Transportation Research Part C 121 (2020) 102834

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Stop detection for smartphone-based travel surveys using geo-spatial context and artificial neural networks

Valentino Servizi *, Niklas C. Petersen, Francisco C. Pereira, Otto A. Nielsen

Department of Technology, Management and Economics, Technical University of Denmark (DTU), Kgs. Lyngby, Denmark

ARTICLE INFO

Keywords:
GPS + GIS fusion
Stop detection
Smartphone based travel surveys
ANN
CNN
RNN
Point based classification
GPS
Trajectories

ABSTRACT

The problem of stop detection is at the base of many current and upcoming smartphone-based travel survey technologies and directly impacts the quality of many downstream operations. The inference of departure/arrival time, mode, and purpose of a trip, for example, rely on the stop/motion patterns represented by smartphone sensor-data. As users handle smartphones for various purposes and their preferences determine different device positions while traveling, accelerometer and gyroscope, for instance, often present ambiguities that prevent accurate stop detection.

To mitigate the impact of these ambiguities, we combine spatial time-series, i.e. GPS, with spatial context information retrieved from Open Street Map, which we represent as multi-dimension tensors. This project explores simple representations, such as dummy variables, and novel multidimensional representations, which are bench-marked through the classification performance of specialized artificial neural network (ANN), as well as other machine learning (ML) baselines. Our main contribution stems from this novel multidimensional representation of time-series fusion with spatial context, combined with the corresponding specialized ANN classifier. The results show a stop detection score improvement on the baselines between 3% and 6.5%.

1. Introduction

Smartphone-based travel surveys' (SBTS) study of users behavior in transport networks, relies heavily on GPS trajectories.

As people *move* for a purpose and *stop* to fulfill that purpose, we define the subset of a GPS trajectory where the user travels on any transportation mode between origin and destination, as *motion*; everything else, we define as *stop*.

Stemming from SBTS, literature on mode detection shows how motion branches out; literature on purpose imputation, how stop branches out. Both deal with trip segmentation, and contribute to the automatic generation of Travel Diaries (TDs) that are presented to users for ground truth collection, within a continuous *validation loop* (Servizi et al., 2019). In this cycle, ground truth supposedly improves machine learning algorithms for TDs generation, and vice versa.

However, in terms of GPS point density, short-duration stops, such as a pick-up, or a drop-off, are substantially different than stops with a longer duration, such as home or work stay. The same applies to motion; for example, when one moves by car versus walking.

In many cases the two categories result entangled on both time and space. Let us consider two representative examples:

* Corresponding author.
E-mail address: valse@dtu.dk (V. Servizi).

https://doi.org/10.1016/j.trc.2020.102834
Received 25 April 2020; Received in revised form 9 September 2020; Accepted 7 October 2020
Available online 3 November 2020

- Alighting from a bus is often seen as an *instant stop* since one changes transportation mode from bus to walk; nevertheless, during this transition the user is never stationary.
- During a bus trip, discontinuities at traffic lights have rarely short duration, as at each bus stop between origin and destination. However, there is no transition between transportation modes.

In addition, because of a standard deviation above 40 m (Bierlaire et al., 2013; Kim et al., 2018), GPS error itself can confuse inference, depending on building surroundings, atmospheric conditions, or relative satellite positioning, worsening the challenge of points' or segments' classification.

As (Zahabi et al., 2017) shows, these aspects could explain the severe ambiguities in GPS trajectories collected by origin–destination surveys, especially in short transfers.

Besides, the two classes of *stop* and *motion* suffer from a high imbalance since one moves only a small fraction of the day. In Denmark, for example, the average travel time measured in minutes/person/day is 57 min (Christiansen, 2006). Thus, the stop class might overwhelm any ML classifier, resulting in poor classification performance.

A wide range of classifiers is available in the SBTS field (Koushik et al., 2020). We identify two groups, point- and segment-based classifiers (Prelipcean et al., 2016). In the first case, these methods classify each GPS position; in the second, a segment composed of multiple GPS points. Among the most common trajectory segmentation methods for stop classification, rule-based methods seem very competitive (Shen and Stopher, 2014); we also find various practical clustering approaches (Aslak, 2019).

Literature has shown that existing methods suffer the entanglement of these two classes, as both thresholds definition and features engineering are not effective in catching both short stops and long discontinuities during motion. Inaccurate classification of *stops*, leads to trip over- or under-segmentation; thus, to automatic generation of inexact TDs. In case of over-segmentation, TDs will present at least one true trip-leg split in two or more classified trip-legs, while in case of under-segmentation, TDs will present at least two true trip-legs merged into one classified trip-leg.

The under-segmentation problem occurs when all the points of a stops are classified as motion; the over-segmentation, when a number of consecutive motion points are classified as stops, or viceversa. Under-segmentation bias, e.g., in correspondence of instant-stops, is critical for users' TDs validation. In the assumption that one remembers such an instant-stop and is very committed to the survey, he or she should manually add any missing stop and the activity performed within each of the corresponding space–time ranges.

Over-segmentation errors are as tedious to correct as the under-segmentation ones (Kim et al., 2018). Both these errors might lead to unacceptable ground truth (Prelipcean et al., 2016). In light of the above considerations, how can we improve the discrimination between stop and motion by processing GPS trajectories?

To succeed, many machine learning (ML) methods exploit multiple sensors, and often geographical information systems (GIS). Among the best-performers, we find support vector machines (SVM), fuzzy logic (FL), random forests (RF), and probabilistic models, e.g., hidden markov models (HMM) (Servizi et al., 2019).

In contrast, emerging methods based on ANN as in Dabiri and Heaslip (2018), Xiao et al. (2016) and Jiang et al. (2017), show classification potential due to their flexibility in learning multiple thresholds from multi-dimension tensor representations, e.g., images. However, ANN methods seldom combine GPS with GIS data, possibly with dummy variables, never through richer tensor representations, never for mode or stop detection.

In order to deal with the binary classification of GPS trajectories, including challenging short stops and long discontinuities detection in realistic datasets, we incorporate spatial context information, such as public transport network, points of interest (POI), and land use. Thus, fusing spatial features retrieved from GIS with GPS, we help better capture heterogeneous temporal and spatial thresholds through various ANN configurations.

The paper analyzes multiple ANN classifiers, specialized for point-based classification of trajectories fused on GIS data as multi-dimension tensor representations. Intuitively, these representations describe spatial-context as images; whereas dummy variables, as single-pixel images. We compare ANN against two ML classifiers. First, Infostop (Aslak, 2019), which is an unsupervised effective hybrid rule- and clustering-based method; second, a random forest, which the literature describes as one of the most effective supervised classifiers. Every classifier, except Infostop, is tested with two data representations: GPS features only, and GPS features augmented with dummy variables extracted from GIS. ANN models, as part of our contribution, allow tests with a third representation of data, which is GPS features augmented with multi-dimension-dummy variables, as tensor, extracted from GIS.

We work with a high-resolution realistic dataset which includes shared mobility trips, where users are committed in high quality ground truth collection. In Section 2, we present the literature review on GPS stop detection, and we position of our contribution within SBTS and ANN related work. In Section 3, we describe in detail our contribution in terms of methodology for fusion of GPS with GIS data, and the ANN classifiers. In Section 4 we present and discuss input data, results, validaiton process and benchmark. We conclude in Section 5, including future directions.

2. Related work

In the following sections we describe literature on stop detection method, and we pinpoint what pertains ANN.

2.1. Review of existing stop detection methods

A GPS segment is considered a stop candidate if it lays within a topologically closed polygon for a certain time (Alvares et al., 2007).

Since the introduction of GPS travel surveys, the identification of trips by stop detection has presented multiple challenges. For example, the GPS device's cold-start requires over 1 min to acquire the device position, during which this device generates erroneous data, easy to confuse with short-stops (Stopher et al., 2005). The literature, however, presents simple and effective rule-based methods and heuristics, able to classify GPS positions without any need for labels, which in contrast, are necessary for more advanced supervised machine learning methods. For instance, we could be looking at a stop (Stopher et al., 2005) each time points have (i) null speed, (ii) null or unchanged bearing, (iii) following positions that present a change below 15 meters in either latitude or longitude, (iv) and all these conditions persist for 120 s.

In smartphones, the cold-start problem has been mitigated by introducing, e.g., the Assisted-GPS (GPS), which retrieves the satellites' position from Internet, and estimates the device position triangulating the GSM signal strength received from the antennas in proximity, each time GPS satellites are not in sight (Servizi et al., 2019). This solution, however, comes at the cost of a larger GPS error (see Section 1). In this scenario, a broadly applied heuristic considers whether consecutive GPS positions, within a 1-min range, persist or not within a spatial range of 50 meters (Schuessler and Axhausen, 2009). The literature shows many developments in this direction, employing clustering techniques (Tietbohl et al., 2008; Zheng et al., 2009; Guidotti et al., 2015; Xiang et al., 2016), which can learn unsupervised and find stops within GPS trajectories. In multiple-step approaches, personal- (Guidotti et al., 2015), and geographical-context (Zheng et al., 2009) can augment trajectories' information and improve stops candidates' classification. Density-based spatial clustering of applications with noise (DBSCAN) is at the base of most frameworks; some of these frameworks can even find stop candidates directly on raster image representations (Wang and McArthur, 2018). Many other effective probabilistic unsupervised methods are available, as for example kernel-based (Thierry et al., 2013; Hariharan and Toyama, 2004), generative (Nurmi and Koolwaaij, 2006; Xiao et al., 2019), and discriminative (Liao et al., 2007), such as kernel-density algorithms, hidden markov models, and conditional random fields.

