IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 1, 2020, accepted July 30, 2020, date of publication August 7, 2020, date of current version August 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014901

Transportation Mode Detection by Embedded Sensors Based on Ensemble Learning

BANDAR ALOTAIBI, (Member, IEEE)
Department of Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia
e-mail: b-alotaibi@ut.edu.sa

ABSTRACT Context-aware computing has become a certainty due to the widespread use of smartphone devices equipped with sensors. A wide range of services, such as vehicular traffic monitoring and smart parking, can be accomplished with the help of awareness of user mobility. Transportation mode detection (TMD) using machine learning algorithms and the data captured from smartphone embedded sensors have attracted research community attention. In this research, ensemble learning is utilized to differentiate between transportation modes, including walking, standing, riding a train, driving a car, and riding a bus. The ensemble learning consists of three classifiers; each classifier votes independently on the instances, and the majority vote is applied for robust generalization. The proposed method was validated using three datasets; the samples included in these datasets were gathered by smartphone sensors (belonging to heterogeneous users), such as rotation vector sensors, accelerometers, uncalibrated gyroscopes, linear acceleration, orientation, speed, game rotation vector, sound, and gyroscopes. The proposed ensemble learning method achieves an accuracy of 89%, 93%, and 95% on the first, second, and third datasets, respectively, when 10% and 90% of the data are used for testing and training, respectively. On the other set of experiments, in which 30% and 70% of the data are used for testing and training, respectively, the proposed method yields accuracies of 86.8%, 92.1%, and 94.9% on the first, second, and third datasets, respectively. The proposed method shows promising results compared to existing human activity recognition (HAR) methods.

INDEX TERMS Context-aware computing, embedded sensors, ensemble learning, human activity recognition, Internet of Things (IoT), transportation mode detection.

I. INTRODUCTION

Context-aware computing was introduced by Schilit and others in 1994. The terminology is defined as ''the possibility to exploit the changing environment with a new class of applications that are aware of the context in which they are run'' [1]. Recently, context-aware computing has become a reality due to the ease of sensor deployment through IoT devices (e.g., smart wearables such as smartwatches and fitness trackers) or smartphone devices [2]. For instance, embedded sensors are integrated with smartphones to realize the surrounding areas, recognize user activity, and locate user location [3]. A popular research topic (i.e., a type of human activity recognition (HAR)) known as transportation mode detection (TMD) has gained noticeable attention [4]–[6]. TMD is considered a HAR because smartphones carried by users and embedded with sensors such as accelerometers and gyroscopes (a.k.a., microelectromechanical systems (MEMS)) can be used to deduce what activity the phone carrier is performing.

The awareness of a person's transportation mode facilitates planning for urban transportation [7], which was performed in the past via telephone interviews, questionnaires, and travel diaries [8], [9]. With these manual approaches, the traveler could utilize several modes of transportation and might not remember the certain times of the performed transportation modes. Thus, context-aware computing replaces the traditional method, which is considered limited, not up-to-date, erroneous, and expensive [10]. Moreover, human wellness and physical activities can be observed using smartphones [11], smartwatches, or fitness trackers. Such applications can also provide daily activities and the number of calories burned. Additionally, the activities that affect the environment can also be estimated, and environmental hazard exposure can be traced. Knowing the transportation mode of travelers helps shops send real-time information to customers in the

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

form of advertisements. For instance, detecting a consumer driving a car allows for gas stations to send vehicle services or gas coupons as advertisements. Unlike life critical systems that cause death or injuries to people, environmental hazards, or expensive equipment damage, such as heart-lung machines, ventilation systems, and fire alarms, where an ideal 100% precision is essential, transportation mode detection can tolerate a margin of error ranging from 5% to 10%.

Currently, three distinguished types of methods are utilized to detect transportation modes: sensor-based techniques [12], global positioning system (GPS)-based techniques [13], and cellular-network-based attempts. Furthermore, several research papers [14]–[16] have attempted to build patterns of transportation modes and improve the accuracy of these techniques. Moreover, some industries, through their applications and operating systems (e.g., Android operating system), have implemented their own TMD in which users are informed of their mode of transportation. However, some of these applications or systems limit their capabilities of detecting activities to four categories: riding a bicycle, riding in a vehicle, standing, and walking [3]. All of the techniques share the following identical general principle for the detection of the transportation mode: collecting raw data from sensors integrated with mobile phones, preprocessing the raw data to be fed into the classification model, feeding part of the preprocessed data into the classifier for training purposes, and passing the rest of the preprocessed data to the predictor for generalization purposes.

Although TMD methods have achieved considerable success in many applications, there are still some limitations regarding the studies that have been proposed to help improve the usefulness of these methods. First, due to the lack of a benchmark dataset, the published results are hard to compare. Additionally, the proposed techniques are not well defined because of either the limited number of smartphone users involved in the testbed or the heterogeneity of sensors that have been utilized (i.e., it is hard for industries to adopt one of these methods). Fortunately, two recent datasets [3], [17] have been published in which researchers can compare their results. Moreover, a large number of proposals have used binary classification in which the classifier needs to distinguish between only two modes of transportation. For instance, smart parking techniques only considered modes of transportation: riding a vehicle and walking. The notification is sent once a busy or free area is found [18]. Finally, various techniques have been proposed to detect the mode of transportation using a single machine learning algorithm to improve the accuracy. Certain sets of samples might be fit well by a single machine learning algorithm; however, overfitting or underfitting might become visible to the rest of the samples. Thus, even with parameter optimization, an upper limit of accuracy might be reached when utilizing one machine learning algorithm. Combining various machine learning algorithms (ensemble learning) is used to overcome this shortcoming [19]. Recently, several ensemble methods, such as random forests (RFs), gradient boosting (GB), and extreme gradient boosting (XGBoost), have been investigated to differentiate the mode of transportation. However, these ensemble methods can only combine machine learning algorithms with identical types, such as decision trees (DTs). However, ensemble methods that combine machine learning algorithms with different types, such as stacked learning, have not been utilized by the TMD research community. Stacked learning bundles lower level models of different types by a higher level generalized model to maximize accuracy.

Motivated by the released public datasets and the room for improvement, a stacked learning model consisting of various machine learning algorithms with different types (i.e., the investigated machine learning algorithms in this study that generate the most accurate final prediction when bundled together in the stack) is presented to detect the transportation mode using available smartphone sensors that remarkably improve the detection accuracy compared to TMD methods.

The contributions of this research can be summarized as follows:

1) A comprehensive study of different machine learning algorithms is presented to investigate which machine learning algorithm best fits the problem of TMD.
2) Based on the observations from this study, a new ensemble method is proposed to differentiate transportation modes. This method achieves promising results compared to existing techniques.

The remainder of this article is organized as follows. Section II summarizes the related methods, the proposed method is introduced in Section III, and IV presents the experimental settings and the utilized dataset that help to evaluate the proposed method. Section V presents the parameter optimization, Section VI validates the proposed method and analyzes the results, Section VII discusses the results, and Section VIII concludes the research paper.

II. RELATED WORK

Due to the widespread use of smartphones and the Internet of things (IoT) devices that are equipped with embedded sensors, the context-aware computing topic has regained interest in the research community. TMD is an important field in which embedded sensors play an integral role in fulfilling its purpose. TMDs can be categorized into three main types of techniques: sensor-based, GPS-based techniques, and cellular-network-based attempts. GPS-based techniques are the most famous category because of the ease of accessibility to GPS in smartphones and the high achieved performance in terms of accuracy in differentiating between pedestrian movements and riding vehicles. The GPS-based techniques share two shortcomings: the accuracy degrades once the smartphone carrier enters an indoor environment or passes by an urban canyon due to multipath fading and smartphones equipped with battery-constrained capability (i.e., these methods consume batteries) [15].

