Semi-Supervised Deep Learning Approach for Transportation Mode Identification Using GPS Trajectory Data

Sina Dabiri, Chang-Tien Lu, Kevin Heaslip, and Chandan K. Reddy, *Senior Member, IEEE*

Abstract—Identification of travelers' transportation modes is a fundamental step for various problems that arise in the domain of transportation such as travel demand analysis, transport planning, and traffic management. In this paper, we aim to identify travelers' transportation modes purely based on their GPS trajectories. First, a segmentation process is developed to partition a user's trip into GPS segments with only one transportation mode. A majority of studies have proposed mode inference models based on hand-crafted features, which might be vulnerable to traffic and environmental conditions. Furthermore, the classification task in almost all models have been performed in a supervised fashion while a large amount of unlabeled GPS trajectories has remained unused. Accordingly, we propose a deep **SE**mi-Supervised **C**onvolutional **A**utoencoder (**SECA**) architecture that can not only automatically extract relevant features from GPS segments but also exploit useful information in unlabeled data. The SECA integrates a convolutional-deconvolutional autoencoder and a convolutional neural network into a unified framework to concurrently perform supervised and unsupervised learning. The two components are simultaneously trained using both labeled and unlabeled GPS segments, which have already been converted into an efficient representation for the convolutional operation. An optimum schedule for varying the balancing parameters between reconstruction and classification errors are also implemented. The performance of the proposed SECA model, trip segmentation, the method for converting a raw trajectory into a new representation, the hyperparameter schedule, and the model configuration are evaluated by comparing to several baselines and alternatives for various amounts of labeled and unlabeled data. Our experimental results demonstrate the superiority of the proposed model over the state-of-the-art semi-supervised and supervised methods with respect to metrics such as accuracy and F-measure.

Index Terms—Deep learning, semi-supervised learning, convolutional neural network, convolutional autoencoder, GPS trajectory data, trip segmentation, transportation mode identification

1 Introduction

The mode of transportation for traveling between two points of a transportation network is an important aspect of users' mobility behavior. Identifying users' transportation modes is a key step towards many transportation related problems including (but not limited to) transport planning, transit demand analysis, auto ownership, and transportation emissions analysis. Traditionally, the information for modeling the mode choice behavior was obtained through travel surveys. High cost, low-response rate, time-consuming manual data collection, and misreporting are the main demerits of the survey-based approaches [1]. With the rapid growth of ubiquitous GPS-enabled devices (e.g., smartphones), a constant stream of users' trajectory data can be recorded. A user's GPS trajectory is constructed by connecting GPS points of their GPS-enabled device. A GPS point contains the information of the device geographic location at a particular moment. Mining trajectory data, which contain rich spatio-temporal information regarding human activities, provokes several transport-domain applications such as incident detection, mobility pattern extraction, and transport mode inference [2]. In this study, we aim to predict a user's transportation mode purely based on their GPS trajectories.

A majority of models for learning transportation modes from GPS tracks consists of two steps: (1) extracting features from GPS logs, (2) feeding features to a supervised learning method for the classification task. Unlike many other data sources, the GPS-based trajectory does not contain explicit features for inferring transportation modes, which calls for feature engineering. Much of the current literature has generated hand-crafted features using the descriptive statistics of motion characteristics such as maximum velocity and acceleration [3], [4], [5]. After creating a pool of manual attributes, a wide range of traditional supervised mining algorithms has been used for performing the classification task including rule-based methods, fuzzy logic, decision tree, Bayesian belief network, multi-layer perceptron, and support vector machine [1].