Some of these methods can be implemented to learn supervised by labels, or combined in multi-step approaches. Among the resulting hybrid solutions, we mention the integration of DBSCAN with a support vector machines (Gong et al., 2015), where the latter is a supervised-method. Within the realm of SBTS, Zhao et al. (Zhao et al., 2015) describes stop detection in Future Mobility Survey (FMS), which is a popular sensing platform for prompted recall surveys. The method uses a basic space/time-range set of rules for finding stop-candidates. Then, it retrieves frequent-place signatures from users' personal information, it merges continuous stops with still-mode, and it removes stops in excess after mode detection. Still-mode is an entity closely correlated with stops. To detect this class as one of a larger set of modes, this classifier is trained by labels. Using SBTS data and labels, thus a supervised ML method, a random forest can be very effective in still-mode detection and other transportation modes (Zhou et al., 2017).

Methods that use GPS trajectories can rely on derived features, such as: distance, speed, acceleration, bearing change, and time intervals. These features come in multiple flavors and are strictly related to the motion/still states of the device (Servizi et al., 2019; Wang et al., 2019; Yazdizadeh et al., 2019). Time also correlates to human activities, as work and leisure; thus, time of day, time of the week, time of the month, and time of year bring valuable information (Servizi et al., 2019; Wang et al., 2019; Yazdizadeh et al., 2019). Personal- and spatial-context data, which are very informative on where and why people travel, can improve methods performance. In the former category, we have features such as home and workplace addresses, socio-demographics, and trips history; in the latter category, e.g., land-use, nearby bus stops, routes, road, and rail network (Servizi et al., 2019; Wang et al., 2019; Yazdizadeh et al., 2019).

Depending on the available information within our application scenario, although supervised methods are proven very successful, in practice, collecting reliable labels from large users' populations, is hardly manageable with SBTS. On the contrary, unsupervised methods are often applied but do not provide the required level of accuracy. Therefore, they are combined with other methods. From this perspective, ANN have massive potential, as they can learn both supervised (Dabiri and Heaslip, 2018), semi-supervised (Dabiri et al., 2020), and unsupervised (Lim et al., 2020). Furthermore, some of the ANN configurations that are proven very effective in multidimensional data representations, such as convolutional neural networks (CNN), could benefit from a richer representation of the spatial context, which so far is limited to dummy variables (Xiao et al., 2016). For further detail on work related to ANN, see Section 2.2.

2.2. ANN trip classifiers and corresponding data representations

We describe further relevant literature on ANN classifiers, by positioning our work, within the space that each of the following subsections identify.

2.2.1. ANN configurations

Artificial Neural Networks are a very successful parametric nonlinear modeling approach, for pattern recognition and classification, that maps a tensor $\mathbf{X}$ of input variables to a tensor $\mathbf{Y}$ of target variables. The subset of models we use in this work, includes Feed-Forward Networks (FFN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN). These network models may specialize in different data structures.

FFN incorporate multiple layers of logistic regression with continuous and discontinuous nonlinearities. CNN are invariant to certain transformations of the input, like translations, scaling, small rotations, and elastic deformations. RNN have the property of taking into account inputs processed in preceding chunks of a sequence, while producing the output from the last input. CNN are widely applied to images; RNN, to time series.

An ANN can result from a combination of sub-models as FFN, CNN, and RNN. Each model can be determined choosing a number of hyperparameters (HP). The resulting model's optimal parameters are specified based on a training dataset, by minimizing the loss

function between the output of the model and the values of the target variable corresponding to the same input. In our case, FFN enable compounding the outputs of the preceding RNN and CNN models.

However, the likelihood function at the basis of ANN training is not a convex function of the model parameters. Consequently, training these models requires the allocation of substantial computational resources. Specifying the network parameters within a maximum likelihood framework involves the solution of a nonlinear optimization problem leveraging on backpropagation, which is a gradient-based algorithm. Therefore, to find a sufficiently good minimum, we need a grid-search and run such an algorithm multiple times, each time using a different combination of hyperparameters. The resulting performance, evaluated after training the model on an independent validation dataset, determines which hyper-parameters represent the best combination to solve the problem (Bishop, 2006, Ch. 5), in this case, the classification of stop and move points.

2.3. Classification score and setup

F1-score can measure the performance of a classifier. It is the harmonic mean of precision and recall $\mathrm{F1} = 2\frac{P \cdot R}{P+R}$, where precision $P = \frac{T_p}{T_p+F_p}$, recall $R = \frac{T_p}{T_p+F_n}$. T_p stands for *True Positives*, e.g. stops classified as stops when stop is the target class, and motion classified as motion when motion is the target class. F_p stands for *False Positives*, e.g. motion points classified as stop points when stop is the target class, vice versa when motion is the target class. F_n stands for *False Negatives*, e.g. stop points classified as motion points when stop is the target class, vice versa when motion is the target class. In Dabiri and Heaslip (2018), F1 scores are the weighted average calculated on 5 classes, and on 5-fold cross validation. In Jiang et al. (2017), the authors pick a random sample of the users to compose training, validation, and test set, then F1 score is computed on 4 classes, and on the test set only. We call this 1-fold cross validation method *leave-one-out*.

In this work, we split the dataset in three partitions, and we leave one out. We find optimal hyperparameters with the remaining two partitions, which we call training (TR) and validation (VA). To estimate the overall error distribution, we fix the optimal hyperparameters, and we proceed in two steps. First, we estimate performance on the unseen partition, which we call test (TE). Second, we perform a k-fold validation on the whole dataset.

2.4. Classification of GPS representations

We refer to two papers on mode detection. In Jiang et al. (2017) a Recurrent Neural Network (RNN) classifies points over four modes, which are: walk, bus, bike, and car. This network is composed by a two layers bidirectional gated recurrent unit (GRU) with maxout activation function, which process discretized speed features after an embedding layer. Dabiri and Heaslip (2018) present a CNN that classifies GPS segments over five modes: walk, bus, bike, car, and rail. Three convolutional layers using Rectified Linear Unit (ReLU) activation function compose the network, which process a four channel tensor representing: bearing rate, speed, acceleration and jerk. None of these two works implement data fusion with GIS. Referring to Dabiri and Heaslip (2018), a CNN approach seems to record the best performance over multiple baselines computed on the same settings, but with different machine learning methods, which are: (i) K-Nearest neighbor (KNN), with a range of neighbors between 3 and 40, finding 5 as optimal value; (ii) SVM with regularization in the range 0.5 and 20, finding 4 as optimal value; (iii) Decision tree (DT), with maximum depth of tree between 1 and 40, finding 10 as optimal value; (iv) Random forests (RF), with number of trees between 5 and 100, finding 85 as optimal value; (v) FFN, with number of hidden layers between 1 and 10, finding 1 as optimal value.

Dabri et al. extends the work on ANN classifier in Dabiri et al. (2020), so that the training can be semi-supervised.

2.5. Classification of GPS + GIS fusion representations

For purpose imputation, which is the classification of the activities performed at the stops (see Section 1), we found an example in Xiao et al. (2016). The paper presents an FFN classifying a feature vector that includes land-use type (LT) and points of interest, coded as dummy variables. For ANN classifiers specialized in mode detection and stop detection, we did not find examples of GPS + GIS fusion. In general, we found no examples of GPS + GIS fusion with representations beyond the dummy variables' space. As already mentioned in the Section 1, smartphones are equipped with multiple sensors. Although data recorded from these sensors can be fused on the time dimension (Wang et al., 2019) with GPS, we do not perform any fusion with other sensors at this time.

2.6. Classification approach

We found two main approaches, which are point-based and segment-based classification. As the name suggests, the entities that are processed and classified in the first case, are the features observed at each time step, while in the second case are sequences of observations (Prelipcean et al., 2016) on the time dimension. To allow the comparison of competing classifiers, (Prelipcean et al., 2016) proposed a penalization method to link the F1 score with the distances represented by the classification errors. By applying such a method, the authors show that point-based classifiers are superior than segment-based classifiers. Therefore, we choose to focus on a point-based classifier.

2.7. Comparability between different methods

There is consensus in the field about the lack of standardization for validating and comparing the performance of competing classifiers. Jiang et al. (2017) and Dabiri and Heaslip (2018) represent an evident example. Even though classifications are performed on the same dataset, they present different complexity as one deals with four classes, and the other with five; the validation setup is also different (see Section 2.3). Therefore, F1 scores comparison between these two tasks is meaningless. To mitigate the problem, in Section 2.6 we find a solution proposed by Prelipcean et al. (2016); in Wang et al. (2019), the authors propose both dataset and workflow for k-fold cross validation, which could provide a standardized baseline. Similarly to Wang et al. (2019), to estimate the distribution of the classification errors, we fix the models' optimal HP, and we apply the same experiment setup to ANN methods and baselines. Thus, we compare F1-average distributions.

3. Methodology

In this section we describe our two main contributions: the fusion between GPS and GIS with a multi-dimension tensor, and the ANN configurations specialized in the classification of such a tensor.

3.1. Data requirements

Our approach rely on two datasets: *User GPS trajectories* and *Geo-spatial context data from Open Street Map* (OSM). While the latter is a global defined dataset, the GPS trajectories will in most use cases origin from a geographically constrained area, e.g. country, or region where the travel survey is conducted. It is important that the two datasets cover the same range both with respect to space and time. For example, obviously, the subset of OSM used should cover the geographical area of the survey. To be meaningful, however, the OSM data should further describe a spatial context within the same *time frame* in which users generated their trajectories, as the geo-spatial context should not be altered.

