: training, validation, and test set. A training data-set trains the ML model using a supervised ML algorithm (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 [31]. 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 sensing or user errors within the data. In all ML-based approaches, some participants collect a training data-set and label the transport modes used after every trip to build a true basis. 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 a 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. Smartphone mobility introduces such a noise 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 fundamental to use some techniques such as data filtering (i.e., removing the data that does not represent real user positions) and smoothing (to help reduce random noise present in the data) to ensure better accuracy [32].

4.3 Feature extraction

Raw data collected by different sensors is typically segmented into frames using a sliding window, and then features are extracted from such frames. The extracted features are used for learning and classification tasks [12, 33].

The size of the sliding window can affect the classification accuracy, response time, and memory size [13]. On the one hand, a small window size decreases the classification accuracy due to certain features not being effective (e.g., accelerometer frequencies); on the other hand, a large window size may introduce new sources of noise in the data. Moreover, both window size and sampling rate are vital parameters which influence computation and power consumption of ML algorithms.

Most local TMD solutions rely on time and frequency domain features to transform sensing information into lower dimensional sets of features [3, 8, 10]:

- **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 [13, 25].
- **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 [25].

4.4 Classification and training

This section provides an overview of the classification step and 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 be bi-class (i.e., assigning a given email to the "spam" folder or to "inbox") 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 [34]. 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 categorized into 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 predict 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 [35].[2].

Table 1 presents common classifiers and their corresponding categories, for local TMD. The algorithms considered are those 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 to run in a smartphone) [12]. Furthermore, the total studies (last column of Table 1) represents the number of local TMD solutions 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 [7]:

- **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/or 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 a 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 remote server beforehand, it must be adapted to resource-constrained devices such as smartphones (i.e., fewer resources when compared to a server) where the transport mode of a traveler is found.
- **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. In short, the requirement of Internet access imposes more delay to the classification step.

During the last few years, various classifiers have been developed to run locally on smartphones. The most commonly-used classifiers in local approaches (see Table 1)

[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 that 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 [12, 36]. The posterior probability is the probability an event will happen after all evidence has been taken into account [37].

Table 1 Common implemented classifiers on smartphones for local TMD

Classifiers	Category	Relevant systems	Total systems
Random Forest (RF)	discriminative	[3, 20, 22, 32]	4
Decision Tree (DT)	discriminative	[8, 9, 22, 32]	4
[K-]Nearest Neighbors ([K]NN)	discriminative	[3, 8, 20, 22]	4
Support Vector Machines (SVM)	discriminative	[8, 22, 32, 38]	4
Naive Bayes (NB)	generative	[3, 8, 22]	3
Neural Networks (NN)	discriminative	[10, 21, 22]	3
Adaptive Boosting (AB)	discriminative	[9, 22]	2
Hidden Markov Model (HMM)	generative	[8, 9]	2
Bayesian Networks (BN)	generative	[22, 38]	2

are decision tree (DT), random forest (RF), support vector machine (SVM), and K-nearest neighbor (KNN).

In some studies, two classifiers are combined in different ways, thereby creating a hierarchical or multi-layer classification system [2, 8]. For instance, in Reddy's work [8], a decision tree followed by a dynamic hidden Markov model (DHMM), has been proposed as a two-stage classification system.

Most the previous works evaluated between different ML algorithms to determine the most accurate classification model (or their combination) 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 its prediction accuracy [34]. The other metrics are useful to provide an overall insight into classifier performance. Therefore, in this document, we first introduce all the metrics and then emphasize the accuracy of TMD approaches for 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 (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.

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

- **Precision** - indicates the number of instances correctly classified as positive out of the total instances classified as positive (see (2)). Precision is a good measurement for determining when the costs of false positives are high. For instance, in email spam detection, a false positive means that an email that 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.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- **Recall** - indicates the number of items correctly classified as positive out of the total actual positives (see (3)). Applying the same understanding, recall is a good measure to determine when the costs of false negatives are high; e.g., 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 actual positive instances.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

- **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 (4)).

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

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 metrics mentioned above depends on 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 for warning possible respiratory patients to stay at home when CO2 pollution is higher than a specific threshold. In this situation, recall is a crucial metric because detecting cars mistakenly as other modes (FN) leads to measuring less CO2 emission and can put respiratory patients in danger. Also, assume that the municipality defines fines for car drivers in the most polluted days. In this use case, precision is more important than recall. The reason is that fining 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, local TMD approaches have not used this method due to smartphones' computation and battery limitations.
- **Remote training** means that the training of the classifier is usually done on a server or on any 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 [20].

5 TMD requirements

In this section, we introduce the main requirements of TMD systems. Different TMD approaches made an effort to meet (at least) one or a combination of the requirements already mentioned: high accuracy, minimize delay, minimize resource consumption and high 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 for every TMD system. As already mentioned (see the 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 a 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. For example, when a TMD application provides a personalized scorecard for tracking environmental impacts of one's activity, the accuracy should be more than 90% [8]. Another example is a company using a TMD application to count the number of people (and possibly reward them) using their bicycles (in the home-office path) instead of their own car or public transportation; in this case, the accuracy should be 100%. The assessment of previously proposed local TMD systems suggest that none of them is able to achieve the highest accuracy (i.e., 100% accuracy) for detecting all transportation modes (i.e., fine-grained transport mode detection) [3, 9, 20]. 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 that is used to train the classification model. There are at least three techniques that are used to evaluate a classifier's accuracy. One technique is splitting the training data-set by using two-thirds for training and the third for estimating performance. In the second technique, 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. Finally, the leave-one-out validation is a special case of cross-validation in which all validation subsets (i.e., on which the model is evaluated) consist of a single instance. Although this type of validation is computationally more expensive, it is useful when the most accurate estimation of a classifier's error rate is required [34].

Another critical 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 correct 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 that improve the accuracy of a TMD system.

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 (i.e., at a human time scale) to detect the transport mode used. 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, the lower delay is a crucial factor (in addition to accuracy).

For example, a TMD application should alert a user when she/he should start walking towards the bus station in order to be on time for work [11], or to inform the user about the real-time location of the relevant buses [8]); so, in such cases, delay in detection is intolerable, i.e., if the TMD application does not detect the transport mode 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 three components: 1) computing-time, 2) transmission time or (network) latency, and 3) modal change time. Computing time is the time spent by the proposed classifier (running locally) to classify different transport modes taking into account their pre-defined classes. It is clear that a complex classifier and too many features may cause a long 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 avoided. Moreover, the application does not need to be connected to the Internet for detecting the transport modes and presenting the results.