However, the feature engineering not only requires expert knowledge but also involves biased engineering

- *S. Dabiri is with the Department of Civil Engineering and the Department of Computer Science, Virginia Tech, Blacksburg, VA 24061. E-mail: sina@vt.edu.*
- *C-T. Lu and C. K. Reddy are with the Department of Computer Science, Virginia Tech, Blacksburg, VA 24061. E-mail: ctlu@vt.edu, reddy@cs.vt.edu.*
- *K. Heaslip is with the Department of Civil Engineering, Virginia Tech, Blacksburg, VA 24061. E-mail: kheaslip@vt.edu.*

Manuscript received 17 June 2018; revised 15 Jan. 2019; accepted 28 Jan. 2019. Date of publication 1 Feb. 2019; date of current version 1 Apr. 2020. (Corresponding author: Sina Dabiri.)
Recommended for acceptance by M. A. Cheema.
Digital Object Identifier no. 10.1109/TKDE.2019.2896985

1041-4347 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

justification and vulnerability to traffic and environmental conditions. For example, one may use the maximum speed of a GPS trajectory as a discriminating feature. The immediate criticism is that the maximum velocity of a car might be equal to bicycle and walk modes under a congested traffic condition. The other expert may choose the top three velocities and accelerations of the user's GPS trajectory as a potential solution for lack of information about traffic conditions. Nonetheless, another specialist might critique this solution by asking why not using the top four velocities or why not the minimum instead of maximum? Automated feature learning methods such as deep learning architectures is a remedy to the above-mentioned shortcomings. Recently, researchers have shown an increased interest in leveraging deep learning algorithms for addressing challenging transportation-related problems [6], [7], [8].

Fig. 1 depicts the overview of our framework for identifying the transportation mode(s) of a GPS trajectory related to a user's trip. Since travelers might commute with more than one transportation mode for making a single trip, the first step is to partition the GPS trajectory of a trip into segments, in which every GPS segment contains only one transportation mode. Next, features of each GPS segment is automatically extracted using a deep learning framework based on Convolutional Neural Network (CNN). Accordingly, one of our main challenges is to convert the raw GPS segment into an adaptable layout for CNN schemes. First the basic motion characteristics of every GPS point in a segment including relative distance, speed, acceleration, and jerk are computed [9]. This results in generating a sequence of numerical features for every type of motion characteristic. Next, the computed motion sequences are concatenated to create a 4-channel tensor for every GPS segment, where every sequence is equivalent to a channel in an RGB image. Such a new representation not only yields a standard arrangement for the CNN scheme but also describes the kinematic motion of a transport mode. Furthermore, in contrast to the hand-designed approaches, our proposed representation involves all GPS points of a user's trajectory rather than a small subset in the hand-designed approaches such as GPS points with maximum velocity or acceleration. Finally, the transportation mode of a GPS segment is inferred by training a CNN-based deep learning architecture on the converted GPS segments.

Moreover, almost all the current research work on travel mode identification has built their models using only labeled trajectories. Nonetheless, a significant portion of trajectories in GPS data might not be annotated since the acquisition of labeled data is a more expensive and labor-intensive task in comparison with collecting unlabeled data. Using the unlabeled data in addition to the labeled ones allows us to capture more properties of the data distribution, which can potentially further improve the decision boundaries and provide a better generalization on unseen records [10]. Thus, our main objective in this paper is to improve the CNN classifier by leveraging the power of deep unsupervised learning algorithms such as Convolutional AutoEncoder (Conv-AE). A deep **SE**mi-Supervised **C**onvolutional **A**utoencoder (**SECA**) architecture, that integrates Conv-AE and CNN classifier, is proposed. Both components are simultaneously trained by minimizing a cost