We assume to have GPS trajectories for U different users, and for each user $u \in \{1, \ldots, U\}$ we denote the i^{th} GPS point, with $i \in \{1, \ldots, N_u\}$ having the following information:

- $t_{u,i}$: Time of the GPS point.
- $(lat_{u,i}, long_{u,i})$: Position components of the GPS point.
- $br_{u,i}$: Bearing between the GPS point i and $i-1$ in radians (Dabiri and Heaslip, 2018).
- $d_{u,i}$: Distance between GPS point i and $i-1$ in meters.

We denote the total number of GPS points $N = \sum_{u=1}^{U} N_u$. We further define the temporal context of each position using a *dummy variable* (1), as the discretization of $t_{u,i}$, taking into account that there is a significant difference between trip distributions during evening and night compared to the rest of the day (Li et al., 2016):

$$\mathbf{ToD}_{u,i} = \begin{cases} [1,0,0,0,0], & \text{if } t_{u,i} \in (00:00, 06:00] \\ [0,1,0,0,0], & \text{if } t_{u,i} \in (06:00, 10:00] \\ [0,0,1,0,0], & \text{if } t_{u,i} \in (10:00, 14:00] \\ [0,0,0,1,0], & \text{if } t_{u,i} \in (14:00, 18:00] \\ [0,0,0,0,1], & \text{if } t_{u,i} \in (18:00, 00:00] \end{cases} \tag{1}$$

The geo-spatial context, based on Open Streep Map (OpenStreetMap contributors, 2017), is a graph data model consisting of three basic data structures: nodes, ways, and relations. Each of these can represent physical features as shapes. For example, roads as segments, buildings and land use as polygons, intersections as points. Unique tags pinpoint each feature. We consider each specific value of a tag as distinct type of geo-spatial feature, whether they are attached to nodes, ways, or relations. We capture a selected subset of these feature types for use as the surrounding geo-spatial context of a given GPS trajectory point. The number of selected feature types is denoted C.

3.2. Data fusion process

To enrich GPS trajectories, we fuse each GPS point on the surrounding geo-spatial context, considering Open Street Map features within proximity of the GPS point. The result of the data fusion process is a multi-dimension tensor corresponding to each GPS position, $\mathbf{I}_{u,i} \in \mathbb{R}^{W \times H \times C}$. W and H represent the extension of the geo-spatial context in respectively horizontal and vertical direction from $(lat_{u,i}, long_{u,i})$ in some fixed unit (e.g. 10 m steps), and C is the number of geo-spatial context features captured. The spacial center of the $\mathbf{I}_{u,i}$ tensor is always the corresponding GPS data point. If a specific geo-spatial feature, c, is present in proximity j, k from $(lat_{u,i}, long_{u,i})$ then $\mathbf{I}_{u,i,j,k,c} = 1$, otherwise $\mathbf{I}_{u,i,j,k,c} = 0$.

However, naïvely querying for each GPS data point, $(lat_{u,i}, long_{u,i})$ to construct $\mathbf{I}_{u,i}$, is not computationally feasible. Hence, we pre-compute the geo-spatial context for a grid, $\mathbf{F} \in \mathbb{R}^{W' \times H' \times C}$ that is optimized for fast lookup and covers the entirety geographical area of interest. To build the grid, first, we calculate shapes for each cell in the grid using some fixed unit (e.g. 10×10 m). Second, we build a spatial-index using the *R-Tree* algorithm (Guttman, 1984), which allows us for any OSM shape to lookup all intersecting cells on

average $O(\log_2(W'H'))$ time (see Alg. 1). As a consequence, all GPS points, $(lat_{u,i}, long_{u,i})$, that are located in same cell will be assigned the same geo-spatial context. This is an acceptable trade-off since $\log_2(W'H') \ll N$, and thus the data fusion cost can be considered linear on the number of GPS points.

For each of the subset of selected Open Street Map feature types to be captured, c, we find all intersecting cells, j,k, in $\mathbf{F}$, using the *R-Tree index*, and assign $\mathbf{F}_{j,k,c} = 1$, and otherwise 0 (see Alg. 2).

At this point any $\mathbf{I}_{u,i}$ is simply a subset of $\mathbf{F}$, specifically given that the corresponding GPS point $(lat_{u,i}, long_{u,i})$ is located in cell j,k then $\mathbf{I}_{u,i}$ can be calculated in constant time using (2).

$$j,k|(lat_{u,i}, long_{u,i}) \vDash \mathbf{I}_{u,i} = \mathbf{F}_{j-\frac{W}{2}\ldots j+\frac{W}{2},k-\frac{H}{2}\ldots k+\frac{H}{2}} \tag{2}$$

A convenient visualization of $\mathbf{F}$ is presented in Fig. 1. Intuitively, we can think of tensor $\mathbf{F}$ as C images having size $W' \times H'$, stacked one on top of another. Each image, $c \in \{1, \ldots, C\}$ corresponds to the c^{th} spatial-context feature, which we visualize with a distinct color on pixels where $\mathbf{F}_{j,k,c} = 1$, and transparent otherwise.

3.3. ANN architecture design and optimization

For the point-based binary classification of GPS trajectories, this paper investigates two different models, using different ANN methods, and two competing data representations, which are:

1. RNN, using only the kinematic features derived from *GPS*, as defined in (3);
2. RNN, using the representation with dummy variables defined in (6);
3. A combination of RNN and CNN, using the data representation defined in (7).

Against the method we propose in Point 3, we include also a rule-based baseline model and a random forest (see Section 3.4). We tested the former with *GPS* as input, and the latter with both input representations described in (3), and (6).

3.3.1. Structural hyperparameters

As we are performing a classification task, the cross entropy is our loss function[1], which we can optimize testing various strategies, introducing further hyperparameters, i.e. optimizer, regularization rate, learning rate, and number of epochs (van Laarhoven, 2017).

To cope with severe class imbalance, while minimizing the loss function we can penalize the *stop* in favor of *motion*, according to the relative weight normalized on the size of the smaller class. We can also avoid training and back-propagating on batches representing the larger class only, by setting a large value for the hyperparameter *batch size*.

To keep exploding gradients under control when gradient convergence becomes very unlikely for large noise variance (Bengio et al., 1994), and avoid quick deterioration of the loss function, after back propagation we can apply a gradient clipping (Pascanu et al., 2013) strategy, introducing a specific hyperparameter named clipping rate.

To train a ANN model and specify its optimal parameters based on a training dataset, we pick one set containing one value for each of the hyperparameters listed in both this and the following sections, and we run one optimization loop.

To cope with the strong class imbalance mentioned in Section 1, which after training, on the validation set, translates into the classification of all observations as *stop*, we set a large batch-size and we penalize the larger class in the optimizer.

Based on the results of each training cycle, we can assess whether and how to tune each HP within a new set of HP. Every new set must be tested in a new training cycle. In Table 1 we report both hyperparameters and corresponding range of experimented values, while in the following sections we present in depth the hyperparameters describing the final configuration of our three ANN models.

Other critical hyperparameters shared across different ANN structures, are the following.

1. To compound the output of complex configurations, e.g. CNN and RNN, we can use FFN towards the end, where we need to find the optimal number of layers and neurons per layer.
2. To capture non-linear relationships we can rely on different Activation Functions (AF) through the whole network.
3. To improve the learning performance, we can implement batch normalization (Ioffe and Szegedy, 2015) between each layer, beginning from the input, throughout the network output.
4. Softmax is a special AF, implemented in the last layer to output probabilities summing to 1[2].
5. To contain over-fitting, we can implement dropout (Srivastava et al., 2014) layers within the network architecture, and L2 regularization (van Laarhoven, 2017) in the optimization algorithm.

3.3.2. RNN using only kinematic features

RNN repeatedly transform a sequence of inputs, where each input has the same dimensionality, R, and the length of the sequence, L is a hyperparameter. The output is a function of input and hidden state. The latter is updated based on the input vector and itself. The hidden state, then, is used to process the next input vector. The dimension of the hidden state is a hyperparameter. Multiple RNN layers

[1] Cross Entropy. Retrieved from web 26/11/2019.
[2] Softmax. Retrieved from web 26/11/2019.

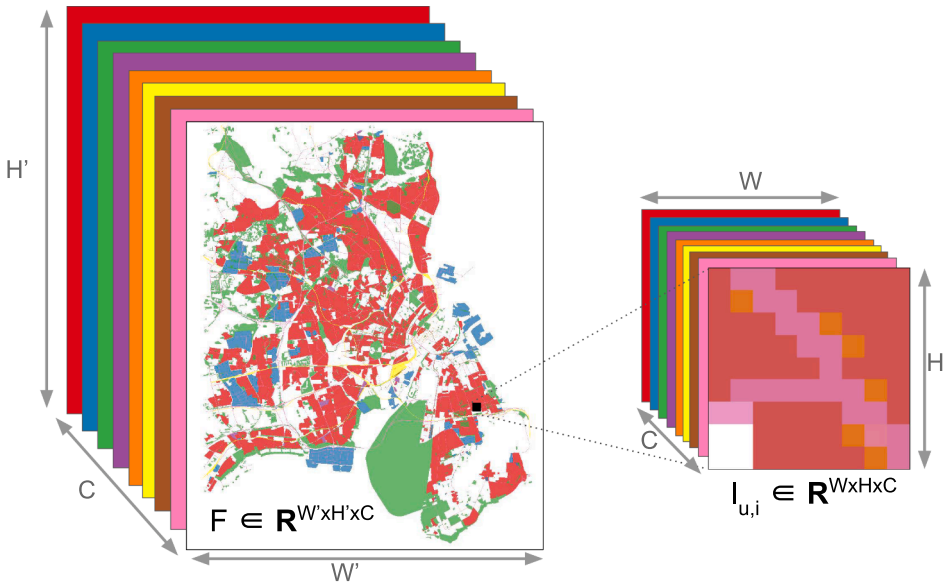

Fig. 1. Visual representation of 8 of the 11 channels of the grid: landuse residential, landuse industrial, landuse meadow, landuse commercial, shops, rail roads, roads, bus stops.