There is another type of delay that can affect the TMD's overall time; we call it modal change time. Modal change time has been considered in a few studies, particularly in Hemminki's work [9] in which this type of delay is named latency. However, it has a different meaning from what we referred to previously. In Hemminki's study, the delay of detecting changes in the transport mode has been introduced [9] meaning the delay in detecting the correct modality when the transport mode has changed. To measure the modal change time, Hemminki used underfill rate metric; this metric measures the rate at which there is a delay in detecting the correct modality when the modality has changed. Hemminki compared the modal change time of their approach with Reddy's solution. From the results, it was observed that detecting changes from or to pedestrian activities in both approaches impose a similar delay. However, Hemminki's system has a higher modal change time than the system of Reddy when it comes to the detection of motorized modalities. To reduce such a delay, Hemminki suggests infusing any sensor capable of stationary detection (i.e., GPS, GSM, or WiFi) when the importance of low modal change time outweighs the battery usage.

5.3 Resource consumption

Resource consumption accounts for the number 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 done 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 smartphone may run simultaneously.

5.3.1 CPU and memory usage

CPU and memory consumption is an important aspect of TMD systems. 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 only one study that has evaluated the CPU and/or memory usage of their solution running on smartphones [8]. However, it is a vital parameter to evaluate a TMD application performance.

5.3.2 Battery usage

Most users expect a TMD application without a significant impact on their smartphone battery. That is why one of the main challenges of a smartphone-based TMD application (along with the quality of its sensors and the local classification) is battery usage minimization.

Different factors influence battery usage in a TMD application. The type and number of sensors (used for the data collection step), amount and dimensionality of collected data, window size, sampling rate, and the computing power used for classification are the important factors that can define the amount of energy that a TMD application uses. 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 is kept minimum (not to influence the user experience during the application usage).

5.4 Generalization

A generalized system, in the context of TMD, is a system in which its accuracy is not significantly affected by different parameters such as the smartphone position (i.e., during data collection), user variation or the geographical location. In fact, a generalized system provides great flexibility regarding the position of the sensing device during data collection (e.g., a smartphone carried in a backpack, pocket, etc.); it also shows almost the same detection accuracy regarding new users with different ages, education, and different travel behaviors (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 user convenience is concerned, it is important to design and develop a TMD system whose accuracy is not dependent on the smartphone placement. Individuals often carry their smartphone in different positions. For example, 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 running often attach the phone to the arm or chest area. Instantaneous changes in the smartphone position may affect the classification and cause low accuracy [3]. Therefore, when designing a TMD system, it is important to collect the data with various smartphone placements to ensure the system generalization regarding the device position.

5.4.2 User independency

Another factor that defines a generalized classifier for TMD is user independency. We can validate a TMD system generalization regarding user independency with the help of two different aspects. The first aspect actually discuss how user-friendly a TMD application is; is it easy for different users of different ages, 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 without the need for user-specific training. The second aspect concerns how the user variation impacts the classification accuracy itself. The proposed classifier should detect transport modes with the same accuracy for users with different walking/running patterns, speed, and travel behaviors. To accomplish this goal, the classifier must be trained with a variety of data that covers a wide range of users with different walking/running patterns, speed, and travel behavior.

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 almost the same accuracy in different geographic locations with various road and public transportation conditions. To achieve this requirement, data-sets should be collected from various locations with different road and public transportation conditions.

For example, when the classifier is trained with collected data from some European cities and then tested with the data collected from various destinations abroad (e.g., 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 is 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, assessing the classifier generalization with different locations is necessary.

6 Existing TMD systems

There are some other surveys reviewing different TMD approaches. Nikolic et al. [12] provides a review of TMD solutions based on smartphones but not necessarily for local classification; Biancat et al. [14] 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. [11] 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. [13] 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 from not investigating dependent parameters to different types of smartphones such as sensor battery usage. Huang [39] provides a systematic review of TMD, which only concentrates on the mobile phone network data that telecommunication operators collect. However, access to such data is not always feasible in all areas and by all operators, thus making Haungs' study impractical. Therefore, in this section, we describe the solutions that local TMD systems apply to meet the requirements already described in Section 5: high accuracy, minimum delay, minimum resource consumption (i.e., battery, CPU and memory usage), and high generalization. We compare local TMD solutions 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 start by presenting data collection in Section 6.1. 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) we present the generalization requirement in Section 6.2, given that it is mainly affected by the first step. We merged 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.

As far as resource consumption (i.e., battery, CPU, and memory) is concerned, we address it in Section 6.4 after the data collection, preprocessing, and feature extraction steps. The reason is that all these three steps impact the number of resources used. We have compared the accuracy of different approaches and presented their classification solution in Section 6.5. Finally, the 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, the data collection duration, and the number of test users are the key parameters for the data collection step of local TMD systems, which influence the classification accuracy.

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

Additionally, position independency of the sensor (where the user puts her/his smartphone usually during data collection) is another critical factor affecting a TMD system's generalization. Table 2 provides a brief comparison of the sensors used, the duration of data collection, the number of test users, users attributes, and sensor positions (during the data collection step) for each local TMD solution that we present next.

6.1.1 Reddy

Reddy et al. [8] proposed a classification system that uses both the GPS and the accelerometer to detect transportation modes. The data-set used for training and testing was obtained from 16 individuals, 8 male and 8 females 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 the 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 the leave-one-user-out mode experiment, an average accuracy of 93.6% and a minimum accuracy of 88.2% were achieved. Overall, the results of these two experiments suggest that the proposed system in Reddy's work can fulfill the user independency requirement. The reason is that the increase in the accuracy achieved by the two experiments is minimal compared to 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 and tested in different locations, Reddy's solution does not meet the location independency requirement.

Table 2 Sensors usage in local TMD systems

Systems	Sensors	Data collection duration	Number of test users	Users attributes	Sensor positions
Reddy [8]	GPS, ACC	120h	16	gender: 8 males and 8 females, age: 20-45	arm, waist, chest, hand pocket, bag
Hemminki [9]	ACC	150h	16	NA	trouser pockets, bag, jacket pockets
Martin [20]	GPS, ACC	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 [3]	ACC, Mag, Gyr	79h	8	various gender, age: 20-45	rear/front trouser pocket, jacket pocket
Marra [32]	GPS	1053 days, 4625 days[1]	41, 625 [2]	39 students, 2 co-authors	NA
Soares [38]	GPS, WiFi, Cellular	NA	18	students	NA
Liang [22]	ACC	14h (about 2h for 7 modes)	4	users	user's preferences
Zhao [21]	ACC, Gyr	71.4h,[3] 18h[4]	11	volunteers	waist
Wang [10]	ACC, Gyr, Mag	25.3h	NA	students	NA

[0]Note: ACC=Accelerometer; Gyr=Gyroscope; Mag=Magnetometer; GPS=Global Positioning System; GSM=Global System for Mobile communication; NA=Not Available.
[1]Validation data-set
[2]Number of user for validation
[3]Training set
[4]Testing set

6.1.2 Hemminki

Hemminki et al. [9] presented a novel accelerometer-based technique for fine-grained TMD on smartphones. One of the main contributions of this work is estimating the gravity component of accelerometer measurements, which provides more accurate and robust estimation during motorized transportation.[3] Hemminki et al., collected over 150 hours of transportation data by 16 individuals from 4 different countries (both for the training phase and for the testing when evaluating the system). Three smartphone models (Samsung Nexus S, Galaxy S2, and S3) were used for data collection with 60Hz and 100Hz accelerometer sampling rates.