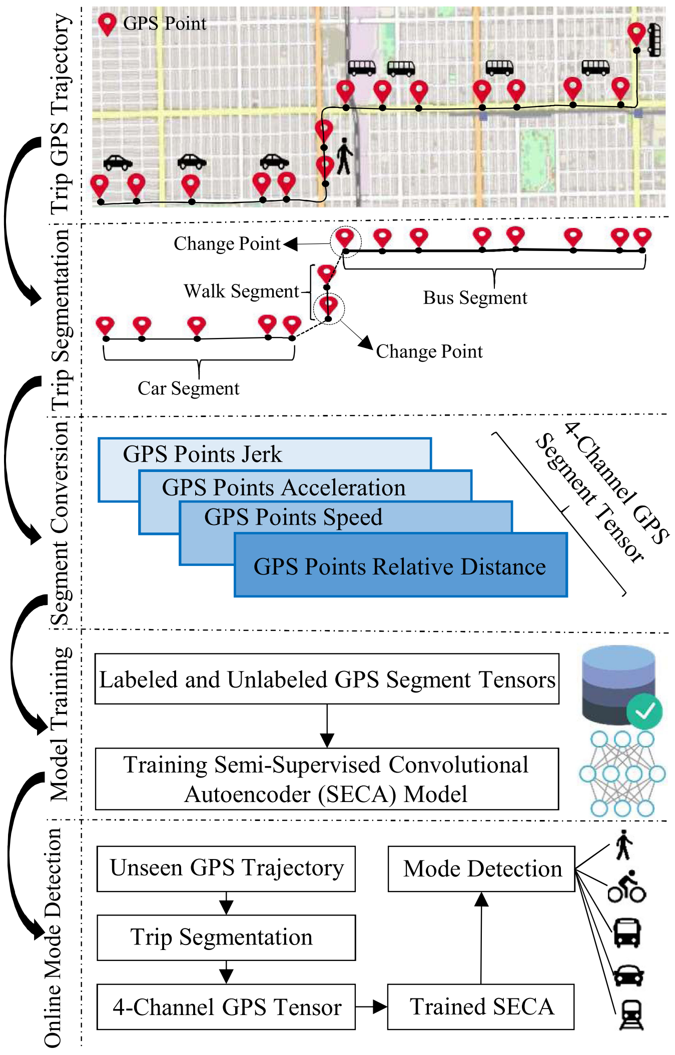

Fig. 1. Overview of our framework for detecting transportation mode of GPS trajectories. A raw GPS trajectory of a user's trip is first partitioned into a set of GPS segments with only one transportation mode. Next, each GPS segment is converted to a 4-channel tensor. The created labeled and unlabeled tensors are then used for training our proposed SECA model. The trained SECA model is finally used for detecting the transportation mode(s) of an unseen GPS trip.

function that is a linear combination of unsupervised and supervised losses. Tuning the hyperparameters that connect these losses is the most challenging part of our training procedure. The key contributions of this work are summarized as follows:

- *Developing a two-step trip segmentation process.* Due to the inherent need of CNN-based models for having a fixed-size input, the GPS trajectory of a trip is first uniformly partitioned into the GPS segments with a fixed size. Afterward, for the first time in this domain, a discrete optimization algorithm is deployed to detect the points where the transportation mode changes. The output of this step is a pool of GPS segments with only one transportation mode.
- *Designing an efficient representation for raw GPS trajectories.* A new procedure is developed for converting a raw GPS segment, which is a sequence of GPS points, to an efficient and appropriate representation for using in deep learning architectures. The proposed

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

representation contains the information of *all* GPS points in the GPS segment and allows the very deep architecture and training algorithm to extract discriminating and high-level features for the task-at-hand.

- *Developing a novel deep semi-supervised convolutional autoencoder (SECA) architecture.* A deep semi-supervised model is proposed to leverage both *unlabeled* and labeled GPS trajectories for predicting transportation modes. The model contains Conv-AE and CNN classifier for unsupervised and supervised learning, respectively.
- *Building an effective schedule for tuning balancing parameters.* Since the main objective is to simultaneously training the unsupervised and supervised components of the SECA model, a novel and efficient schedule is proposed for varying the balancing parameters that combine reconstruction and classification losses.
- *Conducting an extensive set of experiments for performance evaluation and comparison.* The results reveal that our SECA model outperforms several supervised and semi-supervised state-of-the-art baseline methods for various amounts of labeled GPS segments. Furthermore, the performance results demonstrate the superiority of the proposed trip segmentation process, the designed representation for GPS segments, the schedule for varying hyperparameters, and the model structure by comparing with several alternatives.