Cellular network-based attempts employ record-keeping via three techniques to determine the transportation mode of mobile phone owners: signaling data, handovers, and call detail records (CDRs). These records are kept by telecommunication companies for management assistance, maintenance, and billing. Consequently, users are not required to make any efforts and are not aware of the stored records that can be used to estimate their locations and mode of transportation through cellular towers. These methods share a deficiency related to revealing the data privacy of end users. Although mobile phone networks employ anonymity mechanisms to conceal the identity or location of end users before data analysis, recognizing four spatiotemporal points can be a key factor in revealing the identity or location of a given user with high probability. Thus, new techniques, i.e., those compatible with governmental authorities or intergovernmental organizations (e.g., the European Union (EU)), and regulations such as general data protection regulations (GDPRs) should be developed to protect the data privacy and anonymity of mobile phone users [20].

Sensor-based methods use the raw data captured by smartphone embedded sensors. The captured data in the form of series reading are preprocessed, and the dimensionality of the captured data might be reduced to improve the performance or to make the approach less complex. The preprocessed data are then fed into a machine learning algorithm for training purposes. Consequently, the weight of the training phase is used to predict the mode of transportation captured by the smartphone embedded sensor. The most likely smartphone embedded sensor that best fits the area of the TMD is the accelerometer, which has two advantages: its ability to help detect human activities and its light energy consumption [21]. One of the most widely used machine learning algorithms in TDM is RFs [22]–[24].

The authors of [4] investigated several machine learning algorithms to generalize a multiclass method capable of recognizing the mode of transportation. The dataset that the authors used consists of five classes: bike, car, bus, walk, run. The authors tested various machine learning methods, including support vector machines (SVMs), DT, k-nearest neighbors (KNNs), bagging, and RFs. The raw data were captured from three smartphone embedded sensors: the rotation vector, accelerometer, and gyroscope. During the training and testing phases, the authors utilized both the out-of-bag error and k-fold cross-validation to select the model that performs better. A feature selection method known as minimum redundancy maximum relevance (mRMR) was used to eliminate the features that negatively affect the classifier performance. The authors found that data captured by smartphone embedded sensors provide meaningful patterns that distinguish between different types of transportation modes. The authors compared the performance of the tested machine learning algorithms after applying a feature selection method and found that the RFs ensemble method and SVMs performed better than the other methods. The authors also attempted to combine both the simulated annealing (SA) algorithm (i.e., to combine important features that could improve the performance) and RFs to improve the performance.

Xiao *et al.* [25] proposed a TMD method based on ensemble learning in which three machine learning algorithms were combined. The authors used the GPS to detect the transportation mode. To create global features and extract various local features, the authors utilized a statistical technique in which features were generated from subtrajectories. Then, these features were fed into the classifier in the training phase. The ensemble model consists of three base methods: GB, XGBoost, and RFs. The most accurate method among the base estimators is the XGBoost method (its accuracy is 90.77%) when it is applied to the GEOLIFE dataset [26]. The authors also reduced the time required to predict the mode of transportation by using a feature selection method (i.e., a tree-based ensemble technique).

The authors in [27] studied transportation mode classification utilizing big data generated by smartphone embedded gyroscope, magnetometer, and accelerometer sensors. They investigated three well-known machine learning algorithms to recognize transportation modes, specifically, SVMs, KNNs, and DTs. The authors used large-scale experiments in which the performance of these methods was compared in terms of accuracy, model size, and prediction time. The performed experiments show promising results in which SVMs perform better than the other two methods in terms of accuracy, but it was the most expensive method in terms of time.

Shafique and Hato [28] examined four machine learning algorithms, specifically adaptive boosting (AdaBoost), RFs, DT, and SVM, to differentiate the exercised mode of transportation. There are four detected modes of transportation: walk, train, bike, and car. The generated data are collected from three Japanese cities, and the target sensor is an accelerometer. Among the tested methods, RFs outperform the other methods. The second best method is DT, then SVMs and the worst method in terms of accuracy is AdaBoost.

Fang *et al.* [29] proposed a deep learning-based approach to effectively construct a nonlinear shape between labeled and sequential data captured by sensors. This approach utilized a deep neural network (DNN) and built two types of feature sets. One feature set was small compared to the other feature set to investigate the tradeoff between performance and battery consumption. The other only considered the performance of the proposed method. The authors compared their method with three machine learning algorithms: SVMs, KNNs, and DT. The proposed method yielded better results compared to the other three methods.

Liang and Wang [30] proposed a deep learning technique (i.e., convolutional neural network (CNN)) to automatically detect the transportation mode using data generated by accelerometer sensors. Various machine learning algorithms are tested to compare the proposed method with the other methods. The authors tried several CNN architectures and proposed a lightweight and accurate architecture. The proposed architecture is compared with several machine learning algorithms. The proposed CNN model outperforms the

other methods, and Gaussian naive Bayes (GNB) is the least accurate method among the tested methods.

III. PROPOSED METHOD

The proposed framework shown in Fig. 1 utilizes several machine learning algorithms to discriminate the mode of transportation performed by the subjects. The predictions of three machine learning algorithms namely, DT [31], RFs [32], and GB [33], are fed into an ensemble to better generalize the predictions by taking the majority vote of these three algorithms. Subsequently, the predictions of the ensemble along with two other machine learning algorithms, namely, KNNs [34] and random subspace [35] (a.k.a., bagging) are bundled into stacked learning in which a neural network architecture known as multilayer perceptron (MLP) [36] is used to eventually predict the mode of transportation. Twelve machine learning algorithms are intensively tested using two data splits to determine which machine learning algorithms when bundled together in the stack produce a final prediction that best fits the TMD problem. Thus, the base estimators are selected because when bundled together in the stack, the final prediction of the stack is the most accurate. With the exception of SVMs that are eliminated because of the time complexity (i.e., linear SVM (LSVM) and nonlinear SVM (NLSVM)), these machine learning algorithms, including RFs, bagging, GB, XGB, AdaBoost [37], DT, MLP, KNNs, logistic regression (LR), and GNB are tested in each phase of the proposed method in an interchangeable manner, and the most accurate attempt is chosen.

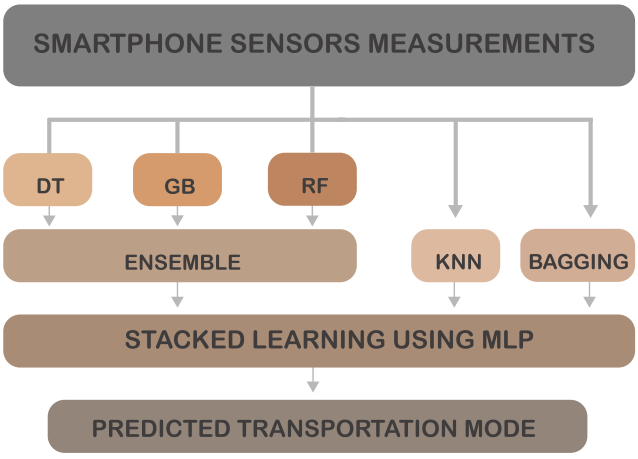


FIGURE 1. Representation of the proposed stacked method that is bundled with three machine learning algorithms, namely, KNNs, bagging, and an ensemble method consisting of three base estimators: DT, GB, and RFs.

More formally, to train the model, let s represent the activity measurements captured by the subject smartphone embedded sensors and t denote the target class labeling that activity, thus:

$$t = \begin{cases} 0 & \text{if } s \text{ represents standing still activity} \\ 1 & \text{if } s \text{ represents riding a bus activity} \\ 2 & \text{if } s \text{ represents riding in a vehicle activity} \\ 3 & \text{if } s \text{ represents riding a train activity} \\ 4 & \text{if } s \text{ represents a walking activity} \end{cases}$$

The whole training set can be denoted by:

$$d = (s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n) \tag{1}$$

Moreover, the feature space F (i.e., always the vertical depiction of the dataset) consists of several characteristics representing smartphone embedded sensor measurements applied to all the samples in the dataset. For instance, the first feature f_1 in the dataset can represent the average of the first sensor measurement, the second feature f_2 the standard deviation of the first sensor measurement, and so on.