The data-set was collected from different placements to ensure the position independency requirement. The three most common placements for smartphones 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 work against variations in device positioning. When using leave-one-user-out cross-validation, the small variance of the results demonstrates their system's robustness across

[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 [9].

user variations, as well. Finally, the generalization capability of 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 Martin

Martin et al. [20] proposed a system that collects data using the smartphone's GPS and accelerometer. The data was collected by 6 students (i.e., 2 male and 4 female) aged 18-25, participating in an undergraduate summer project. Limiting the users to a single-type (only students) group is one of the drawbacks of their system. 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 is that, there is no clear information about the evaluation or test of such a requirement with data coming from a wider variety of locations.

6.1.4 Guvensan

Guvensan et al. [3] introduced a novel multi-tiered architecture which relies on the accelerometer, gyroscope, and magnetometer to collect a total of 79 hours of data involving 8 participants of 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, generate a generalized classifier in relation to the user and position independency requirements. 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 [40] which collected transportation mode data in Taiwan. The results of such a test suggest that the system proposed by Guvensan respects the location independency requirement.

6.1.5 Marra

Marra [32] proposes a GPS tracking application; he claims that their system consumes very little battery 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 involved) 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 to as the 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 to 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 that in many cities such information is not available. Finally, there is no information to justify the position independency requirement in this work.

6.1.6 Soares

Soares et al. [38] 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 user) 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 Iguaçu regions, in the route between these cities and in the downtown of Rio de Janeiro. However, this data does not suffice to respect the location independency requirement.

6.1.7 Liang

Liang work [22] proposes a light-weighted TMD system using only the accelerometer in a smartphone. This solution is not completely clear about being a local TMD approach, but we assume it is local based on the information in its conclusion section. In total, 14 hours of data (about 2 hours of data for 7 different modes: stationary, walk, bicycle, car, bus, train, subway) was collected by 4 users. Users can hold their phones freely in any orientation (according to their preference), making this system generalized regarding the position independency requirement. To minimize the influence of position and rotation, the magnitude value of the accelerometer is calculated. However, the small number of participants in this study limits its user independency capability. 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 on campus. Collecting data for each mode in different locations introduces a generalized approach regarding location independency.

6.1.8 Zhao

Zhao et al. [21] 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 was 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, $16755 \times 128 \times 6$ of data is used for training and $4239 \times 128 \times 6$ of data is used for testing. The authors sampled data with a 50 Hz frequency and divided it into windows (i.e., frames) of 128 values each; each frame is 2.56s long ($128 \div 50$). Having 16755 frames for 6 different modes (i.e., stationary, walk, run, bicycle, bus, and subway) results into 71,4h (i.e., $16755 \times 2.56s \times 6 \div 3600$) of data for training. Likewise, 4239×6 frames results into 18h (i.e., $4239 \times 2{,}56s \times 6 \div 3600$) of data for testing. This work provided no information regarding the users' attributes and location when collecting data. Therefore, the assessment of this system regarding user and location independency is not possible.

6.1.9 Wang

Wang et al. [10] collected a data-set of accelerometer, gyroscope, and magnetometer sensors on an Android application. The total duration of the data set is 25.3h. The data-set includes data of all subway lines in Beijing, 20 bus lines also in Beijing and High Speed Railway(HSR) trains between Beijing and Shanghai. Such a location variation (only for the HSR mode) is not enough to categorize this system as being generalizable regarding location independency. In addition to the general data set, a data set known as shaking data is collected as complementary in each mode to evaluate the system's robustness. Shaking data is an alternative to the movement of smartphones by users. However, this work did not provide any information regarding the placement of the smartphones and the number of test users. Therefore, position and user independency are not reported by this work.

6.2 Generalization

In Table 3 we provide a comparison between different local TMD solutions regarding their independence with respect to sensor position (position independency), user variation (user independency), and location independency. In some solutions, position independency has been considered, and such TMD systems 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

Table 3 Generalization requirements fulfilled by local TMD approaches

Systems	Position independency	User independency	Location independency
Reddy [8]	Yes	Yes	No
Hemminki [9]	Yes	Yes	Yes
Martin [20]	Yes	No	No
Guvensan [3]	Yes	Yes	Yes
Marra [32]	NA	Yes	No
Soares [38]	NA	No	No
Liang [22]	Yes	No	Yes
Zhao [21]	No	NA	NA
Wang [10]	NA	NA	No

Note: NA=Not Available

pocket, chest or waist [3, 8, 9, 20, 22] 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 [9, 22].

Moreover, in some systems, user and location independency are examined; for user independency, data from different users with a variety of attributes is collected, 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 (i.e., data filtering removes the data that do not represent the correct information [32]) and transform the filtered data to a computationally efficient set of features. Features are extracted from the whole trip (trip-based extraction), from a segment[4] (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. In Table 4 we provide these parameters for the preprocessing step along with the achieved accuracy.

6.3.1 Reddy

Reddy et al. [8] proposed a noise filtering step in which invalid GPS points are discarded; such invalid points occur when the smartphone 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. The variance along with 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. 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 [9] 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 levels 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).

Frame-based features capture characteristics of high-frequency motions caused by a user during pedestrian activity or motorized periods (i.e., from vehicle's 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 Martin

In Martin's work [20] 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 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 built classification models based on these. The extracted features are the

[4]It is assumed that within each segment, the transportation modality remains unchanged.

[5]Correlation-based feature selection (CFS) is a feature subset selector that eliminates irrelevant and redundant attributes.

Table 4 Preprocessing and Feature extraction performed by local TMD approaches

Systems	Sensors	Sampling rate[1]	Window size[2]	Pre-processing	Feature selection	Overall accuracy
Reddy [8]	ACC, GPS	ACC:1-10 GPS: Variable [3]	1	filtering	CFS frame-based	93.6%
Hemminki [9]	ACC	60, 100	1.2	low pass filtering	[frame-, segment-, peak-] based	80.79%
Martin [20]	ACC, GPS	ACC:5 GPS:1	30, 60, 90, 120	filtering	PCA, RFE segment-based[4]	96.8%
Guvensan [3]	ACC, Gyr, Mag	5, 10, 10, 50, 100[5]	5, 10, 20, 40, 60[5], 80	NA	frame-based	94.57%
Marra [32]	GPS	every 38 seconds	whole trip	filtering, smoothing[6]	trip-based	86.75%
Soares [38]	GPS, WiFi, Cellular	between 1, 2 seconds	90	filtering	frame-based	69.8%
Liang [22]	ACC	50	2.56, 5.12, 10.24	smoothing, filtering[7]	frame-based	94.48%
Zhao [21]	ACC, Gyr	50	2.56	filtering	frame-based	92.8%
Wang [10]	ACC, Gyr Mag	10	1	filtering[8]	frame-based	91.28%

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.