Table 1
List of hyperparameters and values' range tested in various ANN configurations.

Recurrent Neural Network Layers	$\in [1, 4]$
Hidden State Dimension	$\in [1, 30]$
Sequence length	$\in [3, 60]$
Convolutional Neural Network Layers	$\in [1, 4]$
Filters	$\in [16, 256]$
Filter Kernel	$\in [(1, 1), (5, 5)]$
Padding	$\in [0, 1]$
Stride	$\in [1, 2]$
Max Pooling Kernel Size	$\in [2, 3]$
Max Pooling Stride	$= 2$
Fully connected Layers	$\in [1, 4]$
Fully Connected Units	$\in [10, 2048]$
Dropout	$\in [0.2, 0.8]$
Activation Function	$\in$ {Rectified Linear Unit (ReLU), Leaky ReLU, Hyperblic Tangent (Tanh)}
Optimizer	$\in$ {Adam, Stochastic Gradient Descent (SGD), RMSProp}
L2 regularization Weight Decay	$\in [10^{-12}, 10^{-6}]$
Learning Rate	$\in [10^{-5}, 10^{-1}]$ implementing LR decay
Epochs	$\in [1, 100]$
Gradient Clipping Rate	$\in [0.3, 0.5]$
Batch Size	$\in [8, 12000]$
Items Shuffling during training	$\in$ Yes, No

can be stacked within the same network architecture, where the ouput of one layer becomes the input of the next: the number of layers is a hyperparameter.

1. Sequence length, L: Here it represents the sequence length in term of GPS points, thus time steps, that we feed into the network to preserve the temporal context of our time series.
2. Hidden State Dimension: Here we *store* previous sequences of GPS points, keeping a memory of the temporal context through the whole sequence-processing.
3. RNN Layers Number: By processing the time series in input, each layer outputs a new time series for the next layer, opening the possibility of decomposing and learning complex patterns. In case of bi-directional RNN, we have two blocks of layers. One will process the features derived from our GPS time series forward, as $F(GPS_0) \rightarrow F(GPS_t)$; the other, backward, as $F(GPS_t) \rightarrow F(GPS_0)$. Both will contribute to the same output.

Let $p_{u,i}$ be the representation of any GPS point through kinematic features, such that:

$$p_{u,i} \vDash d_{u,i} \parallel br_{u,i} \parallel \mathbf{ToD}_{u,i} \tag{3}$$

In this configuration (see architecture in Fig. 3), we represents $p_{u,i}$ as in (3).

The kinematic features consists of distance, d, and bearing, br, i.e. $R = 2$. We arrange the features into a sequence with length, L and process through a RNN followed with batch normalization, ReLU activation and dropout. The output of the RNN-block is concatenated with **ToD**, a dummy variable representing *time of day* (Servizi and Petersen, 2019). We processed the resulting feature vector through a FFN.

In particular, to preserve the time dependency of the time-series, the RNN was a GRU unit (Cho et al., 2014). As in Jiang et al. (2017), we took advantage of the motion laws behind the GPS trajectories, which should be confirmed despite the processing direction of the time series, and we configured GRU as bidirectional. With the bidirectional GRU, each element of the sequence receives the information, through the hidden state, from all the other elements. In this model, we classify the last element of the sequence only (Servizi and Petersen, 2019).

The final version of the model has about 60 000 parameters in total. The resulting optimal HP, selected with *TR* and *VA* partitions, are available in Tables 2 and 3. In Table 8, we report the resulting performance estimated using the optimal set of HP to classify the *TE* partition.

3.3.3. RNN using kinematic and geo-spatial features as dummy variables

From (2) we can obtain also a one-dimensional dummy variable representing the same spatial-context with lower resulution, which we define as:

$$\mathbf{D}_{u,i} \in \mathbb{R}^C \tag{4}$$

such that

$$\mathbf{D}_{u,i,c} = \begin{cases} 1, & \text{if } \Sigma_{j,k}\mathbf{I}_{u,i,j,k,c} > 0 \\ 0, & \text{if } \Sigma_{j,k}\mathbf{I}_{u,i,j,k,c} = 0 \end{cases} \tag{5}$$

The resulting alternative representation of each GPS position within the surrounding spatial context, using a dummy variable, is

$$p_{u,i} \vDash d_{u,i} \parallel br_{u,i} \parallel \mathbf{ToD}_{u,i} \parallel \mathbf{D}_{u,i} \tag{6}$$

This model is a twin of the model described in Section 3.3.2 and illustrated in Fig. 3. The difference is in the input $p_{u,i}$, which we model as described in (6), instead of (3). We include **D** with dimensionality C; thus, since we still include bearing and distance, $R = 2 + C$. This difference impacts on the total number of parameters of the model, in this configuration $\approx$ 82 000. Also the optimization HP are slightly different. The resulting HP are available in Tables 2 and 4; the resulting performance, in Table 8.

3.3.4. CNN + RNN using kinematic and geo-spatial features as tensors

In CNN, the convolution of an input tensor is obtained by striding a filter over such a tensor. Convolutional layers can be stacked. The output of the previous layer is a tensor that becomes the input of the next layer. The output's size of each transformation, depends on the following hyperparameters: number of layers, number and size of filter kernels, filter strides, and padding strategies of the input.

1. Number of Filters. Each filter is responsible of learning a pattern present in the tensor, considering all its channels at the same time, but limited to the size of the filter kernel.
2. Filter Kernel Size. It is the portion of the tensor on which the filter transformation is applied. This transformation convolves the whole tensor.
3. Number of Layers. Multiple convolutional layers are responsible of learning more complex patterns, combining the simpler patterns learned by the previous layers.
4. Stride Size. Small strides provide redundancy in the convolutions, whereas large strides might limit redundancy.
5. Padding Size. Padding strategies, in combination with filter kernel size, allow control on the volume of the convolution's output, enabling deeper or shallower architectures.

Table 2
RNN architecture hyperparameters, final configuration for both RNN with GPS only and RNN with GIS + GIS fusion with dummy variables.

Recurrent Neural Network Layers	2
Hidden State Dimension	4
Fully connected Layers	2
Fully Connected Units	Layer 1→ 512
	Layer 2→ 100
Dropout	0.45
Activation Function	Rectified Linear Unit (ReLU)

Table 3
RNN optimization hyperparameters, final configuration for GPS only.

Optimizer	Adam
L2 regularization Weight Decay	10^{-10}
Learning Rate Decay	Epoch 0–40 → 0.1
	Epoch 41–50 → 0.01
Epochs	50
Batch Size	12000
Gradient Clipping	0.5
Items Shuffling during training	Yes

Table 4
RNN optimization hyperparameters, final configuration for GPS + GIS with dummy variables.

Optimizer	Adam
L2 regularization Weight Decay	10^{-10}
Learning Rate Decay	Epoch 0–30 → 0.1
	Epoch 31–50 → 0.01
Epochs	50
Batch Size	13500
Gradient Clipping	0.5
Items Shuffling during training	Yes

Between convolutional layers, we can have Max Pooling Layers. Intuitively, this transformation is very similar to a convolution, and is defined by two hyperparameters: filter kernel size, and stride. In this layer, a filter kernel strides across the output of the previous convolutional layer, but here it specializes in extracting the most relevant signals towards the next convolutional layer.

The representation of $p_{u,i}$ that we obtain by augmenting kinematic features with spatial context through (3) and (2), is the following:

$$p_{u,i} \vDash d_{u,i} \parallel br_{u,i} \parallel \mathbf{ToD}_{u,i} \parallel \mathbf{I}_{u,i} \tag{7}$$

Compared to a dummy variable describing C geo-spatial features, within a defined range centered in a GPS position, $\mathbf{I}_{u,i}$ provides more detail around the GPS position (see Fig. 2), augmenting the resolution from $\mathbb{R}^C$ to $\mathbb{R}^{W\times H\times C}$.

CNN are a perfect fit extract features from multi-dimension tensors as $\mathbf{I}_{u,i}$, which we use in this configuration augment the kinematic features derived from GPS, with spatial context information. Hence, we combined the network described in Section 3.3.2 with a CNN, as described in Fig. 4. As with the RNN configuration of Sections 3.3.2 and 3.3.3, we classified only the last element of the sequence by assessing the hidden state. The configuration of this network required some further tuning. The resulting HP are available in Tables 5 and 6; the resulting performance, in Table 8. The challenge here was to reduce the total amount of parameters, which in this configuration are approximately 100000.

3.4. Baselines grid-search

The extensive reviews provided in Shen and Stopher (2014), Wang et al. (2019) and Yazdizadeh et al. (2019), report rule-based techniques as the most common methods for stop-detection and trip-segmentation. These methods perform classification based on, e.g., spatial-, time-, speed- or acceleration-thresholds. The same literature presents RF as one of the most effective supervised ML techniques. To assess the performance of our classifier, we pick two baselines: a rule-based method, which is unsupervised, and a RF, supervised.