The proposed model is divided into three phases. In the first phase, the samples and their labels are assigned to three classifiers: DT, GB, and RFs. Given both the sample vectors in the form of real numbers: $S_n \in R^i$, where $n = 1, \ldots, N$ represent the samples and the target vector $T \in R^j$, the DT classifier groups the instances with identical labels together by recursively partitioning the dimension space. Since it is a multiclass classification problem, the output is in the form of $0, 1, \ldots L-1$, for a given decision node d, exemplifying a region R_d in the dimension space in which the observation is Z_d, thus

$$r_{dl} = \frac{1}{Z_d} \sum_{S_n \in Rd} I(T_n = l) \tag{2}$$

is the probability of l observations in the decision node d, where $I(.)$ is an indicator function that receives 1 if the statement is true, and 0 otherwise; therefore, the value of $I(T_n = l)$ is 1 if this statement is true, and 0 otherwise.

Then, the Gini impurity function G is applied

$$G(X_d) = \sum_{l} r_{dl}(1 - r_{dl}) \tag{3}$$

where X_d represents the training data in decision node d.

In that phase (i.e., the first phase), the samples and their labels are given to another machine learning technique, namely, GB, in which base estimators (a.k.a., weak learners), usually in the context of DTs, are the building blocks that fulfill the demand of the final substantial ensemble prediction model. DTs grow sequentially with respect to errors. Each tree learns from the mistakes of the previous tree to improve performance. Let $M_1(s) = t$ denote the DT model that fits the data. The next DT $H_1(s)$ learns from the mistakes of the previous DT

$$H_1(s) = t - M_1(s) \tag{4}$$

This new DT is added to GB, which is denoted as

$$M_2(s) = M_1(s) + H_1(s) \tag{5}$$

This procedure continues until interruption occurs by a certain mechanism such as cross validation. The ensemble model, known as an additive model of o individual trees, is built after several stages are performed (similar to other boosting methods), as shown in the next equation, and an arbitrary differential loss function is optimized for generalization purposes [38].

$$m(s) = \sum_{o=1}^{O} m^{o}(s) \tag{6}$$

The third machine learning algorithm utilized in this phase is RFs. RFs is an ensemble method consisting of trees, and each tree utilizes random variable collection. Given the samples and their labels to RFs, the aim is to predict the class target t using a prediction function $f(S)$ in which this function is used to reduce the loss value and defined by a given loss function. The prediction function is constructed utilizing some base estimators $e_1(s), \ldots, e_i(s)$. The base estimators are integrated to establish the ensemble predictor $f(s)$ using voting as follows:

$$f(s) = arg \max_{t \in T\prime} \sum_{i=1}^{I} I(t = e_i(s)) \tag{7}$$

where $T\prime$ is a set of potential values of t.

In the second phase, the predictions of the three classifiers are passed to the ensemble. Let the classifier's decision of c^{th} be $D_{c,t} \in \{0, 1\}$. where $c = 1 \ldots, C$ represent the classifiers. If the target class $t\prime$ is chosen by the c^{th} classifier, then $d_{c,t} =$ 1 and 0 otherwise. The plurality voting is utilized in which the predicted class is the one that has received the highest number of votes, as shown in the next equation [39]:

$$\sum_{c=1}^{C} D_{c,t} * = \max_{c} \sum_{c=1}^{C} D_{c,t} \tag{8}$$

Two other machine learning techniques come into play in this phase: KNNs and bagging. The input of the KNNs algorithm is the k-closest training instances constructed in the feature vector. The output is a member of a specific class in which a given object is selected by the neighbors of that object using a plurality vote. The function of KNNs is approximated locally, and the classification is carried out after postponing all computations; thus, KNNs is a typical example of lazy learning or instance-based learning [40]. Let $K(s) = t$ denote the label function in which a training instance is assigned a class target out of l possible labels $t \in 0, 1 \ldots, L-1$ by that function. Thus, the nearest neighbors of k for a query data point $s^{[p]}$ are:

$$Q_k = \left\{ \left\langle s^{[1]}, K\left(s^{[1]}\right) \right\rangle \ldots, \left\langle s^{[k]}, K\left(s^{[k]}\right) \right\rangle \right\} \tag{9}$$

and the hypothesis of the KNNs can be defined as:

$$h\left(s^{[p]}\right) = arg \max_{t \in \left\{0,1\ldots,L-1\right\}} \sum_{i=1}^{k} \delta(t, K\left(s^{[i]}\right)) \tag{10}$$

where δ is a delta function known as the Kronecker delta named after its inventor Leopold Kronecker.

The last technique in this phase is bagging (a.k.a., random subspace). Different bootstrap samples are generated by invoking a base learning algorithm to train several base learners. Subsampling the data with replacement in the training set produces bootstrap samples that have the same size as the size of the training set. Once the base learners are acquired, the random subspace technique is used to combine the base learners and takes the majority vote in which the prediction occurs when a specific class gets the majority. More formally, let B denote the base learning algorithm, and R is the number of rounds in the learning process. For all the rounds $r = 1 \ldots, R$, bootstrap samples b_s are produced from the data d (i.e., $d_r = b_s(d)$), and a base learner h_r is trained from the bootstrap samples $h_r = B(d_r)$. Thus, the iteration of this process generated the following output.

$$f(s) = arg \max_{t \in T\prime} \sum_{r=1}^{R} I(t = h_r(s)) \tag{11}$$

where the value of $I(t = h_r(s)$ is 1 if this statement is true and 0 otherwise [41].

In the last phase, a metalearner (i.e., a special type of DNN called MLP) is utilized to combine the second phase learning algorithms, namely, ensemble learning, KNNs, and bagging, that have been produced from the training data. At least three layers are involved in constructing MLP: an input layer, a hidden layer, and an input layer. The hidden layer is the computational unit (neuron). The objective of the hidden layer is to receive an input from other sources or hidden layers and generate an output. The following equation is utilized by MLP:

$$f(s) = V_2 j(V_1^T s + a_1) + a_2 \tag{12}$$

where V_1 is the input layer weight, V_2 is the hidden layer weight, a_1 is the hidden layer bias, a_2 is the output layer bias, and j is the activation function. This research utilizes the hyperbolic tangent activation function, which can be represented as:

$$j(x) = \frac{y^x - y^{-x}}{y^x + y^{-x}} \tag{13}$$

where $y^x - y^{-x}$ represents the hyperbolic sine of y and $y^x + y^{-x}$ represents the hyperbolic cosine of y.

$f(s)$ is the size of a vector of the number of classes that passes via the softmax function as follows:

$$softmax(x)_n = \frac{exp(x_n)}{\sum_{i=1}^{k} exp(x_i)} \tag{14}$$

The nth element of input to softmax is denoted by x_n, and the number of classes is represented by k. This function produces a vector consisting of the probabilities in which sample s is categorized to each class and the class with the highest probability is the output [42].

The dataset consisting of the samples and their labels is the input to this phase. Let $B_1 \ldots, B_R$ denote the second

phase learning algorithms and B denote the current phase learning algorithm. For all the rounds $r = 1 \ldots, R$, a second phase individual learner d_r using the second phase learning algorithm b_r and applying it to the original dataset d, so mathematically speaking $d_r = b_s(d)$. A new dataset $d' = \emptyset$ is generated from the previous process. For all the new training samples $n = 1 \ldots, i$ and for all the new rounds $r = 1 \ldots, R$, classify the training instance s_i by utilizing d_r (i.e., z_{nr}) to produce the following dataset.

$$d' = d' \cup \left\{\left(\left(z_{n1}, z_{n2}, \ldots, z_{nR}\right), t_n\right)\right\} \tag{15}$$

After the new dataset is completely generated, the current phase learner c' is trained utilizing the current phase learning algorithm B and the newly produced dataset d′; thus, $c' = B(d')$. This phase generates the following output:

$$f(s) = c'\left(c_1(s) \ldots, c_R(s)\right) \tag{16}$$

IV. EXPERIMENTAL SETTINGS

The device that is used to evaluate the proposed method is a laptop equipped with a Windows 10 operating system, i7 central processing unit (CPU), and 32 gigabyte (GB) random access memory (RAM). The Anaconda distribution and Python programming language are used to utilize the well-known machine learning library (i.e., Scikit-learn).