[1]Sampling rate is presented in Hz; except those mentioned in the table

[2]Window size is in seconds.

[3]GPS is not uniformly sampled instead the GPS sampling is started when the user is outdoor.

[4]Movelets

[5]Best choice

[6]Kalman filter

[7]Gravity filter

[8]Gravity and Auto-regressive Moving Average Model (ARMA) filter

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.

6.3.4 Guvensan

In Guvensan's work [3], 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 of 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 proven success in the area of TMD [41]. The new set of features is suggested to understand better the movement of a vehicle (e.g., the total amount of time 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.

6.3.5 Marra

The application proposed in Marra's work [32] collects users' location information approximately every 38 seconds. To reach a balanced trade-off between data quality and battery usage 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 speeds 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 [42] 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.

6.3.6 Soares

In Soares et al. work [38], user location traces (containing altitude, latitude, longitude, precision and timestamps) 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 1 and 2 seconds. Traces with a horizontal error margin bigger than 200 meters or timestamp difference lower than 1 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.

6.3.7 Liang

Liang et al. [22] performs data preprocessing with Matlab [43] in two steps. In the first step, the gravity component of acceleration measurements, which is generated by the earth, is removed. It is important to remove the gravity component from the collected acceleration data. The reason is that the gravity component influences all the travelers; thus, it does not help to differentiate different modes. After removing the gravity component, the acceleration data is called linear acceleration.

In the second step, the data is smoothed to reduce the influence of large fluctuations. These fluctuations result from sudden user movements (such as picking up the phone while driving). The data smoothing is performed with the help of the central moving average algorithm [44]. After preprocessing, the magnitude of the 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 to balance battery usage and data precision.

6.3.8 Zhao

In Zhao et al. system [21], 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 the same size (i.e., 2.56s) and 50% overlap size from which later features are extracted. This study uses bidirectional long short-term memory (Bi-LSTM) 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.

6.3.9 Wang

In Wangs' system [10] data preprocessing is composed of 3 phases: coordinate transformation (i.e., removing the gravity component from the acceleration on all three axes), Auto-Regressive Moving Average (ARMA) filtering, and data splitting. The ARMA model is a time series model which is usually used to model or compensate for random noise from the sensor readings without introducing a high delay. The ARMA filter preserves the low-frequency part of the signal and removes the high-frequency ones. After data preprocessing, the data is processed into uniform, smoothed, and structured frames. Wang et al. use a frame size of 1 second with a sampling rate of 10Hz. Then, 9 time-domain and 5 frequency-domain features are extracted from the generated frames. Finally, the features extracted by the use of min-max scaling[6] are normalized before being fed to the classifier. Such a normalization accelerates the convergence of the LSTM network and alleviates overfitting[7] to some extent.

6.4 Resource consumption

This section shows different local approaches regarding their battery, CPU, and memory usage. Such approaches have proposed distinct solutions to achieve minimum energy spending. Some used less power consuming sensors (when compared to others) such as the accelerometer [9, 21, 22] for the data collection, or used a low sampling rate [32]. Note that only a few studies (summarized in Table 5) investigated the resource consumption of their solution [8–10]. Along with resource measurements, we provide the list of sensors, sampling rate, and F-measure value.

Except Reddy et al. [8], other studies did not measure CPU and memory usage of their system. Comparison of battery usage of Reddy's and Hemminki's works suggest that (possibly) achieving higher F-measure requires the use of more battery-consuming sensors (such as the GPS). However, such a claim is not justifiable when comparing the sensors used in Wang and Hemminki's systems. A possible explanation for this contrast is the difference in sampling rates in these two systems. Based on a study previously done [46] there is a correlation between sampling rate and battery usage. Moreover (in addition to the sensors used), there are many other factors that influence the F-measure; the ML algorithm, features extracted, number of motorized modes differentiated, etc., are such influencing factors.

6.4.1 Reddy

Reddy's work [8], 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 smartphone such as game playing, music playing, etc. Reddy et al. reported 6.91 percent for CPU usage (i.e., far below services such as playing games, music, or videos) and 29.64 MBs for memory usage (i.e., just above memory usage when the smartphone is idle). To limit the battery consumption, Reddy et al. [8] (i.e., used accelerometer and GPS sensors) proposed a solution with an algorithm that turns on the classifier and starts GPS logging only when a user status changes to an outdoor setting. The classifier is turned off once the GPS locks are lost for a predefined period (e.g., when the user status is indoors). The algorithm proposed in Reddy's work relies on changes in the primary GSM cell tower as a trigger to check the start of outdoor trips. Thus, instead of uniformly sampling the GPS receiver (as a high 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 are less dense, 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) [47]. The reported battery usage for Reddy's system is 0.425 watts.

6.4.2 Hemminki

Hemminki et al. [9] did not analyze the CPU and memory usage but performed a coarse-grained evaluation of power consumption (0.085 watts as shown in Table 5). In addition, not using the GPS sensor helped to outweigh the associated energy costs.

6.4.3 Martin

Martin [20] did not analyze the CPU and memory usage. With regards to battery usage, Martin proposed a method that reduces the dimensionality of the data as much as possible to reduce the substantial burden on battery life.

[6] Min-max scaling re-scales the features in any range into a new range. Usually, the features are scaled between 0-1 or (-1)-1. When min-max scaling is applied, each feature remains the same while taking place in the new range thus, preserving all relational properties in the data [45].

[7] Overfitting occurs when the model perceives the noise and random fluctuations of the training data as concept and learns them. Therefore, it affects the generalization ability of the classifier negatively[45].

Table 5 Local TMD studies with resource consumption analysis

Systems	Sensors	Sampling rate (Hz)	CPU usage (%)	Memory usage (MBs)	Battery usage (W)	F-measure
Reddy [8]	ACC, GPS	ACC:1-10 GPS: Variable	6.91	29.64	0.425	93.64%
Hemminki [9]	ACC	60, 100	NA	NA	0.085	81.08%
Wang [10]	ACC, Gyr, Mag	10	NA	NA	0.063	90.43%

Note:ACC=Accelerometer; GPS=Global Positioning System; Mag=Magnetometer; Gyr=Gyroscope; NA=Not Available; W=Watts. We calculated F-measure based on values for recall, precision and/or confusion matrix provided in the papers.

6.4.4 Guvensan

Guvensan et al. [3] did not analyze the resource consumption of their system. However, this work has implicitly considered battery usage by choosing low battery-consuming sensors (i.e., gyroscope, accelerometer, and magnetometer).