The rest of this paper is organized as follows. Existing works on both mode detection methods and deep semi-supervised schemes are listed in Section 2. After providing some preliminaries in Section 3, details of our mode detection framework are explained in Section 4. Our experimental results are reported in Section 5. Finally, the paper is concluded in Section 6.

2 Related Work

To date, several studies have deployed various data sources (e.g., GPS, mobile phone accelerometers, and Geographic Information System) or a combination of them for inferring users' transportation modes [7], [11]. In this section, we review the studies that have utilized the GPS data for designing a mode detection model since this is the primary focus of this paper. A comprehensive and systematic review of existing techniques for travel mode recognition based on GPS data is available in [1]. The paper provides an excellent comparison of various approaches in three categories including GPS data preprocessing, trip/segmentation identification, and travel mode detection. After reviewing GPS-based detection models, we will also briefly discuss various semi-supervised deep learning architectures that have been studied in the literature for different applications.

2.1 GPS-Based Mode Detection Models

As mentioned earlier, feature extraction and classification are two major tasks in the GPS-based mode detection frameworks. Since the classification is often performed by traditional supervised algorithms (e.g., support vector machines, decision trees, etc.), the feature-extraction design is the primarily discerning factor among various mode detection frameworks.

In two seminal studies by Zheng et al. [3], [4], a supervised framework based on hand-crafted features was proposed. Using the commonsense knowledge of the real world, a trip is first partitioned into segments by detecting the walk segments. Then, a set of manual yet robust features were identified for every segments and fed into machine learning algorithms (e.g., decision trees) for the classification task. Features were divided into basic and robust groups. The basic group primarily contains descriptive statistics of velocity and acceleration of all GPS points while the robust group is composed of the heading change rate, stop rate, and the velocity change rate. They demonstrated that the robust features are less vulnerable to traffic conditions; however, using a combination of basic and robust features results in higher accuracy. Xiao et al. [5] generated new features by computing more descriptive statistics such as mode and percentile. The number of features were further augmented by introducing local features through profile decomposition algorithms. Menp et al. [12] found that spectral features of speed and acceleration are significantly effective based on statistical tests while auto- and cross-correlations, kurtoses, and skewnesses of speed and acceleration were not useful. Nevertheless, research on semi-supervised mode inference is really scarce, Rezaie et al. [13] performed a semi-supervised label propagation method, yet based on a limited number of hand-crafted features including speed, duration and length of a trip, as well as the proximity of a trip start and end points to the transit network.

A small body of literature has sought to integrate hand-crafted and automated features that are extracted using deep neural networks [9], [14], [15]. After converting a raw GPS trajectory into a matrix with the image format, a type of deep learning algorithm is employed to obtain high-level representations for the classification task. Nonetheless, to the best of our knowledge, none of the deep learning approaches for mode detection has been built upon simultaneously using both labeled and unlabeled trajectories. Furthermore, no optimization technique has been exploited for trip segmentation.

2.2 Semi-Supervised Deep Learning Approaches

Semi-supervised frameworks based on deep learning algorithms (e.g., recurrent and convolutional neural networks, and autoencoders) have been exploited for a variety of tasks, mainly in computer vision and natural language processing fields [16], [17]. Existing literature on semi-supervised deep-learning architectures falls into two major groups: (1) Two-step process, in which the network is first trained in an unsupervised fashion as the pre-training step, and then the supervised component of the model is tuned using the labeled data. (2) Joint process, in which both the unsupervised and supervised components (i.e., the entire network) are concurrently trained.

One technique for the pre-training phase is to sequentially train the network layers [18]. Each layer is pre-trained with an unsupervised learning algorithm such as autoencoders and Restricted Boltzman Machine as a separated block while its input is the output of the previous layer.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

Indeed, the layer-wise unsupervised training strategy helps optimization by finding a good initial set of weights near a good local minimum, which sets the stage for a final training phase [19]. In the next stage, the deep architecture is fine-tuned by performing a local search from a reasonable starting point. Another pre-training strategy is to train the network in its entirety, rather than greedy layer-wise training, and then fine-tune the model using weights obtained from the first phase as an initialization. Using the obtained weights in the first step as a starting point for the supervised learning gives rise to a better stabilization and generalization of the model.