To validate the generalization of the proposed method, a recent dataset [43] is used, which is one of the only two TMD sensor-based public datasets that have been found. This dataset consists of 5,894 trips representing five modes of transportation. As shown in Table 1, this dataset is favored because it is more thorough, unlike the other dataset [44] that captures sensor measurements from three users using one device model (i.e., Huawei Mate 9). This dataset collects sensor measurements from thirteen different users (three female and ten male) with diverse genders, occupations, Android versions, device models, and ages using the Android app. There are no restrictions imposed by the creators of the dataset on the use of the Android app that captures the mode of transportation. Therefore, each user records the sensor measurements when an activity is performed to simulate real-world situations. Each user assigns an action to perform to record the sensor measurements generated by that action. The modes performed by these individuals include standing still, walking, riding in a car, riding a bus, and riding a train. The dataset has been constructed to follow general practices taken by authors of other datasets in the TMD area [12].

The sensor events are then gathered by an application that sets a maximum frequency of 20 hertz (Hz). The maximum sampling rate of 20 Hz is a decent choice because the frequency elements of body motions are measured and recognized to be lower than 20 Hz [45], [46]. Furthermore, the vibration frequency of the vehicle seat while the engine is on ranges between 3 and 5 Hz, even when the vehicle is not in motion [47]. The subjects using this application were asked to perform the five activities and were authorized to start and stop capturing measurements when performing one of these activities, enter their names, and label the performed activities. Some information is provided once the change in the measured parameters occurs, such as the sensor's name, sensor's raw data, and timestamp. Each measurement is stored in a comma-separated values(CSV) file; thus, the dataset contains 226 CSV files (i.e., a total of 31 hours of recording, as shown in Table 2) representing the actions taken by the individuals performing the five activities. The distribution of these activities in the dataset is as follows: walking activities correspond to 26% of the dataset, driving car activities correspond to 25%, standing still class records represent 24%, riding a train represents 20% of the dataset, and only 5% represents measurements of individuals riding a bus.

TABLE 1. Dataset consisting of sensor measurements collected from thirteen different subjects with diverse genders, occupations, device models, and ages.

ID	Gender	Occupation	Age	Device Model
1	M	student	30	LG G2
2	F	student	27	Sony XPERIA Z3 Compact
3	M	student	30	Nexus 5
4	M	office worker	36	Huawei Honor 5X
5	M	stage director	36	Huawei P8 Lite
6	M	researcher	27	Samsung Galaxy S3 Neo
7	M	photographer	32	Samsung S7
8	F	bartender	32	Huawei Tag-101
9	F	student	24	Motorola Moto G
10	M	student	22	Huawei P9
11	F	office worker	31	Nexus 5
12	M	researcher	31	Samsung Galaxy S6
13	M	retired	60	Nexus 5

TABLE 2. A total of 31 hours of recording, representing the actions taken by the individuals performing the five activities.

Activity	Measurement time	Activity	Measurement time
Bus	1:44:35	Car	7:53:50
Train	6:20:25	Walk	8:20:25
Still	8:20:25	**Total**	**31:48:50**

The collected dataset is a combination of measurements captured by all available twenty-three sensors in smartphones. Among these twenty-three available sensors, fifteen sensors are supported by enough smartphones, so these sensor measurements are retained for further analysis. Due to bias measurements captured by some sensors, six sensors were eliminated from the dataset, including pressure, magnetic field, uncalibrated magnetic field, proximity, gravity, and light. The remaining nine sensors are used to construct the dataset, including rotation vector sensors, game rotation vector, accelerometers, uncalibrated gyroscopes, linear acceleration, orientation, speed, sound, and gyroscopes. Measurements generated by speed and sound sensors are converted to positive values.

The dataset was divided into time windows. The size of the time window was set to 5 seconds by the creators of the dataset for two reasons: to capture the action being performed and following other researchers' suggestions [12], [48]–[50] in which a 5-second window was the perfect candidate to

TABLE 3. The first dataset contins 12 features, the second dataset encompasses 32 features, and the third dataset comprises 36 features. Each dataset consists of 5,894 samples which are measured by smartphones carried by 13 subjects, accumulated of 31 hours of recording, and sampled at a maximum frequency of 20 Hz. Note: maximum is denoted by max, min denote minimum, μ denote mean, and standard deviation is denoted by σ.

(a) The first dataset features.

Sensors	Features
gyroscope	max, min, μ, and σ
accelerometer	max, min, μ, and σ
sound	max, min, μ, and σ

(b) The second dataset features.

Sensors	Features
gyroscope	max, min, μ, and σ
uncalibrated gyroscope	max, min, μ, and σ
game rotation vector	max, min, μ, and σ
rotation vector	max, min, μ, and σ
linear acceleration	max, min, μ, and σ
orientation	max, min, μ, and σ
accelerometer	max, min, μ, and σ
sound	max, min, μ, and σ

(c) The second dataset features.

Sensors	Features
gyroscope	max, min, μ, and σ
uncalibrated gyroscope	max, min, μ, and σ
game rotation vector	max, min, μ, and σ
rotation vector	max, min, μ, and σ
linear acceleration	max, min, μ, and σ
orientation	max, min, μ, and σ
accelerometer	max, min, μ, and σ
sound	max, min, μ, and σ
speed	max, min, μ, and σ

measure the whole action being performed by a specific subject. Therefore, setting a small window size results in misclassification because some modes require a certain amount of time that is greater than 3 seconds (e.g., to reveal the activity characteristics of walking, at least 3 seconds of accumulated data is needed). Correspondingly, setting a large window size introduces noise from various other activities interfering with the targeted mode [45]. For the nine sensors, the maximum, the minimum, the mean, and the standard deviation have been calculated for each window size and inserted into three datasets that represent the features (i.e., columns in the CSV files). Thus, for each sample (i.e., 5-second window size), a total of 4 features for each sensor was constructed. Moreover, the missing values were averaged with the values of the given feature.

Furthermore, since the number of samples measured by the sensors representing bus activity is much lower than those of the rest of the activities, as shown in Table 2, all the measurements have been lowered to match the bus measurements (i.e., to balance the samples belonging to each class). The sensor measurements construct three benchmark datasets for evaluation purposes. The number of samples for the dataset is 5,894, representing the activities performed (i.e., the journeys taken) by the 13 subjects. As shown in Table 3, the first dataset consists of 12 features in the form of maximum, minimum, mean, and standard deviation calculations, exemplifying the measurements captured by three sensors, namely, the gyroscope, accelerometer, and sound. The second dataset contains 32 features representing the measurements captured by the rest of the sensors, with the exception of speed (i.e., the gyroscope, uncalibrated gyroscope, game rotation vector, rotation vector, linear acceleration, orientation, accelerometer, and sound). The last dataset consists of 36 features representing all the sensor measurement data, including those captured by the speed sensor.

V. PARAMETER OPTIMIZATION

To improve the performance of the proposed method in terms of the four evaluation measures while maintaining decent time complexity, the parameters of the base estimators were optimized. To reduce the complexity of the ensemble methods such as RFs, GB, and bagging that rely on DT to build their models, the default values such as 2 minimum number of samples split, 1 minimum number of sample leafs, and Gini criterion function (i.e., responsible for measuring the split's quality) were utilized. The important hyperparameters of the other base estimators were chosen based on grid search. Among (10, 20, 30, 40, 50, 60, 70, 80, 90, and 100) trees in RFs, GB, and bagging, 10, 100, and 10 were selected, respectively.