6.4.5 Marra

Marra et al. [32] 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. No information regarding CPU and memory usage is provided.

6.4.6 Soares

Similar to Marra's work, Soares et al. [38] claimed that users provided positive feedback about battery consumption while using their application. However, no evaluation was done regarding battery, CPU and memory consumption in this work.

6.4.7 Liang

Liang et al. [38] have a claim over a light-weighted and energy-efficient TMD system by using only the accelerometer sensor, which consumes less energy than the other motion sensors [23]. However, the authors did not provide any evaluation for their claim. Thus, no information regarding CPU and memory usage was provided.

6.4.8 Zhao

Zhao's work [21] is similar to Guvensan's work with regard to not analyzing any resource consumption. However, this work has used two sensors (i.e., gyroscope and accelerometer) of the sensors used in Guvensan's work by removing the most battery-consuming one from the list (i.e., magnetometer). So, we can say that this work also has implicitly considered battery usage. However, the authors did not evaluate their system regarding any resource (i.e., battery, CPU, and memory).

6.4.9 Wang

Wang et al. [10] used light-weighted sensors (i.e., accelerometer, gyroscope, and magnetometer) with a low data sampling rate (10 Hz) to reduce battery usage. The suggested system in Wang's work consumes 16mAh per hour (i.e., 0.063 watts assuming the battery voltage equal to 3.8 V) with data collecting, processing, and classification on smartphones. This work did not report the CPU and memory usage of the developed system.

6.5 Classification and training

Choosing which specific learning algorithm a system should use is a critical step for TMD. Most current systems 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) [48].

In some systems, two classifiers are combined in different ways, thereby creating a multi-layer or hierarchical

classification; e.g., decision tree and dynamic hidden Markov model (DHMM) are combined in Reddy's work [8].

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 were implemented. Moreover, in this table, the training phase approach (i.e., local or remote) is also mentioned; this table does not mention any system 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 affect the overall accuracy. However, it is of great importance to assess the ability of different approaches regarding fine-grained motorized TMD. Note that in this table, all three reported accuracy metrics (overall, walk, motorized) are the evaluation results of the best proposed solution for each work. Furthermore, note that for Hemminki's, Guvensan's and Soares's systems, we calculated the accuracy metrics based on the provided confusion matrix (provided in their article) and (1).

6.5.1 Reddy

Reddy et al. [8] 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

Table 6 Classifiers implemented locally

Systems	Algorithms	Phone	Platform	Training approach
Reddy [8]	DT, KMC, NB, NN, SVM CHMM, DT+DHMM	Nokia N95	Symbian	NA
Hemminki [9]	AdaBoost +DHMM	Samsung Nexus S Samsung Galaxy S2 Samsung Galaxy S3	Android	NA
Martin [20]	KNN, RF	NA	NA	remote
Guvensan [3]	KNN, RF, J45, NB	Samsung Galaxy S4, LG G3	Android	NA
Marra [32]	LR, SVM, DT, RF [1]	NA	Android	NA
Soares [38]	SVM, BN, Dtab, MLP	Samsung Galaxy S3 Mini	Android	NA
Liang [22]	NB, BN, DT, KNN, SVM, CNN, LSTM, AB, RF, RNN	Google Nexus 5X, Google Nexus 6	Android	remote
Zhao [21]	Bi-LSTM NN, RNN, LSTM, Multi-LSTM, Bi-LSTM	Huawei P9, Xiaomi,	Android	remote
Wang [10]	LSTM	NA	Android	remote

Note: 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; NA=Not Available.

[1] Algorithms used for private mode detection)

Table 7 Local TMD approaches modes, classifier and accuracy

Existing systems	Classes	Algorithms	Overall accuracy	Walk/ stationary accuracy	Motorized accuracy
Reddy [8]	stationary, walk,bicycle, run, motorized	DT, KMC NB, NN SVM, CHMM DHMM+DT[1]	93.6%	90.8%/ 97.8%	94.5%
Hemminki [9]	stationary, walk, bus, , train, car, tram, metro	DHMM+ AdaBoost	80.79%	98.18%/ 93.92%	80.01%
Martin [20]	walk, bicycle, car, bus, rail	KNN, RF[1]	96.8%	99%	95%
Guvensan [3]	stationary, walk, car, bus, tram, train, metro, ferry	KNN, RF[1] NB, J48	94.57%	98.75%/ 98.64%	95.6%
Marra [32]	walk, bus, train, tram private modes[2]	LR, SVM, DT, RF[1]	86.75%	95.83%	77.08%
Soares [38]	walk, bicycle, bus	SVM, BN, Dtab, MLP	69.8%	69.7%	66.6%
Liang [22]	stationary, bicycle, car bus, subway, train, walk	NB, BN DT, KNN RF, RNN SVM, CNN[1] LSTM, AB	94.48%	93%/ 99.7%	92.47%
Zhao [21]	stationary, walk, run, bicycle, bus, subway	Bi-LSTM NN[1] RNN, LSTM Multi-LSTM, Bi-LSTM	92.8%	99%/ 95%	96.59%
Wang [10]	bus, subway, HSR, others, elevator	LSTM	91.28%	88.59%[3]	95.97%

Note: 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; HSR= High Speed Railway; NA=Not Available.

[1]Best solution

[2]Bicycle and car

[3]Others class (i.e., walk and stationary)

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 old, 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. Reddy's system's main limitation is its inability to differentiate fine-grained motorized modes (e.g., car, bus, train, etc.). 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 [22]).

6.5.2 Hemminki

A three stage hierarchical classification system is suggested in Hemminki's work [9]. At the root, a kinematic motion classifier 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 uses a motorized transport. When motorized transportation is detected, the classification proceeds to a motorized classifier which is then responsible for classifying the current activity into five modalities: bus, train, metro, tram, and car. Each one of the three classifiers mentioned above is considered a variant of AdaBoost as an instance-based classifier.

Adaptive boosting or AdaBoost (introduced by Freund [49]) extends the idea of boosting by tuning the weight of samples that 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 that arises when weak classifiers can not identify relevant trends in the data.

In Hemminki's system, decision trees with a depth of one or two are used as the weak classifiers.[8] 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.

[8] A weak classifier is one that performs better than random guessing but still performs poorly at classification.

6.5.3 Martin

Martin's work [20] 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 was introduced by Bai et al. [50]. 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. The mode is detected for new data by determining which mode contains movelets that closely match the new observed time series. The movelets approach is extended to handle the two parallel time series defined by GPS and accelerometer measurements. 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 specialized versions of k-nearest neighbors (KNN) and random forest (RF), incorporated with two dimension reduction strategies referred to 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 one 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).

6.5.4 Guvensan

The multi-tiered architecture proposed by Guvensan et al. [3] consists of 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.

The 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).[9]

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 is 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).

6.5.5 Marra

Marra et al. [32] 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 vehicles). 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 the 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).[10]

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 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 into 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 the 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 was random forest. The results suggest that private mode detection algorithm achieved an overall accuracy of 86.75%.