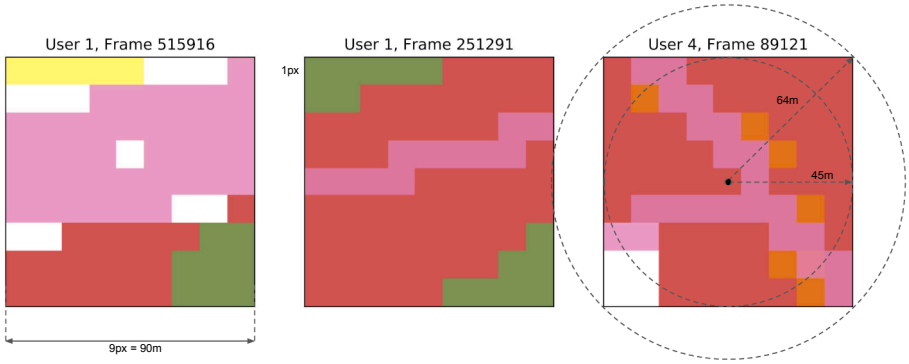

Fig. 2. Example of the 9x9 frames used as input for the CNN model. The channels are visualized using the same colors as used in Fig. 1.

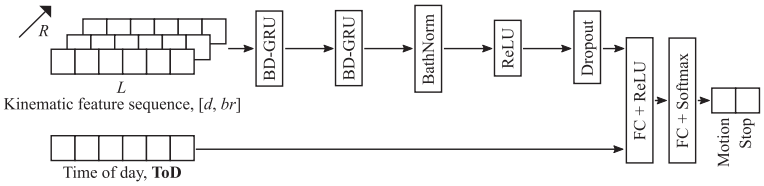

Fig. 3. Network architecture for RNN using only kinematic features.

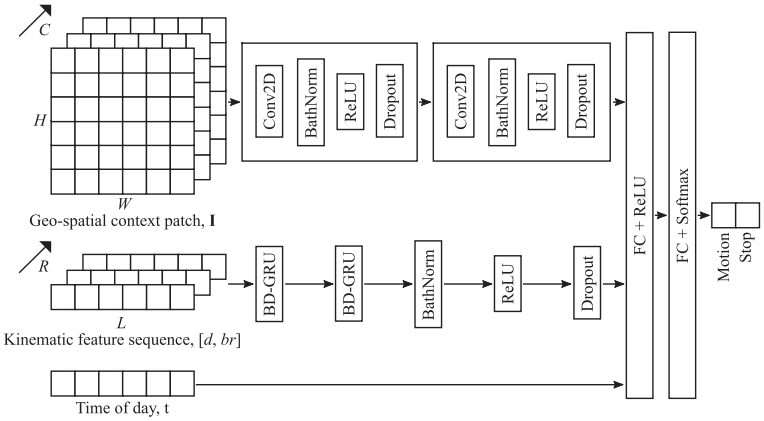

Fig. 4. Network architecture for CNN + RNN model.

1. Infostop (Aslak, 2019), an efficient python package recently published on GitHub, allows unsupervised stop detection and labeling of stationary events from a GPS trajectory. By building a network that links stationary events, identified as nodes within a critical space–time range, and clustering such a network using two-level Infomap[3], the algorithm provides a label for each point and stop event. This method does not support GPS + GIS data fusion in any way.
2. Sklearn, a very popular python package broadly used in Machine Learning, provides the RF algorithms used. To improve the classification accuracy and reduce over-fitting, RF relies on multiple decision tree predictors and averaging. To train this classifier, the first step is bootstrapping (Breiman, 2002), which consist in sampling a number of training sub-sets from the main training dataset. In the second step, each training sub-set is split into in-bag (Breiman, 2002) (IB) and out-of-bag (Breiman, 2002) (OOB), sized one-third and two-thirds of such a sub-set. Then, while the OOBs are left out, a decision tree is constructed on each IB, by sampling randomly the attributes to determine the decision split (Breiman, 2002). Results from each decision tree will be averaged, thereby providing the final classification. Because the OOB step, RF performance estimation during training is also unbiased

[3] Infomap is a network clustering algorithm based on the Map equation (Rosvall et al., 2009).

Table 5
CNN + RNN architecture hyperparameters, final configuration.

Recurrent Neural Network Layers	2
Hidden State Dimension	5
Convolutional Neural Network Layers	2
Filters	Layer 1→ 16
	Layer 2→ 8
Filter Kernel	3
Padding	1
Stride	1
Max Pooling Kernel Size	Layer 1→ 3
	Layer 2→ 2
Max Pooling Stride	1
Fully connected Layers	2
Fully Connected Units	Layer 1→ 512
	Layer 2→ 100
Dropout	0.45
Activation Function	Rectified Linear Unit (ReLU)

Table 6
CNN + RNN optimization hyperparameters, final configuration.

Optimizer	Adam
L2 regularization Weight Decay	10^{-10}
Learning Rate Inv.Decay	Epoch 0–10 → 0.01
	Epoch 11–50 → 0.1
Epochs	50
Gradient Clipping	0.3
Batch Size	13500
Items Shuffling during training	Yes

(Yazdizadeh et al., 2019). K-fold cross-validation, however, may still be appropriate to replicate the same conditions when comparing RF performance with other methods (see Section 2.7).

For the grid search, we refer to the random forest only, as Infostop is not a parametric method. We performed two grid-searches: For $p_{u,i}$ represented as in (3), and for $p_{u,i}$ represented as in (6), which means with and without dummy variables. For the task, we used *GridSearchCV* a dedicated functionality of Sklearn, where we specified $TR \cup VA$ as training partition. After 5-fold estimations on the set of hyperparameters described in Table 7, we obtained two set of optimal HP, one for each data representation. Results show no significant difference between the two configurations (see Table 8).

4. ANN classification performance and benchmark

The first remark is about the class imbalance between stop and motion. Most of the public datasets, as for example (Wang et al., 2019; Zheng and Fu, 2011), have relatively balanced classes and the stop class represent approximately 20% of the total. Unsurprisingly, the dataset we use presents realistic stop and motion proportions which are 80% and 20% of the total. Thus, the challenge for a model specialized on stop detection, on a realistic dataset, is detecting motion points. Suppose a model predicts the stop class only, as this large class overwhelms the model. Since stop is our target class, precision would be 80%, recall would be 100%, accuracy would be 80%, and F1-score would be 88.89%. By switching target class from stop to motion, the same prediction event would result in 0%

Table 7
Random forest grid search hyperparameters.

Number of estimators	$\in \{100, 200, 500\}$
Max features	$\in \{\text{auto}, \text{sqrt}, \log 2\}$
Max depth	$\in \{4, 6, 7, 8\}$
Criterion	$\in \{\text{gini}, \text{entropy}\}$

Table 8
Comparison between ANN models and baselines. Performance off the sample, after grid search, computed on TE partition (see Section 4.2).

Model	Mode	Precision	Recall	F1-average
RNN with GPS only	Motion	66%	86%	82%
	Stop	95%	85%	
RNN with GPS + GIS fusion (6)	Motion	68%	86%	83%
	Stop	95%	86%	
CNN + RNN with GPS + GIS fusion	Motion	**81%**	**88%**	**89%**
	Stop	**96%**	**93%**	
Infostop (Aslak, 2019) with GPS only	Motion	74%	73%	83%
	Stop	91%	91%	
Random Forest (RF) with GPS only	Motion	79%	80%	86%
	Stop	93%	93%	
RF with GPS + GIS fusion (6)	Motion	80%	79%	86%
	Stop	93%	93%	

precision, 0% recall, 0% accuracy and 0% F1-score (see Section 2.3). Due to this heavy class imbalance, F1-weighted-average on the class size would be a misleading metric. Therefore, within the TE partition, to assess and compare our models we need to measure F1-average. In case of k-fold validation, the average of the k resulting F1-averages should be weighted on the size of each TE_{k-fold}. For both models implementation and performance calculation, we used Python, Sklearn and PyTorch (Paszke et al., 2019).

4.1. Case study dataset

In 2018, The Center of Transport Analytics at the Technical University of Denmark, tested Mobile Market Monitor[4] (MMM), which is the commercial version developed from FMS, and collected GPS trajectories of $U = 12$ users, for about 24 user · days.

To manage the data fusion between GPS and GIS (see Section 2.5), we restricted the case study to the Copenhagen Capital area (see Fig. 1), consisting of approx. 1.5 million GPS trajectory points. Furthermore, we applied some preliminary cleansing, by excluding any point at the end of time intervals > 300 s and space intervals covered at speeds > 42 m/s ($\approx$150 km/h). The resulting dataset counts about 1.45 million GPS points (Servizi and Petersen, 2019).

We applied various filters; for example, removing any point at the end of time intervals > 60 s (instead of > 300 s), would further reduce the dataset by about 1.6% of the total. By removing also sequences of GPS observations counting less than 5 consecutive points, we obtain a clean dataset of over $N \approx 1.42$ million GPS points. These filters are extremely effective in removing faulty GPS observations: speed above the threshold is unrealistic; time intervals between observations above 60s, are the symptom of, e.g., battery saving routines, or GPS satellites out of sight; segments with less than 5 consecutive observations represent time-series too short to be classified, and can be a symptom of noise in the data.

For $\mathbf{F}$, we chose a 1×1 cell size of 10×10 m, and the resulting grid size is $H' = 2845, W' = 2331$, and thus grid consists of more than 6.6 million cells (see Fig. 1).

In our experiments, $\mathbf{I}_{u,i}$ represents a squared area of 90×90 m using a cell size of 10×10 m, and thus $H = W = 9$ (see Fig. 2). We capture up to $C = 11$ feature types from OSM data, for usage of the same cell space, such as bus stop, road, commercial land use, etc. Each of the 11 tensors' channels, is dedicated to one and only one spatial-context feature, which enables the representation of up to 11 mixed land-uses with 10×10 m resolution.