In the second phase of the proposed method, grid search utility was utilized to select the most valuable KNNs algorithm options to determine the mode of transportation from three hyperparameters, namely, metrics of similarity, number of neighbors, and the weight function. From three metrics of similarity, namely, Euclidean, Manhattan, and Minkowski, the latest was chosen. Five neighbors were investigated (3, 5, 7, 9, and 11) to optimize this parameter, in which 3 was the best fit for the data. The weight function was the last explored parameter with two options, namely, uniform or distance, in which uniform was the chosen function. The purpose of the uniform function is to equalize the weight for the data points in each neighborhood, while the distance function gives closer neighbors of the query data point more influence than the rest of the data points.

In the last phase of the proposed method, four MLP hyperparameters were considered, namely, the number of hidden layers, the number of neurons in each hidden layer, the activation function, and the optimizer. In addition to one input layer and one output layer, two hidden layers were used to minimize the time complexity. The first layer consists of 80 neurons, while the second layer consists of 10 neurons. For the hidden layers, identity, logistic, hyperbolic tangent (i.e., tanh), and ReLU activation functions were tested, in which hyperbolic tangent performed better than the other three functions. From the two tested weight optimizers (Adam and stochastic gradient descent), Adam was chosen because it works very well with datasets that have a sufficient number of samples (i.e., thousands of samples) such as the TMD dataset.

VI. RESULTS

This section presents the results of the proposed method along with the results of the different machine learning algorithms that have been tested in the experiments. The performance

TABLE 4. Evaluation measures of all algorithms on the three datasets using two different data splits sorted by accuracy in descending order.

	10% testing data split											
	1st Dataset				2nd Dataset				3rd Dataset			
Method	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Proposed	**89.0%**	**89.1%**	**89.4%**	**89.0%**	**93.2%**	**93.2%**	**93.5%**	**93.3%**	**95.6%**	**95.5%**	**95.7%**	**95.6%**
RFs	86.3%	86.3%	86.7%	86.4%	90.8%	90.9%	91.1%	90.9%	95.1%	95.0%	95.2%	95.0%
Bagging	85.6%	85.7%	86.0%	85.7%	92.4%	92.5%	92.5%	92.4%	93.7%	93.7%	94.0%	93.8%
GB	84.9%	85.2%	85.3%	85.2%	90.0%	90.2%	90.2%	90.1%	93.9%	93.9%	94.1%	93.9%
XGB	82.4%	82.7%	82.8%	82.6%	86.9%	87.2%	87.2%	87.1%	93.1%	93.1%	93.3%	93.1%
AdaBoost	79.2%	79.4%	79.6%	79.3%	83.6%	83.4%	83.7%	83.5%	92.5%	92.6%	92.6%	92.6%
DT	78.6%	78.8%	78.9%	78.8%	83.7%	83.5%	83.8%	83.6%	91.9%	91.8%	92.0%	91.9%
LSVM	66.6%	67.5%	67.2%	66.4%	74.1%	74.5%	74.2%	73.9%	76.8%	76.8%	76.6%	76.5%
NLSVM	63.9%	63.6%	64.1%	63.3%	71.2%	82.3%	71.1%	73.0%	73.2%	86.3%	73.1%	75.6%
MLP	62.9%	65.4%	63.8%	62.7%	70.2%	76.1%	71.1%	69.9%	84.4%	84.5%	83.9%	83.9%
KNNs	60.0%	61.2%	60.8%	59.8%	77.5%	77.8%	78.4%	77.6%	83.6%	83.6%	84.0%	83.5%
LR	59.2%	60.1%	59.6%	59.0%	71.2%	71.5%	71.2%	71.1%	70.3%	70.7%	70.3%	70.1%
GNB	55.3%	58.9%	55.5%	53.6%	57.3%	60.4%	57.3%	56.5%	52.4%	55.6%	52.5%	50.1%
	30% testing data split											
	1st Dataset				2nd Dataset				3rd Dataset			
Method	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Proposed	**86.8%**	**86.8%**	**86.9%**	**86.8%**	**92.1%**	**92.1%**	**92.1%**	**92.1%**	**94.9%**	**94.9%**	**94.9%**	**94.8%**
GB	84.1%	84.1%	84.2%	84.1%	90.4%	90.5%	90.4%	90.4%	92.9%	92.9%	92.9%	92.9%
RFs	83.9%	83.9%	84.1%	83.9%	87.8%	87.8%	87.9%	87.8%	92.8%	92.8%	92.9%	92.8%
Bagging	83.0%	83.0%	83.1%	83.0%	88.6%	88.6%	88.6%	88.6%	92.0%	92.0%	92.1%	92.0%
XGB	80.4%	80.5%	80.4%	80.4%	87.9%	87.9%	88.0%	87.9%	92.1%	92.2%	92.1%	92.1%
AdaBoost	77.2%	77.3%	77.3%	77.3%	82.9%	82.8%	82.9%	82.8%	89.8%	89.7%	89.8%	89.8%
DT	76.9%	77.0%	77.0%	77.0%	82.2%	82.1%	82.2%	82.1%	89.8%	89.7%	89.8%	89.8%
MLP	67.6%	68.7%	67.8%	67.8%	67.3%	72.8%	67.9%	66.1%	77.9%	80.0%	78.1%	78.0%
LSVM	65.0%	65.7%	65.1%	64.9%	72.3%	72.6%	72.5%	72.2%	76.0%	76.2%	76.2%	76.0%
NLSVM	63.0%	63.3%	63.2%	62.8%	66.3%	79.1%	66.1%	67.8%	68.3%	84.7%	68.1%	70.7%
LR	59.6%	60.2%	59.8%	59.4%	67.0%	67.2%	67.2%	66.9%	68.8%	68.8%	69.0%	68.7%
KNNs	58.9%	60.1%	59.2%	58.9%	73.9%	74.3%	74.2%	73.8%	80.9%	81.2%	81.1%	80.8%
GNB	56.1%	59.4%	55.4%	45.2%	58.6%	61.5%	58.1%	58.2%	54.9%	57.8%	55.6%	52.9%

of the tested methods along with the proposed technique were evaluated using four measures: precision, recall, f1-score, and accuracy. The precision is calculated as $\frac{TP}{FP+TP}$, the recall is calculated as $\frac{TP}{TP+FN}$, the f1-score is calculated as $2 \cdot \frac{precision \cdot recall}{precision + recall}$, and the accuracy is calculated as $\frac{TP+TN}{TP+TN+FP+FN}$.

In the first set of experiments, each of the three datasets was split into 90% training and 10% testing. The proposed method was evaluated and compared with the other machine algorithms using four evaluation metrics, namely, accuracy, precision, recall, and F1-score, as shown in Table 4. The proposed method outperformed the rest of the methods when tested on the first dataset; its accuracy is 89%, which is better than the second-best method (i.e., RFs) by approximately 3%. It is also better than the other methods when tested on the second dataset; its accuracy is 93.2%, which is better than the second-best method by 2.4%. Finally, the performance of the proposed method is similar to that of the second-best method on the third dataset; it is better than the second-best method by only 0.5%. In the second set of experiments, each of the three datasets was split into 70% training and 30% testing. As shown in Table 4, the proposed method outperformed the rest of the methods on the three datasets. The accuracy of the proposed method was better than the second-best method (i.e., GB) by approximately 3%, 2%, and 2% on the first, second, and third datasets, respectively.