6.5.6 Soares

Soares [38] 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.[11] Those chunks classified as non-motorized by the SVM are then classified into walk or bicycle using a Bayesian network and a 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 that uses SVM, MLP, decision table, and Bayesian network.

The mean accuracy observed by the Multilayer Perceptron inference was approximately 69.8%. Due to the

[9] 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 [51].

[10] 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.

[11] 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 [52].

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% and 42.2% were correctly classified by a Bayesian network and by a decision table, respectively.

6.5.7 Liang

In Liang's work [22] a deep learning model using convolutional neural network (CNN) is suggested. In this system, a CNN is built on the one-dimension acceleration data to detect stationary, walk, bicycle, car, bus, subway, and train in every time 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.

6.5.8 Zhao

In Zhao's work [21] a deep Bi-LSTM neural network model is trained to detect 5 modes (in addition to stationary): 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 an overall accuracy of 92.8%. The results of experiments show that the most challenging modes to differentiate by deep Bi-LSTM neural network are subway and stationary.

6.5.9 Wang

Wang et al. [10] proposed a novel two-stage long short-term memory (LSTM) classifier to achieve a trade-off among accuracy, complexity, and delay. The proposed classifier consists of two classifiers with different sequence lengths. The shorter one is a binary classifier distinguishing elevator mode and the longer one implements a finer classification of 4 modes, including bus, subway, high-speed railway (HSR), and others (i.e., low-speed modes such as stationary and walking). Finally, the classification results are smoothed in the time domain as a post-processing step. The authors used an ensemble decision module to smooth the predictions in the post-processing step. Wang's system achieved 91.28% overall accuracy and 88.59% accuracy for others class when applying the two-stage classifier.

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 systems reviewed have implicitly improved the latency of their solution (in comparison with remote classification). However, most of the studies did not evaluate all components of the delay of their system. Note that in Hemminki's work [9], a different definition of latency is presented as it expresses the delay between modal changes (see Section 5.2).

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. The results presented by Wang [10] prove our claim when comparing the performance of their proposed system using different LSTM window sizes; it suggests that the accuracy increases with the increase of window size since the prediction attends to more information. However, larger window size leads to higher delays during mode switching (what we refer as modal change time). For example, in this work for a window size of 5 seconds there is a delay equal to 2.5 seconds; for a window size of 10 seconds, the delay is equal to 5 seconds and for a window size of 30 seconds, a 15 seconds delay is reported.

7 Methodology

While doing this survey, we were concerned about minimizing bias and ensuring reproducibility of the results regarding the previously considered works. 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 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 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 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 clearest 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 solutions are not only relevant and important but also they are referenced or compared by some other recent local TMD systems [3, 22, 32].

We excluded those studies and applications which used camera and microphone during data collection (such as Miluzzo's work [15]), or those that used cameras for annotating the ground truth during the data collection step (e.g., Mun's work [53]). The reason is that using cameras or microphones raise many 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 that defined class labels such as sitting, standing, walking downstairs or upstairs are excluded because such class labels are out of the scope (i.e., they are not our goal classes).

8 Discussion

In this section, we provide a summary of our findings, an overview of the limitations of available local TMD systems, and a possible solution for addressing them. This information can guide future TMD system designers.

The first decision for designing a TMD system is to choose the correct set of sensors. The accelerometer and the GPS are the two most common sensors used for data collection. The results show that using single accelerometer solutions [9, 22] combined with a suitable classifier and feature extraction can achieve a good result (i.e., 80.79% overall accuracy for Hemminki's and 94.48% for Laing's). 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 [8].

The most relevant requirement of all local TMD approaches is accuracy (i.e., achieving the highest possible). The overall accuracy achieved by different approaches is quite different due to the differences in their applied solutions for TMD. Some systems 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 [20]. For example, in Reddy's work [8] motorized modes are not differentiated, while Guvensan [3] provides a fine-grained TMD system. Thus, it is important 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 (see Table 1).

Some approaches examined other algorithms rather than ML-based. However, their final assessment suggests that their approach did not necessarily achieve better results; for example, using movelets in Martin's work [20], or introducing algorithms matching a user's path with the available operational data of the public transports and detecting transitions using user's travel history in Marra's work [32].

When it comes to the selection of an ML algorithm, 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 for local TMD(it does not matter if different sensors and sampling rates are used as shown in Tables 7 and 4).

TMD systems are designed to be used by service providers to provide their users with fast responses based on their detected mode. So, designing a local TMD system which considers the delay of the system is of great importance. However, most of the reviewed papers in this survey did not evaluate the delay of their system. As mentioned in Section 5.2, delay of a TMD system consists of three components: computing-time, latency and modal change time. 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 [20]. Moreover, it is clear that a complex classifier with too many features may cause a long computing time. To minimize latency, local classification is preferable to remote classification. The last component of delay is modal change time. When the classifier have sufficient data within a resealable window size, it can faster detect the modality

change. For example, Hemminki et al. [9] suggests that using sensors such as GPS, GSM, or WiFi can minimize modal change time when detecting motorized modes. Wang et al. [10] also suggests that shorter window size leads to lower delay (i.e., modal change time) during mode switching (see Section 6.6 for more details). To control delay of a local TMD system, future works should choose a classifier with low complexity, resealable window size, engineered number of features, and low latency network architectures.

Resource consumption is another important aspect of local TMD systems. Given that smartphones have limitations regarding available resources (i.e., compared to servers), local TMD designers should report the resources consumed by their system to identify the most resource-consuming TMD step. The overall observation about resource consumption (e.g., in Reddy's work) is that most of the CPU usage of the proposed system is related to the data collection step (i.e., 2.88%). 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, compared to the data collection step, the CPU usage for the feature extraction and classification steps is insignificant (i.e., 1.85%). The amount of RAM used by a local TMD system is related to the ML algorithm complexity, the number of extracted features, and the sliding window size; thus, it mostly depends on the feature extraction and classification steps. Frequent sensing by battery-consuming sensors (such as the GPS) can significantly increase the amount of battery usage. Therefore, data collection is the most battery-demanding step. For example, the battery usage in Reddy's work is 0.425 Watts in total, of which only 0.003 is used for feature extraction and running the classifier.

The generalization of a local TMD system is a requirement that can define its usefulness regardless of its accuracy. A very accurate local TMD system which limits the group of users is not useful. Moreover, users will stop using the smartphone application after a while if they must carry their smartphone in a specific place of their body to get the most accurate result. Similarly, if a local TMD system performs very well in one geographical location and poorly in another, it may disappoint users. Developing a generalized TMD system requires collecting data from various users and locations with different smartphones in distinct positions.