Among the geo-spatial contexts relevant for transport choice and behavioral study (Schuessler and Axhausen, 2009; Semanjski et al., 2017), we pick those that are represented in the available GIS. The selection consists of the following 11 features from OSM:

1 **Landuse residential**. A polygon that indicates that the area is primarily used for residential houses and homes. Knowing this information could help detect stops when a user is, e.g., at home or paying social visits.
2 **Landuse industrial**. A polygon which indicates that the area is primarily used for industrial buildings. The information could help in detecting stops when a user is, e.g., at work or on a business meeting.
3 **Landuse meadow**. A polygon which indicates that the area is primarily used for parks and forests. The information could help in detecting stops or walking move when a user is, e.g., doing sport or recreation activities.
4 **Landuse commercial**. A polygon around commercial areas, which could contribute in classifying, e.g., stops for meal/eating brakes or entertainment.
5 **Shops**. POIs indicating, e.g., shops and restaurants, which could contribute in detecting stops for, e.g., eating out, shopping, pick-up or drop-off someone.
6–9 **Rail, Metro, Stations and Bus stops**. Points that could contribute in detecting, e.g., motion with specific transport modes, instant-stops for mode transfer, or long-discontinuities at intermediate stations.
10 **Major road network**. Segments corresponding to main road networks, which could help detecting motion by car, bus or bike.

[4] Mobile Market Monitor. Retrieved from web 26/11/2019.

11 **Traffic lights.** Points identifying nodes where a traffic light is present, which could contribute in detecting long-discontinuities of motion.

Fig. 1 shows a visual representation combining 8 of the 11 channels of the grid, where we overlay each channel with its own color and some transparency. Our classifier does not process this visual representation (see Fig. 2), but the tensor $\mathbf{I}$ defined in (2).

4.2. Dataset partitions for grid-search and run time

To prevent information spillover during the grid-search, we cannot build training, validation and test data partitions by blindly shuffling our ≈ 1.4 million GPS points, and then sampling randomly the TR, VA, and TE in some ideal proportions. We need to keep intact the sequence of the u^{th} time series, such that $\forall u \in \{1,\dots,12\}$, the partitions $GPS_u = \{1,\dots,N_u\}$ represent continuous sequences on the time dimension, and are disjoint. Therefore, to perform the HP grid-search following the leave-one-out validation criterion described in Section 2.3, faster than a k-fold validation, first we sampled without replacement 9 random users out of 12, where we denote with s_n the user sampled on the n^{th} draw. Thus, we composed the partitions as in (8), (9),and (10) (Servizi and Petersen, 2019).

$$\begin{aligned} TR &= \cup_{u=s_1}^{s_8} GPS_u = \{1,\dots,N_{s_1}|\dots|1,\dots,N_{s_n}|\dots|1,\dots,N_{s_8}\}, \\ & card(TR) = 1,147,396 \end{aligned} \tag{8}$$

$$\begin{aligned} VA &= GPS_{s_9} = \{1,\dots,N_{s_9}\}, \\ & card(VA) = 115,564 \end{aligned} \tag{9}$$

$$\begin{aligned} TE &= \cup_{u=s_{10}}^{s_{12}} GPS_u = \{1,\dots,N_{s_{10}}|1,\dots,N_{s_{11}}|1,\dots,N_{s_{12}}\}, \\ & card(TE) = 165,342 \end{aligned} \tag{10}$$

We used $TR \cup VA$ to find optimal hyperparameters, and TE to provide an estimation of the models' performance.

The grid-search performed for both methods and baselines (see Section 3.3, 3.4), using these partitions, results in the following computation time.

For ANN, we measured that each training cycle lasts ≈ 15 seconds per Epoch. Therefore, with training cycles composed by number of epochs $\in [20,50]$, a grid-search requires a time interval $\in [0.3,7.5] \cdot 10^3$ seconds per HP. One dedicated Graphic Processing Unit (GPU) ran all the computations.

For RF, the total time required for the grid-search is in the interval $[1.5\cdot,2.1] \cdot 10^4$ seconds. In average, the time required for each hyperparameter is in the range $[1.1-1.6] \cdot 10^3$ seconds, with parallel computations across a 16 cores/ 32 threads CPU.

As Infostop is not a parametric method, training time is null.

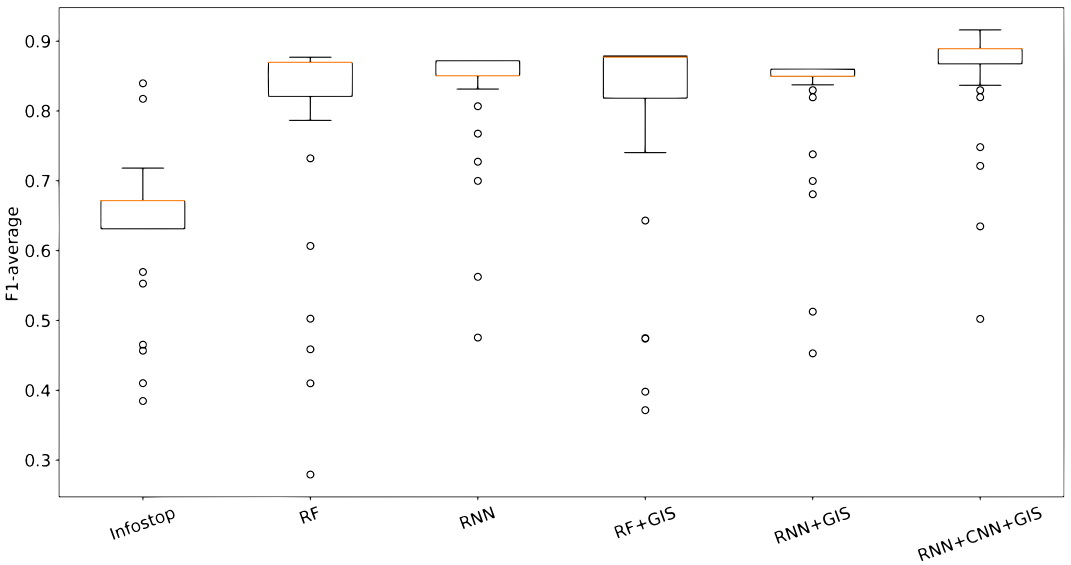

Fig. 5. Box-plot of F1-score performance, across models and baselines, obtained with 12-fold cross validation with $N \approx 1.42$ million GPS observations in total.

4.3. Benchmark

In the previous sections we described the process of finding optimal parameters for both ANN methods and baselines, using different representations of $p_{u,i}$. To find the best hyperparameters we used TR (8) and VA partitions (9), then we estimated the overall performance on the TE partition (10). The results are available in Table 8. To provide a distribution for performance and error, and allow a meaningful comparison across proposed methods and baselines, we fix the aforementioned optimal hyperparameters (see Section 3.3), and then we perform a 12-fold cross validation, split by users.

In this last step $VA = \varnothing$; TR and TE partitions have the same proprieties mentioned in Section 4.2, and are defined in (11).

$$TE_s = GPS_s \text{ and } TR_s = TE_s^{\complement}, \forall s \in [s_1, s_{12}] \tag{11}$$

First, we shuffled the list of users. Following the shuffled users' order $[s_1, s_{12}]$, fold by fold, we rotated each available user in the TE partition, while all the rest of the users composed the TR partition. In this way, we can evaluate the error distribution user by user.

To check for linear correlations between method performance and data noise we define the noise as:

$$\text{Noise}_u = \frac{card(GPS_u^{\text{raw}}) - card(GPS_u^{\text{clean}})}{card(GPS_u^{\text{clean}})} \tag{12}$$

This coefficient is simply the percentage of points removed on each users' trajectory because of data cleansing.

We define three key performance indexes (KPIs):

- The correlation among F1 $-\text{average}_{\text{u-fold}}$ and Noise_u (12);
- F1-score, defined as the average of F1 $-\text{average}_{\text{u-fold}}$, weighted on $card(TE_{\text{u-fold}})$, across the 12-folds;
- F1-std, defined as the average of F1 $-\text{std}_{\text{u-fold}}$, weighted on $card(TE_{\text{u-fold}})$, across the 12-folds.

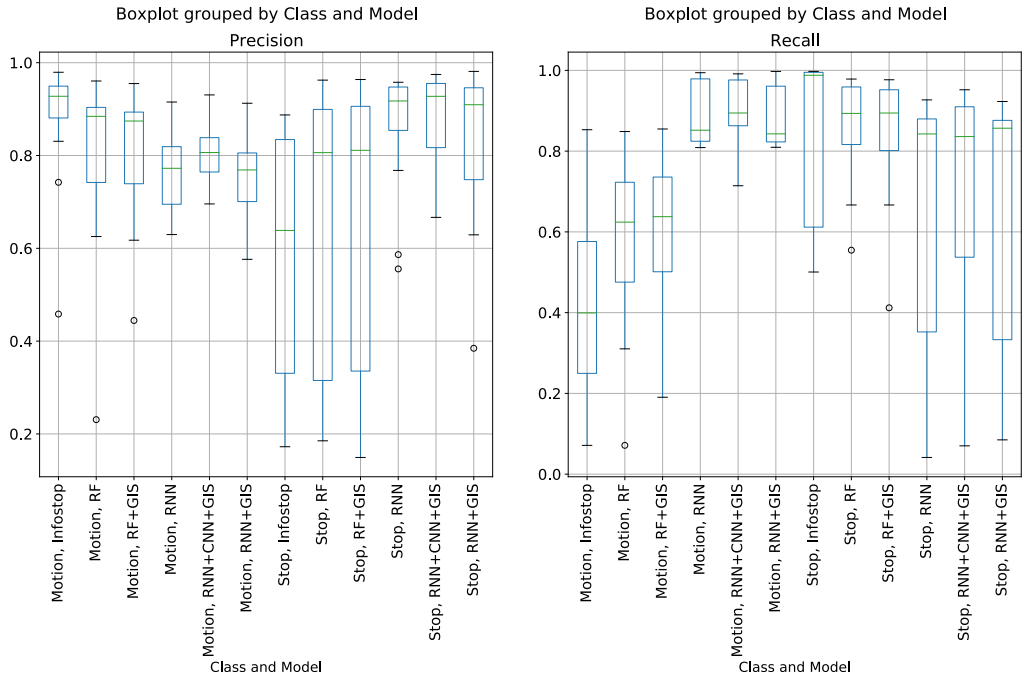

Fig. 6. Distribution of precision and recall performance, across models and baselines, obtained with 12-fold cross validation with $N \approx 1.42$ million GPS observations in total.