The details of each correctly classified class of samples and misclassified samples are shown in Table 5 in the form of a confusion matrix. In the first set of experiments (when the first dataset was split into 90% training and 10% testing), the still class samples were correctly classified 96% of the time, the bus class samples were correctly classified approximately 91% of the time, the car class samples were correctly classified approximately 78% of the time, the train class samples were correctly classified approximately 90% of the time, and the walking class samples were correctly classified 91.5% of the time. In the second dataset (using the same data split, i.e., 90% training and 10% testing), the still and bus class samples were correctly classified approximately 97% of the time, the car class samples were correctly classified approximately 90% of the time, the train class instances were correctly classified 91% of the time, and the walking class samples were correctly classified 92% of the time. In the third dataset, the still class samples were correctly classified 98% of the time, the bus class samples were correctly classified approximately 97% of the time, the car class samples were correctly classified approximately 93% of the time, the train

TABLE 5. Confusion matrices for the stacked proposed method.

10% Testing Data Split Predicted Class

Actual Class	1st Dataset					2nd Dataset					3rd Dataset				
	Still	Bus	Car	Train	Walking	Still	Bus	Car	Train	Walking	Still	Bus	Car	Train	Walking
Still	**99**	0	1	1	2	**100**	0	2	0	1	**101**	0	1	0	1
Bus	1	**99**	2	6	1	0	**106**	1	1	1	0	**106**	1	1	1
Car	2	13	**105**	12	2	1	7	**121**	4	1	0	4	**125**	3	2
Train	2	4	4	**103**	1	3	3	4	**104**	0	3	3	0	**108**	0
Walking	4	4	1	2	**119**	2	5	1	3	**119**	2	3	0	1	**124**

30% Testing Data Split Predicted Class

Actual Class	1st Dataset					2nd Dataset					3rd Dataset				
	Still	Bus	Car	Train	Walking	Still	Bus	Car	Train	Walking	Still	Bus	Car	Train	Walking
Still	**310**	6	3	4	6	**309**	3	8	5	4	**319**	2	1	3	4
Bus	2	**301**	14	15	11	1	**331**	3	3	5	2	**328**	2	5	6
Car	7	40	**273**	27	5	10	16	**306**	15	5	5	14	**317**	10	6
Train	17	15	32	**309**	2	13	7	17	**337**	1	9	4	1	**361**	0
Walking	9	8	5	5	**342**	6	10	3	5	**345**	5	7	1	4	**352**

class samples were correctly classified approximately 95% of the time, and the walking class samples were correctly classified 95% of the time.

In the second set of experiments (when the first dataset was split into 70% training and 30% testing), the still class samples were correctly classified 94% of the time, the bus class samples were correctly classified approximately 88% of the time, the car class samples were correctly classified approximately 78% of the time, the train class samples were correctly classified approximately 82% of the time, and the walking class samples were correctly classified 93% of the time. In the second dataset (using the same data split, i.e., 70% training and 30% testing), the still class samples were correctly classified approximately 94% of the time, the bus class samples were correctly classified approximately 97% of the time, the car class samples were correctly classified approximately 87% of the time, the train class samples were correctly classified 90% of the time, and walking class samples were correctly classified 93% of the time. In the third dataset, the still class samples were correctly classified 97% of the time, the bus and train class samples have been correctly classified approximately 96% of the time, the car class samples were correctly classified approximately 90% of the time, and the walking class samples were correctly classified 95% of the time.

The training and testing times are shown in Table 6. The training time of the proposed method was adequate compared to the other methods, while the testing time was lower compared to the rest of the methods. In the first set of experiments where the dataset was split into 90% training and 10% testing, the training time of the proposed method using the first dataset was approximately 4.4 *s*, approximately 4.5 *s* for the second dataset, and approximately 4.4 *s* for the third dataset. The testing time of the proposed method was 1 *ms* when applied to the first dataset, 2 *ms* when applied to the second dataset, and 2 *ms* when applied to the third dataset. In the second set of experiments where the dataset was split into 70% training and 30% testing, the training times of the proposed method were 3.2 *s*, 3.3 *s*, and 3.5 *s* on the first, second, and third datasets, respectively. The testing times of the proposed method using the three datasets were 4 *ms* for the three datasets.

VII. DISCUSSION

The accuracy of our method outperformed the other methods on the three datasets using two data splits (i.e., 70% training–30% testing data split and 90% training–10% testing data split). The second-best method in terms of accuracy on the first dataset was RFs. The second-best method on the second dataset was GB. The effect of the data split and sensors involved in gathering sensor measurements from smartphone devices equipped with sensors carried by the subjects in experiments is shown in Fig. 2.

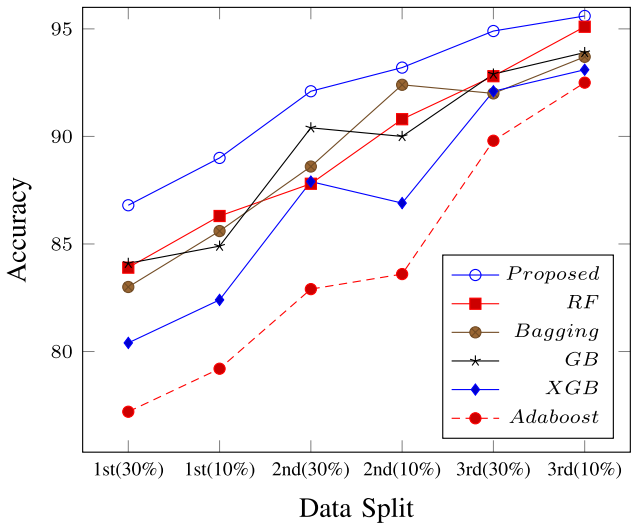


FIGURE 2. Two data splits were used to measure the performance of the algorithms. Moreover, the effect of the sensors involved in the three datasets was investigated.

It is obvious from the figure that as the training data increased (increasing the training from 70% to 90%), the accuracy of all the algorithms increased as well. Moreover, the number of sensors integrated with smartphones

TABLE 6. Training and testing times of all algorithms on the three datasets using two different data splits.

	10% testing data split						30% testing data split					
	1st Dataset		2nd Dataset		3rd Dataset		1st Dataset		2nd Dataset		3rd Dataset	
Method	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Proposed	4.372	0.001	4.496	0.002	4.443	0.001	3.194	0.004	3.330	0.004	3.508	0.004
RFs	0.093	0.001	0.143	0.002	0.151	0.002	2.070	0.027	4.436	0.028	4.580	0.024
Bagging	0.331	0.002	0.823	0.003	0.870	0.003	0.072	0.003	0.104	0.003	0.120	0.003
GB	2.492	0.005	5.412	0.005	5.635	0.005	0.245	0.004	0.596	0.004	0.614	0.005
XGB	1.563	0.007	3.578	0.008	3.713	0.008	1.254	0.022	2.835	0.023	2.926	0.022
AdaBoost	0.058	0.001	0.147	0.001	0.155	0.000	0.043	0.001	0.109	0.001	0.114	0.001
DT	0.056	0.000	0.144	0.000	0.150	0.001	0.043	0.001	0.116	0.000	0.118	0.001
LSVM	149.5	0.029	364.9	0.055	281.5	0.054	3.633	0.004	3.700	0.004	2.899	0.004
NLSVM	2.559	0.144	4.109	0.222	0.151	0.002	94.02	0.066	283.9	0.125	210.3	0.126
MLP	3.547	0.001	4.975	0.001	4.972	0.002	1.539	0.342	2.407	0.533	2.518	0.550
KNNs	0.004	0.001	0.011	0.026	0.009	0.028	0.133	0.000	0.481	0.001	0.475	0.000
LR	0.171	0.000	0.627	0.000	0.728	0.001	0.004	0.049	0.007	0.076	0.009	0.081
GNB	0.002	0.001	0.004	0.001	0.004	0.000	1.254	0.022	0.003	0.002	0.004	0.002

affects the performance. As shown in the figure, the first dataset contains measurements captured by three sensors: sound, accelerometer, and gyroscope. Due to the limited sensors involved in the dataset, the performance of all the algorithms did not exceed 90% (the most accurate algorithm was the proposed method with 90% accuracy when applied to a 10% testing data split). As the number of sensors increased, the results of all the algorithms increased as can be seen when the second dataset was used, which consisted of measurements captured by an accelerometer, game rotation vector, gyroscope, gyroscope uncalibrated, linear acceleration, orientation, rotation vector, and sound. The most accurate technique was the proposed method (its accuracy was 93.2% when applied to a 10% testing data split). Additionally, the accuracy of all the algorithms increased when the remaining sensor measurements (i.e., speed sensor) were involved in the third dataset. The best method remained the proposed method with an accuracy of 95.6%.