9 Conclusion

In this survey, we reviewed local TMD solutions regarding their steps and requirements. We described the most common steps of all such 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 possibility of taking 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, minimize delay, minimize resource consumption, and high level of 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, except two [9, 10], the rest of the studies reviewed in this survey did not measure any component of the delay of their approach.

Thus, we could not make a reasonable comparison regarding the delay measurements in this survey.

Regarding resource consumption, most local TMD studies proposed a solution to limit the battery consumption; however, only three of them [8–10] provided the evaluation for their claim. Moreover, only one study measured the CPU and memory usage of the proposed system [8].

A generalized classifier is defined as being one that maintains the same accuracy regardless 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 [3, 9] meet all the generalization requirements.

Acknowledgements We would like to thank the anonymous referees for many valuable comments that helped improving this survey.

Author Contributions M.K., P.F. and E.J. conceived all the work underlying this article.

Funding Open access funding provided by University of Oslo (incl Oslo University Hospital).

Availability of data and materials Not applicable

Code Availability Not applicable

Declarations

Ethics approval and consent to participate Not applicable

Consent for Publication Not applicable

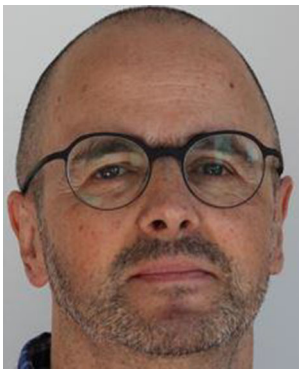
Conflict of Interests The authors declare that they have no competing interests.

References

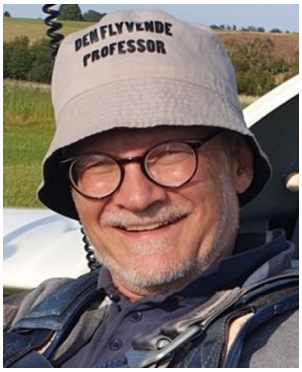
1. Xu H, Gupta S, Rosson MB, Carroll JM (2012) Measuring mobile users' concerns for information privacy
2. Chen K-Y, Shah RC, Huang J, Nachman L (2017) Mago: mode of transport inference using the hall-effect magnetic sensor and accelerometer. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 1(2):8. https://doi.org/10.1145/3090054
3. Guvensan MA, Dusun B, Can B, Turkmen H (2018) A novel segment-based approach for improving classification performance of transport mode detection. Sensors 18(1):87. https://doi.org/10.3390/s18010087
4. de S, Soares EF, Campos CAV, de Lucena SC (2019) Online travel mode detection method using automated machine learning and feature engineering. Futur Gener Comput Syst 101:1201–1212. https://doi.org/10.1016/j.future.2019.07.056
5. Lara OD, Labrador MA (2013) A survey on human activity recognition using wearable sensors. IEEE Commun Surv Tutor 15(3):1192–1209. https://doi.org/10.1109/SURV.2012.110112.00192
6. Zhou X, Yu W, Sullivan WC (2016) Making pervasive sensing possible: effective travel mode sensing based on smartphones. Comput Environ Urban Syst 58:52–59. https://doi.org/10.1016/j.compenvurbsys.2016.03.001
7. Shoaib M, Bosch S, Incel OD, Scholten H, Havinga PJM (2015) A survey of online activity recognition using mobile phones. Sensors 15(1):2059–2085. https://doi.org/10.3390/s150102059
8. Reddy S, Mun M, Burke J, Estrin D, Hansen M, Srivastava M (2010) Using mobile phones to determine transportation modes. ACM Trans Sen Netw 6(2):13–11327. https://doi.org/10.1145/1689239.1689243
9. Hemminki S, Nurmi P, Tarkoma S (2013) Accelerometer-based transportation mode detection on smartphones. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems. SenSys '13. ACM, New York, pp 13–11314. https://doi.org/10.1145/2517351.2517367
10. Wang S, Yao S, Niu K, Dong C, Qin C, Zhuang H (2021) Intelligent scene recognition based on deep learning. IEEE Access 9:24984–24993. https://doi.org/10.1109/ACCESS.2021.3057075
11. Prelipcean AC, Gidófalvi G, Susilo YO (2017) Transportation mode detection–an in-depth review of applicability and reliability. Transp Rev 37(4):442–464. https://doi.org/10.1080/01441647.2016.1246489
12. Nikolic M, Bierlaire M (2017) Review of transportation mode detection approaches based on smartphone data. In: Swiss Transport Research Conference, Ascona, Switzerland
13. Wang L, Gjoreski H, Ciliberto M, Mekki S, Valentin S, Roggen D (2019) Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset. IEEE Access 7:10870–10891. https://doi.org/10.1109/ACCESS.2019.2890793
14. Biancat J, Brighenti C, Brighenti A (2014) Review of transportation mode detection techniques. ICST Trans Ambient Syst 1(4):7. https://doi.org/10.4108/amsys.1.4.e7
15. Miluzzo E, Lane ND, Fodor K, Peterson R, Lu H, Musolesi M, Eisenman SB, Zheng X, Campbell AT (2008) Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application, ACM. https://doi.org/10.1145/1460412.1460445
16. Liang Y, Zhou X, Yu Z, Guo B, Yang Y (2012) Energy efficient activity recognition based on low resolution accelerometer in smart phones. In: International conference on grid and pervasive computing. Springer, pp 122–136. https://doi.org/10.1007/978-3-642-30767-6_11
17. Byon Y-J, Liang S (2014) Real-time transportation mode detection using smartphones and artificial neural networks: performance comparisons between smartphones and conventional global positioning system sensors. J Intell Transp Syst 18(3):264–272. https://doi.org/10.1080/15472450.2013.824762
18. Bellavista P, Berrocal J, Corradi A, Das SK, Foschini L, Zanni A (2019) A survey on fog computing for the internet of things. Pervasive Mob Comput 52:71–99. https://doi.org/10.1016/j.pmcj.2018.12.007
19. Mitchell TM (2009) Mining our reality. Science 326(5960):1644–1645. https://doi.org/10.1126/science.1174459
20. Martin BD, Addona V, Wolfson J, Adomavicius G, Fan Y (2017) Methods for real-time prediction of the mode of travel using smartphone-based GPS and accelerometer data. Sensors 17(9). https://doi.org/10.3390/s17092058
21. Zhao H, Hou C, Alrobassy H, Zeng X (2019) Recognition of transportation state by smartphone sensors using deep Bi-LSTM neural network. Journal of Computer Networks and Communications, 2019. https://doi.org/10.1155/2019/4967261
22. Liang X, Zhang Y, Wang G, Xu S (2019) A deep learning model for transportation mode detection based on smartphone sensing data. IEEE Transactions on Intelligent Transportation Systems. https://doi.org/10.1109/TITS.2019.2951165
23. Google-Android: Motion sensors (2017). https://developer.android.com/guide/topics/sensors/sensors_motion
24. Efthymiou A, Barmpounakis EN, Efthymiou D, Vlahogianni EI (2019) Transportation mode detection from low-power smartphone sensors using tree-based ensembles. J Big Data Anal Transp 1(1):57–69. https://doi.org/10.1007/s42421-019-00004-w
25. Hemminki S (2019) Advances in motion sensing on mobile devices. PhD thesis, University of Helsinki. http://urn.fi/URN:ISBN:978-951-51-5598-6
26. Lowry S (2018) 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]
27. Sohn T, Varshavsky A, LaMarca A, Chen M, Choudhury T, Smith I, Consolvo S, Hightower J, Griswold W, de Lara E (2006) Mobility detection using everyday gsm traces, 212–224. https://doi.org/10.1007/11853565_13
28. Wang H, Luo H, Zhao F, Qin Y, Zhao Z, Chen Y (2018) Detecting transportation modes with low-power-consumption sensors using recurrent neural network. In: 2018 IEEE SmartWorld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud &

big data computing, internet of people and smart city innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp 1098–1105. https://doi.org/10.1109/SmartWorld.2018.00191. IEEE
29. Chen J, Bierlaire M (2015) Probabilistic multimodal map matching with rich smartphone data. J Intell Transp Syst 19(2):134–148. https://doi.org/10.1080/15472450.2013.764796
30. Montoya D, Abiteboul S, Senellart P (2015) 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. https://doi.org/10.1145/2820783.2820852
31. Al-Masri A (2018) What are training, validation and test data sets in machine learning? https://medium.datadriveninvestor.com/what-are-training-validation-and-test-data-sets-in-machine-learning-d1dd1ab09bae
32. Marra AD, Becker H, Axhausen KW, Corman F (2019) Developing a passive GPS tracking system to study long-term travel behavior. Transp Res Part C: Emerg Technol 104:348–368. https://doi.org/10.1016/j.trc.2019.05.006
33. Dogan G (2021) Advances in human activity recognition. Computer 54(5):4–6. https://doi.org/10.1109/MC.2021.3055671
34. Kotsiantis SB, Zaharakis I, Pintelas P (2007) Supervised machine learning: a review of classification techniques. Emerg Artif Intell Applic Comput Eng 160:3–24
35. Ng AY, Jordan MI (2002) On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In: Advances in neural information processing systems, pp 841–848
36. Joshi PM (2018) Generative vs discriminative models. https://medium.com/@mlengineer/generative-and-discriminative-models-af5637a66a3
37. Joshi PM (2018) Joint probability vs conditional probability. https://medium.com/@mlengineer/joint-probability-vs-conditional-probability-fa2d47d95c4a
38. Soares EFdS, de MS Quintella CA, Campos CAV (2017) Towards an application for real-time travel mode detection in urban centers. In: 2017 IEEE 86th vehicular technology conference (VTC-Fall), pp 1–5. https://doi.org/10.1109/VTCFall.2017.8288311. IEEE
39. Huang H, Cheng Y, Weibel R (2019) Transport mode detection based on mobile phone network data: a systematic review. Transp Res Part C: Emerg Technol 101:297–312. https://doi.org/10.1016/j.trc.2019.02.008
40. Yu M-C, Yu T, Wang S-C, Lin C-J, Chang EY (2014) Big data small footprint: the design of a low-power classifier for detecting transportation modes. Proc VLDB Endowm 7(13):1429–1440. https://doi.org/10.14778/2733004.2733015
41. Büber E, Guvensan AM (2014) Discriminative time-domain features for activity recognition on a mobile phone. In: 2014 IEEE Ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP), pp 1–6. IEEE. https://doi.org/10.1109/ISSNIP.2014.6827651
42. Bishop G, Welch G et al (2001) An introduction to the Kalman filter. Proc SIGGRAPH Course 8(27599-23175):41
43. Version M (2016) 9.0. 0 (r2016a). Computer Software, The MathWorks Inc
44. Savitzky A, Golay MJ (1964) Smoothing and differentiation of data by simplified least squares procedures. Anal Chem 36(8):1627–1639
45. Gökhan A, Güzeller CO, Eser MT (2019) The effect of the normalization method used in different sample sizes on the success of artificial neural network model. Int J Assess Tools Educ 6(2):170–192. https://doi.org/10.21449/ijate.479404
46. Horvath Z, Jenak I, Brachmann F (2017) Battery consumption of smartphone sensors. J Reliab Intell Environ 3(2):131–136. https://doi.org/10.1007/s40860-017-0034-1
47. Pathak A, Hu YC, Zhang M, Bahl P, Wang Y-M (2011) Fine-grained power modeling for smartphones using system call tracing. In: Proceedings of the sixth conference on computer systems, pp 153–168. https://doi.org/10.1145/1966445.1966460
48. Bonaccorso G (2017) Machine learning algorithms: a reference guide to popular algorithms for data science and machine learning
49. Freund Y, Schapire R (1999) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci, 55. https://doi.org/10.1006/jcss.1997.1504
50. Bai J, Goldsmith J, Caffo B, Glass TA, Crainiceanu CM (2012) Movelets: a dictionary of movement. Electron J Stat 6:559. https://doi.org/10.1214/12-EJS684
51. Quinlan JR (2014) C4. 5: programs for machine learning
52. Leonel J (2018) Multilayer perceptron. https://medium.com/@jorgesleonel/multilayer-perceptron-6c5db6a8dfa3
53. Mun M, Estrin D, Burke J, Hansen M (2008) Parsimonious mobility classification using gsm and wifi traces. In: Proceedings of the fifth workshop on embedded networked sensors. HotEmNets, Citeseer

Mahdieh Kamalian is a Ph.D. research fellow and a member of the Programming Technology Research Group at the University of Oslo. Her current research area is focused on developing a mobile transport mode detection system to obtain the individual's travel knowledge. She holds a master's degree in the field of computer networks from Amirkabir University of Technology (Tehran Polytechnic) and a bachelor's degree in software engineering from Imam Khomeini International University (IKIU).



Paulo Ferreira holds a Ph.D. from the University of Pierre et Marie Curie (1996) in Computer Science. His MSc (1992) and BSc (1988) are both from Instituto Superior Técnico (University of Lisbon) in Electrotechnical Engineering. He holds the "agregação" since November 2009 from the same University and collaborates with INESC ID where he did the research for several years in the Distributed Systems Group.



Eric Jul has a Ph.D. in Computer Science from the University of Washington (1989). He was a Professor at the University of Copenhagen (1989-2009), a member of Bell Labs in Dublin (2009-2015) and Professor (since 2016) and Head of the Programming Technology Group at the University of Oslo.

He has done research in both Operating Systems and Programming Languages and in the intersection between these areas. He has focused on the Mobility of Data and Programs both at the language level as a co-designer of Emerald and at the implementation level. He currently holds nine patents. He is President of AITO (the organization behind ECOOP) and of the danish Computer Society.