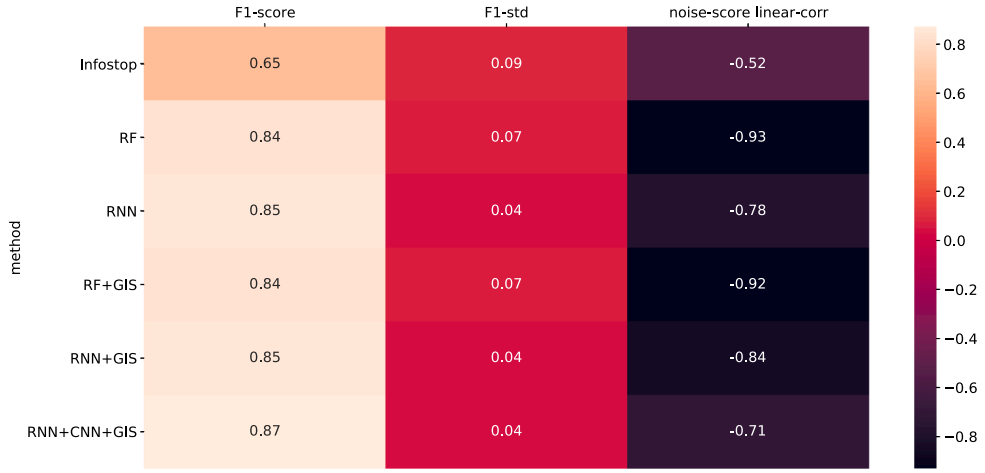

Fig. 7. Methods benchmark. The correlation is among F1 $-\text{average}_{\text{u}-\text{fold}}$ and Noise_u (12). F1-score and F1-std, are the average of F1 $-\text{average}_{\text{u}-\text{fold}}$ and F1 $-\text{std}_{\text{u}-\text{fold}}$, weighted on $card(TE_{\text{u}-\text{fold}})$, across the 12-folds.

In addition to these KPIs, we look at precision and recall. $card(\cup_{u=1}^{12} TE_u) \approx 1.42 \cdot 10^6$ observations contribute to the estimation. The results are presented in Fig. 5 and 6. CNN + RNN with GPS + GIS data fusion (see Section 3.3.4) is the best performer on both F1-related KPIs, and second best after Infostop on linear correlation with noise. The ranking is consistent to the results validated out of sample, after the grid search (see Table 8). From Fig. 6 is evident that this model is significantly better in terms of recall of the motion class, and is less depended than RF on the GPS noise.

4.4. Discussion

The previous sections show that the best classifier is the RNN + CNN model we propose. This model, specializes in the classification of the novel representation of $p_{u,i}$, as described in (7). Fig. 7 shows a strong negative linear correlation between the performance on each fold and the noise in the corresponding *TE* partitions defined in (12). Thus, when the tested partition presents high levels of Noise_u, F1-score is low, as expected. Our method is not significantly better when Noise_u is low. In contrast, when GPS observations are very noisy our method performs significantly better. The comparison with the other ANN models, which differ mainly on how the data is represented, shows that the representation of (7) is quite robust to GPS noise. This performance comes to the cost of heavier computations and more complex training, typical of ANN, which should be considered carefully before deployment on a larger scale (see Sections 3.3 and 4.2).

5. Conclusion

In this work, we used ANN to perform binary point-based classification - stop and motion - on $N \approx 1.42$ million points of GPS trajectories generated and validated by 12 users, during a test conducted with a smartphone-based travel survey (see Section 4.1).

We proposed the following models: RNN with GPS only data, RNN with GPS + GIS data fusion as dummy variables, and a combination of CNN and RNN with GPS + GIS data fusion represented as a multi-dimensional tensor (see Section 3.3). We compared the performance against a clustering method, and a random forest. The latter, with and without GPS + GIS fusion represented as dummy variables. The results show that CNN + RNN is the best performer in terms of F1-average, 3% over both random forest configurations, and 6.5% over Infostop (see Table 8). Results are leave-one-out validated according to (8)–(10).

To ease comparison among methods, performance and error distribution are estimated through a 12-fold cross validation with fixed optimal hyperparameters (11). The ranking is consistent with the aforementioned results. CNN + RNN classifier has the highest F1-score and the lowest F1-std (see Fig. 5). In particular, this method performs significantly better on recall for motion class (see Fig. 6), and when data are noisy (see Fig. 7).

ANN ran mostly on GPU; RF and Infostop, on CPU only (see Section 4). Although the process of GPS fusion with GIS, and training of ANN, is expensive in terms of CPU/GPU run time, in our experiments the difference seems handleable. Grid search and training time difference, seems mostly due to the larger number of hyperparameters of ANN. Yet, RF is faster of approximately one magnitude; Infostop does not require grid search at all. In contrast with RF, however, the method we propose for GPS + GIS fusion has a potential

not exploited yet. On the one hand, this method could assist ANN configurations specializing in unsupervised learning, for the stop detection task. On the other hand, it could support multi-task classification of both transport-mode and trip-purpose, certainly supervised, possibly unsupervised. Future research will verify whether this potential can translate into a tangible asset for SBTS.

Algorithm 1. Map WGS84-grid to Pixel-grid

Result: Reusable Pixel/WGS84 Map, saved on Disk
Input : Square coordinates on UTM32 grid
Output: Map of i^{th} pixel with i^{th} WGS84 cell

`/* initialize geographical area */`
$x_{min}, y_{min} \leftarrow$ lower-bound(SquareCoordinates)
$x_{max}, y_{max} \leftarrow$ upper-bound(SquareCoordinates)
`/* set resolution per pixel (m) */`
$dx \leftarrow dy \leftarrow 10$
`/* image size of spatial context */`
$W', H' \leftarrow \frac{x_{max}-x_{min}}{dx}, \frac{y_{max}-y_{min}}{dy}$
`/* initialize rtree index [36] */`
$idx \leftarrow$ init-rtree-index()
`/* initialize pixel count */`
$i \leftarrow 0$
foreach *x in range(0,W')* **do**
  foreach *y in range(0,H')* **do**
    $x_1 = x_{min} + x * dx$
    $y_1 = y_{min} + y * dy$
    $x_2 = x_{min} + (x+1) * dx$
    $y_2 = y_{min} + (y+1) * dy$
    `/* Set UTM32 bounds */`
    cellUTM32 = setSquare($A_{(x_1,y_1)}, B_{(x_1,y_2)}, C_{(x_2,y_2)}, D_{(x_2,y_1)}$)
    `/* Set WGS84 bounds */`
    cellWGS84 = transform(cellUTM32)
    `/* Map` i^{th} `pixel and` i^{th} `cell */`
    idx(i) = (cellWGS84, x, y)
    i=i+1
  end
end
CloseAndSave(idx)

Algorithm 2. Spatial context representation

Result: Image representation of geo-spatial context
Input : rtreeIndex (see Alg. 1), geo-spatial context features (on WGS84 grid)
Output: **F** of size $W' \times H' \times C$

```
/* load rtree index [36]                                   */
idx ← load-rtree-index()
/* initialize empty tensor                                 */
F (W' × H' × C) ← 0
/* initialize feature index                                */
C ← 0
foreach feature in SpatialContextFeatures do
    /* load shapes on WGS84 grid                           */
    shapes = queryOpenStreetMap(feature)
    /* pick pixels intersecting shapes                     */
    pixels = intersect(idx, shapes)
    foreach pixel in pixels do
        F(x,y,C)=1
        /* each pixel includes coordinates x ∈ [0, W'] and
           y ∈ [0, H'] (see Alg. 1)                        */
    end
    C=C+1
end
```

CRediT authorship contribution statement

Valentino Servizi: Conceptualization, Methodology, Data curation, Software, Investigation, Formal analysis, Writing - original draft, Writing - review & editing. **Niklas C. Petersen:** Methodology, Data curation, Software, Investigation, Validation, Writing - original draft, Writing - review & editing. **Francisco C. Pereira:** Conceptualization, Validation, Supervision. **Otto A. Nielsen:** Conceptualization, Validation, Supervision.

Declaration of Competing Interest

The authors declare that they have no competing interests.

References

Alvares, L.O., Bogorny, V., Kuijpers, B., De Macedo, J.A.F., Moelans, B., Vaisman, A., 2007. A model for enriching trajectories with semantic geographical information, in. In: GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems. https://doi.org/10.1145/1341012.1341041.