Since RFs is the second-most accurate method in the first set of experiments (after the proposed method) and the third-most accurate method in the second set of experiments, the feature importance capability provided by it was utilized to conceive the features that are considered most important in distinguishing the mode of transportation. The second set of experiments was used to screen the most important features, where the data were split into 70% training and 30% testing. Because the third dataset contains all the features (i.e., a total of 36 features), it was used. The ten most important features in distinguishing the mode of transportation applied to the third dataset on the second set of experiments are shown in Table 7.

The training time and testing time of all the algorithms were decent. Although the training time of our algorithm was not the best compared to the other algorithms, the training time was not as important as the testing time because the training was performed only once. The testing time of the proposed method was short. The number of measurements was 5,894, and the testing time for 10% (i.e., approximately 589 measurements) of the data was 4 *ms* when the proposed method was applied to the third dataset, so the prediction

TABLE 7. The ten most important features sorted by importance in descending order.

Index	Feature	Importance
1	σ of accelerometer measurement	0.1314
2	min of speed measurement	0.0685
3	μ of speed measurement	0.0535
4	μ of sound measurement	0.0445
5	μ of gyroscope measurement	0.0436
6	max of speed measurement	0.0414
7	max of linear acceleration measurement	0.0408
8	max of gyroscope measurement	0.0397
9	min of sound measurement	0.0364
10	σ of sound measurement	0.0297

time for one measurement was approximately 7 microseconds (i.e., the result of the division of the testing time and the number of measurements in the test set).

Obviously, the accuracy of transportation mode detection increases as the measurements of more sensors are involved in the detection process. However, increasing the number of sensors might increase the complexity of the detection method (i.e., the time required to detect the transportation mode) and raise some privacy concerns. Although the proposed method achieved superior results compared to the tested machine learning methods and can detect the mode of transportation in a reasonable time, there is room for improvement, which could be investigated further in future work. A variety of lower sampling rates, such as 1 Hz, could be utilized in future work, as suggested by some GPS-based techniques [51] to improve the detection rate of transpiration mode. In addition, the geographic zones taken by participants while performing the transportation mode should be recorded and included in future work to clarify the route taken by participants. Moreover, various window sizes should be studied in future work (e.g., a window size of 1 minute has been suggested by some GPS-based approaches [**?**], [52]).

VIII. CONCLUSION

Context-aware computing is a significant area that has been accomplished in the last two decades with the help of

smartphone devices equipped with sensors. A variety of services ranging from smart parking to vehicular traffic monitoring can be achieved utilizing the awareness of user mobility. Detecting the mode of transportation with the help of machine learning has gained researchers' attention due to the various services that can help accomplish this. In this article, a stacked learning technique was proposed to detect the mode of transportation, such as riding a train, driving a car, standing, walking, and riding a bus, utilizing three datasets that were constructed to measure the activity performed by different subjects using smartphones equipped with various sensors. Extensive sets of experiments that involve 12 machine learning algorithms were conducted to validate the proposed method. The proposed method outperformed the rest of the methods in terms of accuracy on the three datasets with two different data splits. The prediction time of each measurement was so low that it is an ideal option to detect the mode of transportation in real time. In the first set of experiments where the dataset was split into 10% testing and 90% training, the proposed method yielded accuracies of 89%, 93%, and 95% on the first, second, and third datasets, respectively. In the second set of experiments where the dataset was split into 30% testing and 70% training, the proposed technique achieved accuracies of approximately 87%, 92%, and 95% on the first, second, and third datasets, respectively. The high accuracy achieved by the proposed method indicates that the size of the dataset used to evaluate the proposed method is sufficient.

REFERENCES

[1] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proc. 1st Workshop Mobile Comput. Syst. Appl.*, Piscataway, NJ, USA, Dec. 1994, pp. 85–90.

[2] P. Castrogiovanni, E. Fadda, G. Perboli, and A. Rizzo, "Smartphone data classification technique for detecting the usage of public or private transportation modes," *IEEE Access*, vol. 8, pp. 58377–58391, 2020.

[3] C. Carpineti, V. Lomonaco, L. Bedogni, M. D. Felice, and L. Bononi, "Custom dual transportation mode detection by smartphone devices exploiting sensor diversity," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Piscataway, NJ, USA, Mar. 2018, pp. 367–372.

[4] A. Jahangiri and H. A. Rakha, "Applying machine learning techniques to transportation mode recognition using mobile phone sensor data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2406–2417, Oct. 2015.

[5] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. Newslett.*, vol. 12, no. 2, pp. 74–82, Mar. 2011.

[6] J. Biancat, C. Brighenti, and A. Brighenti, "Review of transportation mode detection techniques," *EAI Endorsed Trans. Ambient Syst.*, vol. 1, no. 4, p. e7, Oct. 2014.

[7] Y. Qin, H. Luo, F. Zhao, C. Wang, J. Wang, and Y. Zhang, "Toward transportation mode recognition using deep convolutional and long short-term memory recurrent neural networks," *IEEE Access*, vol. 7, pp. 142353–142367, 2019.

[8] Y. Xiao, D. Low, T. Bandara, P. Pathak, H. Beng Lim, D. Goyal, J. Santos, C. Cottrill, F. Pereira, C. Zegras, and M. Ben-Akiva, "Transportation activity analysis using smartphones," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Piscataway, NJ, USA, Jan. 2012, pp. 60–61.

[9] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu, "Transportation mode detection using mobile phones and GIS information," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, New York, NY, USA, 2011, pp. 54–63.

[10] P. Widhalm, P. Nitsche, and N. Brändie, "Transport mode detection with realistic smartphone sensor data," in *Proc. 21st Int. Conf. Pattern Recognit. (ICPR)*, Piscataway, NJ, USA, 2012, pp. 573–576.

[11] K. Lee and M.-P. Kwan, "Physical activity classification in free-living conditions using smartphone accelerometer data and exploration of predicted results," *Comput., Environ. Urban Syst.*, vol. 67, pp. 124–131, Jan. 2018.

[12] L. Bedogni, M. Di Felice, and L. Bononi, "Context-aware Android applications through transportation mode detection techniques," *Wireless Commun. Mobile Comput.*, vol. 16, no. 16, pp. 2523–2541, Nov. 2016.

[13] S. Reddym, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Trans. Sensor Netw.*, vol. 6, no. 2, p. 13, Mar. 2010.

[14] T. Feng and H. J. P. Timmermans, "Transportation mode recognition using GPS and accelerometer data," *Transp. Res. C, Emerg. Technol.*, vol. 37, pp. 118–130, Dec. 2013.

[15] X. Su, H. Caceres, H. Tong, and Q. He, "Online travel mode identification using smartphones with battery saving considerations," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2921–2934, Oct. 2016.

[16] S. Bhattacharya and N. D. Lane, "From smart to deep: Robust activity recognition on smartwatches using deep learning," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Piscataway, NJ, USA, Mar. 2016, pp. 1–6.

[17] L. Wang, H. Gjoreski, M. Ciliberto, S. Mekki, S. Valentin, and D. Roggen, "Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset," *IEEE Access*, vol. 7, pp. 10870–10891, 2019.

[18] J.-G. Krieg, G. Jakllari, H. Toma, and A.-L. Beylot, "Unlocking the smartphone's sensors for smart city parking," *Pervas. Mobile Comput.*, vol. 43, pp. 78–95, Jan. 2018.

[19] Z. Ma, P. Wang, Z. Gao, R. Wang, and K. Khalighi, "Ensemble of machine learning algorithms using the stacked generalization approach to estimate the Warfarin dose," *PLoS ONE*, vol. 13, no. 10, pp. 1–12, 2018.

[20] H. Huang, Y. Cheng, and R. Weibel, "Transport mode detection based on mobile phone network data: A systematic review," *Transp. Res. C, Emerg. Technol.*, vol. 101, pp. 297–312, Apr. 2019.

[21] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proc. 11th ACM Conf. Embedded Netw. Sensor Syst.*, New York, NY, USA, 2013, p. 13.

[22] B. Wang, L. Gao, and Z. Juan, "Travel mode detection using GPS data and socioeconomic attributes based on a random forest classifier," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1547–1558, May 2018.

[23] J. Urbancic, V. Pejovic, and D. Mladenic, "Transportation mode detection using random forest," in *Proc. Inf. Soc., Data Mining Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2018, pp. 1–4.

[24] M. A. Shafique and E. Hato, "Classification of travel data with multiple sensor information using random forest," *Transp. Res. Procedia*, vol. 22, pp. 144–153, Jan. 2017.

[25] Z. Xiao, Y. Wang, K. Fu, and F. Wu, "Identifying different transportation modes from trajectory data using tree-based ensemble classifiers," *ISPRS Int. J. Geo-Inf.*, vol. 6, no. 2, p. 57, Feb. 2017.

[26] Y. Zheng, H. Fu, X. Xie, W. Y. Ma, and Q. Li. (2011). Geolife GPS Trajectory Dataset-User Guide. Microsoft Research. [Online]. Available: https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/

[27] S.-H. Fang, H.-H. Liao, Y.-X. Fei, K.-H. Chen, J.-W. Huang, Y.-D. Lu, and Y. Tsao, "Transportation modes classification using sensors on smartphones," *Sensors*, vol. 16, no. 8, p. 1324, Aug. 2016.

[28] M. A. Shafique and E. Hato, "Use of acceleration data for transportation mode prediction," *Transportation*, vol. 42, no. 1, pp. 163–188, Jan. 2015.

[29] S.-H. Fang, Y.-X. Fei, Z. Xu, and Y. Tsao, "Learning transportation modes from smartphone sensors based on deep neural network," *IEEE Sensors J.*, vol. 17, no. 18, pp. 6111–6118, Sep. 2017.

[30] X. Liang and G. Wang, "A convolutional neural network for transportation mode detection based on smartphone platform," in *Proc. IEEE 14th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2017, pp. 338–342.

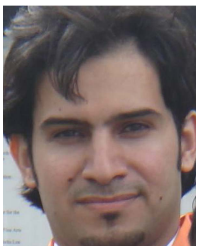
[31] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification Regression Trees*. Boca Roton, FL, USA: CRC Press, 1984.

[32] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[33] J. H. Friedman, "Stochastic gradient boosting," *Comput. Statist. Data Anal.*, vol. 38, no. 4, pp. 367–378, Feb. 2002.

[34] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[35] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.
[36] S. C. Kleene, "Representation of events in nerve nets and finite automata," RAND Project Air Force, Santa Monica, CA, USA, Tech. Rep. RAND-RM-704, 1951.
[37] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Proc. Eur. Conf. Comput. Learn. Theory*. Berlin, Germany: Springer, 1995, pp. 23–37.
[38] B. Boehmke and B. M. Greenwell, "Gradient Boosting," in *Hands-On Machine Learning With R*. Boca Raton, FL, USA: CRC Press, 2019, pp. 221–223.
[39] R. Polikar, "Ensemble learning," in *Ensemble Machine Learning*. Berlin, Germany: Springer, 2012, pp. 1–34.
[40] S. Raschka, "Stat 479: Machine learning lecture notes," Dept. Statist., Algorithm Comparison, Univ. Wisconsin–Madison, Madison, WI, USA, Tech. Rep. STAT479 FS19. L11, 2018.
[41] Z. H. Zhou, "Ensemble learning," *Encyclopedia Biometrics*, vol. 1, pp. 270–273, Jun. 2009.
[42] S. Haykin, "Multilayer perceptron," in *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1994, pp. 178–191.
[43] M. D. Felice, L. Bononi, L. Bedogni, and V. Lomonaco. (2017). *TMD Dataset*. University of Bologna. [Online]. Available: http://cs.unibo.it/projects/us-tm2017/index.html
[44] H. Gjoreski, M. Ciliberto, L. Wang, F. J. Ordonez Morales, S. Mekki, S. Valentin, and D. Roggen, "The university of Sussex-Huawei locomotion and transportation dataset for multimodal analytics with mobile devices," *IEEE Access*, vol. 6, pp. 42592–42604, 2018.
[45] H. Xia, Y. Qiao, J. Jian, and Y. Chang, "Using smart phone sensors to detect transportation modes," *Sensors*, vol. 14, no. 11, pp. 20843–20865, Nov. 2014.
[46] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE Trans. Inf. Technol. Biomed.*, vol. 10, no. 1, pp. 156–167, Jan. 2006.
[47] K. Li, M. Lu, F. Lu, Q. Lv, L. Shang, and D. Maksimovic, "Personalized driving behavior monitoring and analysis for emerging hybrid vehicles," in *Proc. Int. Conf. Pervas. Comput.* Berlin, Germany: Springer, 2012, pp. 1–19.
[48] D. Shin, D. Aliaga, B. Tunçer, S. M. Arisona, S. Kim, D. Zünd, and G. Schmitt, "Urban sensing: Using smartphones for transportation mode classification," *Comput., Environ. Urban Syst.*, vol. 53, pp. 76–86, Sep. 2015.
[49] B. Nham, K. Siangliulue, and S. Yeung, "Predicting mode of transport from iPhone accelerometer data," Mach. Learn. Final Projects, Stanford Univ. Class Project, Stanford, CA, USA, Tech. Rep. CS 229, 2008.
[50] P. Nitsche, P. Widhalm, S. Breuss, and P. Maurer, "A strategy on how to utilize smartphones for automatically reconstructing trips in travel surveys," *Procedia-Social Behav. Sci.*, vol. 48, pp. 1033–1046, 2012.
[51] O. Burkhard, H. Becker, R. Weibel, and K. W. Axhausen, "On the requirements on spatial accuracy and sampling rate for transport mode detection in view of a shift to passive signalling data," *Transp. Res. C, Emerg. Technol.*, vol. 114, pp. 99–117, May 2020.
[52] K. Ellis, S. Godbole, S. Marshall, G. Lanckriet, J. Staudenmayer, and J. Kerr, "Identifying active travel behaviors in challenging environments using GPS, accelerometers, and machine learning algorithms," *Frontiers Public Health*, vol. 2, p. 36, Apr. 2014.
[53] J. A. Carlson, M. M. Jankowska, K. Meseck, S. Godbole, L. Natarajan, F. Raab, B. Demchak, K. Patrick, and J. Kerr, "Validity of PALMS GPS scoring of active and passive travel compared to SenseCam," *Med. Sci. Sports Exerc.*, vol. 47, no. 3, p. 662, 2015.



BANDAR ALOTAIBI (Member, IEEE) received the B.Sc. degree (Hons.) in computer science-information security and assurance emphasis from The University of Findlay, USA, the M.Sc. degree in information security and assurance from Robert Morris University, USA, and the Ph.D. degree in computer science and engineering from the University of Bridgeport, USA. He is currently an Assistant Professor with the Department of Information Technology, University of Tabuk. His research interests include computer vision, network security, mobile communications, computer forensics, wireless sensor networks, and quantum computing.