Aslak, U., 2019. Python package for detecting stop locations in mobility data. URL https://github.com/ulfaslak/infostop.

Bengio, Y., Simard, P.Y., Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Networks 5 (2), 157–166.

Bierlaire, M., Chen, J., Newman, J., 2013. A probabilistic map matching method for smartphone GPS data. Transport. Res. Part C: Emerg. Technol. 26, 78–98. https://doi.org/10.1016/j.trc.2012.08.001.

Bishop, C.M., 2006. Pattern Recogniton and Machine Learning.

Breiman, L., 2002. Manual on setting up, using, and understanding random forests v4.1. https://www.cta.man.dtu.dk/english/national-travel-survey.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation, in. In: EMNLP 2014–2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference. https://doi.org/10.3115/v1/d14-1179 arXiv:1406.1078.

Christiansen, H., 2006. Key figures from the Danish national travel survey. URL https://www.cta.man.dtu.dk/english/national-travel-survey.

Dabiri, S., Heaslip, K., 2018. Inferring transportation modes from GPS trajectories using a convolutional neural network. Transport. Res. Part C: Emerg. Technol. 86, 360–371. https://doi.org/10.1016/j.trc.2017.11.021.

Dabiri, S., Lu, C., Heaslip, K., Reddy, C.K., 2020. Semi-supervised deep learning approach for transportation mode identification using gps trajectory data. IEEE Trans. Knowl. Data Eng. 32, 1010–1023.

Gong, L., Sato, H., Yamamoto, T., Miwa, T., Morikawa, T., 2015. Identification of activity stop locations in GPS trajectories by density-based clustering method combined with support vector machines. J. Modern Transport. https://doi.org/10.1007/s40534-015-0079-x.
Guidotti, R., Trasarti, R., Nanni, M., 2015. TOSCA: TwO-Steps Clustering Algorithm for personal locations detection. In: GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems. https://doi.org/10.1145/2820783.2820818.
Guttman, A., 1984. R-trees: A dynamic index structure for spatial searching. SIGMOD Rec. 14, 47–57. https://doi.org/10.1145/971697.602266. URL http://doi.acm.org/10.1145/971697.602266.
Hariharan, R., Toyama, K., 2004. Project lachesis: Parsing and modeling location histories. In: Egenhofer, M.J., Freksa, C., Miller, H.J. (Eds.), Geographic Information Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 106–124.
Ioffe, S., Szegedy, C., 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Arxiv, 1–11. doi:10.1007/s13398-014-0173-7.2. arXiv:1502.03167.
Jiang, X., de Souza, E.N., Pesaranghader, A., Hu, B., Silver, D.L., Matwin, S., 2017. TrajectoryNet: An Embedded GPS trajectory representation for point-based classification using recurrent neural networks. In: Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering, pp. 192–200. arXiv:1705.02636. http://arxiv.org/abs/1705.02636.
Kim, Y., Pereira, F.C., Zegras, P.C., Ben-akiva, M., 2018. Activity Recognition for a Smartphone and Web- Based Human Mobility Sensing System. IEEE Intell. Syst. 33, 5–23. https://doi.org/10.1109/MIS.2018.043741317.
Koushik, A.N., Manoj, M., Nezamuddin, N., 2020. Machine learning applications in activity-travel behaviour research: a review. Transport Rev. 1–24. https://doi.org/10.1080/01441647.2019.1704307.
Liao, L., Fox, D., Kautz, H., 2007. Extracting places and activities from GPS traces using hierarchical conditional random fields. Int. J. Robot. Res. https://doi.org/10.1177/0278364907073775.
Li, D., Miwa, T., Morikawa, T., 2016. Modeling time-of-day car use behavior: A bayesian network approach. Transport. Res. Part D: Transport Environ. 47, 54–66. https://doi.org/10.1016/j.trd.2016.04.011. URL http://www.sciencedirect.com/science/article/pii/S1361920916302334.
Lim, K., Jiang, X., Yi, C., 2020. Deep clustering with variational autoencoder. IEEE Signal Process. Lett. 27, 231–235.
Nurmi, P., Koolwaaij, J., 2006. Identifying meaningful locations. In: 2006 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, MobiQuitous. doi:10.1109/MOBIQ.2006.340429.
OpenStreetMap contributors, 2017. Planet dump retrieved from https://planet.osm.org, https://www.openstreetmap.org.
Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks. In: 30th International Conference on Machine Learning, ICML arXiv:1211.5063.
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates Inc, pp. 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
Prelipcean, A.C., Gidofalvi, G., Susilo, Y.O., 2016. Measures of transport mode segmentation of trajectories. Int. J. Geograph. Informat. Sci. 30, 1763–1784. https://doi.org/10.1080/13658816.2015.1137297.
Rosvall, M., Axelsson, D., Bergstrom, C.T., 2009. The map equation. Eur. Phys. J. Special Top. 178, 13–23. https://doi.org/10.1140/epjst/e2010-01179-1 arXiv:0906.1405.
Schuessler, N., Axhausen, K.W., 2009. Processing raw data from global positioning systems without additional information. Transp. Res. Rec. 2105, 28–36. https://doi.org/10.3141/2105-04.
Semanjski, I., Gautama, S., Ahas, R., Witlox, F., 2017. Spatial context mining approach for transport mode recognition from mobile sensed big data. Comput. Environ. Urban Syst. 66, 38–52. https://doi.org/10.1016/j.compenvurbsys.2017.07.004.
Servizi, V., Petersen, N.C., 2019. Source code for Stop Detection for Smartphone-based travel surveys using ANN methods. https://github.com/niklascp/dtu-deep-learning-project.
Servizi, V., Pereira, F.C., Anderson, M.K., Nielsen, O.A., 2019. Mining user behaviour from smartphone data: a literature review. arXiv:1912.11259.
Shen, L., Stopher, P.R., 2014. Review of GPS Travel Survey and GPS Data-Processing Methods, doi:10.1080/01441647.2014.903530.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Machine Learn. Res. 15, 1929–1958. https://doi.org/10.1214/12-AOS1000 arXiv:1102.4807.
Stopher, P., Jiang, Q., FitzGerald, C., 2005. Processing GPS data from travel surveys. In: 28th Australasian Transport Research Forum, ATRF 05.
Thierry, B., Chaix, B., Kestens, Y., 2013. Detecting activity locations from raw GPS data: A novel kernel-based algorithm. Int. J. Health Geograph. https://doi.org/10.1186/1476-072X-12-14.
Tietbohl, A., Bogorny, V., Kuijpers, B., Alvares, L.O., 2008. A clustering-based approach for discovering interesting places in trajectories, in. In: Proceedings of the ACM Symposium on Applied Computing. https://doi.org/10.1145/1363686.1363886.
van Laarhoven, T., 2017. L2 regularization versus batch and weight normalization, CoRR abs/1706.05350 (2017). arXiv:1706.05350. URL http://arxiv.org/abs/1706.05350.
Wang, Y., McArthur, D., 2018. Enhancing data privacy with semantic trajectories: A raster-based framework for gps stop/move management. Trans. GIS 22, 975–990. https://doi.org/10.1111/tgis.12334 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/tgis.12334. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12334.
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. https://doi.org/10.1109/ACCESS.2019.2890793.
Xiang, L., Gao, M., Wu, T., 2016. Extracting stops from noisy trajectories: A sequence oriented clustering approach. ISPRS Int. J. Geo-Informat. https://doi.org/10.3390/ijgi5030029.
Xiao, G., Juan, Z., Zhang, C., 2016. Detecting trip purposes from smartphone-based travel surveys with artificial neural networks and particle swarm optimization. Transport. Res. Part C: Emerg. Technol. 71, 447–463. https://doi.org/10.1016/j.trc.2016.08.008.
Xiao, G., Cheng, Q., Zhang, C., 2019. Detecting travel modes from smartphone-based travel surveys with continuous hidden Markov models. Int. J. Distrib. Sens. Netw. https://doi.org/10.1177/1550147719844156.
Yazdizadeh, A., Patterson, Z., Farooq, B., 2019. An automated approach from gps traces to complete trip information. Int. J. Transport. Sci. Technol. 8, 82–100. https://doi.org/10.1016/j.ijtst.2018.08.003. URL http://www.sciencedirect.com/science/article/pii/S2046043018300236.
Zahabi, S.A.H., Ajzachi, A., Patterson, Z., 2017. Transit trip itinerary inference with gtfs and smartphone data. Transp. Res. Rec. 2652, 59–69. https://doi.org/10.3141/2652-07.
Zhao, F., Ghorpade, A., Pereira, F.C., Zegras, C., Ben-Akiva, M., 2015. Stop detection in smartphone-based travel surveys. Transport. Res. Procedia 11, 218–226. https://doi.org/10.1016/j.trpro.2015.12.019.
Zheng, Y., Fu, H., 2011. Geolife GPS trajectory dataset - User Guide, Technical Report November 31, 2011. http://research.microsoft.com/apps/pubs/?id=152176%5Cnhttp://research.microsoft.com/apps/pubs/default.aspx?id=152176.
Zheng, Y., Zhang, L., Xie, X., Ma, W.Y., 2009. Mining interesting locations and travel sequences from GPS trajectories, in: WWW'09 -. In: Proceedings of the 18th International World Wide Web Conference. https://doi.org/10.1145/1526709.1526816.
Zhou, C., Jia, H., Juan, Z., Fu, X., Xiao, G., 2017. A data-driven method for trip ends identification using large-scale smartphone-based gps tracking data. IEEE Trans. Intell. Transp. Syst. 18, 2096–2110.