As mentioned earlier, one of the main objectives in the pre-training step is to prepare a good initialization for the supervised training and avoid a poor generalization of the model. However, with advances in initialization and regularization schemes for deep learning architectures [20], [21], pre-training strategies are getting replaced with joint training strategies. The fundamental idea of joint strategies is to optimize a hybrid loss function, that is a combination of unsupervised and supervised components, with the goal of simultaneously preserving reconstruction and discrimination abilities. While a type of classifier (e.g., multi-layer perceptron or softmax function) forms the supervised part, variants of autoencoders have been widely utilized for performing the unsupervised task. Variational autoencoders [10], convolutional-deconvolutional autoencoders [22], autoencoders based on a Ladder network [16], and recursive autoencoders [23] are typical examples of autoencoders that have been used in deep semi-supervised networks. A substitute for autoencoders is to add an entropy regularization upon unlabeled data into the supervised loss function. At each training step, the unlabeled data are annotated using the updated weights in the previous iteration [24]. A balancing parameter can be used in the hybrid loss function to trade off the supervised and unsupervised parts of the objective function [22], [23]. In Section 5, the effect of balancing parameter(s) in the ultimate performance of joint training strategies is examined.

In addition to be designed for a new application with a unique model configuration, our SECA model is trained using a new schedule for tuning balancing parameters. Unlike similar approaches in literature, the proposed schedule considers a separate balancing parameter for each of unsupervised and supervised components.

3 Preliminaries

This section introduces the preliminaries required to comprehend the proposed framework in Section 4. First, the trip segmentation and mode detection problems are described and then, we show how to compute the motion characteristics of each GPS point. The motion features of GPS points are then used for both creating an input layer in our SECA model and hand-crafted features in the standard machine learning algorithms.

3.1 Definitions and Problem Statements

Before describing the formal statements of the two problems, the notions of GPS trajectory and GPS segment are defined.

Definition 1 (GPS Trajectory). *A user's raw GPS trajectory T is defined as a sequence of time-stamped GPS points $p \in T$, $T = [p_1, \ldots, p_N]$. Each GPS point p is a tuple of latitude, longitude, and time, $p = [lat, lon, t]$, which identifies the geographic location of point p at time t.*

T is divided into *trips* if the time interval between two consecutive GPS points exceeds a pre-defined threshold (e.g., 20 minutes) [3]. Also, the user might commute with more than one transport mode in a single trip. For instance, one may travel to work by first driving to a parking lot, then taking a bus, and finally walking toward their workplace. As a result, a trip is partitioned into multiple GPS segments when the transportation mode changes.

Definition 2 (Change Point). *A change point, denoted as CP, is defined as the place in a trip in which users change their transportation mode. A trip may contain zero, one, or multiple change points.*

Definition 3 (GPS Segment). *A GPS segment is a subdivision of a user's trip, which is traveled by only one transportation mode $y \in Y$, where Y is a set of transportation modes (such as bike and car). A GPS segment is denoted as $SE = [p_1, \ldots, p_M]$, where M is the number of GPS points that forms SE.*

Accordingly, the trip segmentation problem is defined as follows:

Problem 1 (Trip Segmentation). *The trip segmentation problem is defined as detecting the change points CP in the GPS trajectory of a user's single trip. GPS segments SEs with a unique transportation mode are the output this problem.*

The mode detection is a multi-class classification problem that seeks for predicting the correct transportation mode for a given SE. However, the raw SE needs to be transferred to an appropriate format before feeding into a machine learning algorithm. Either a set of hand-crafted features (for traditional machine learning algorithms) or a proper layout (for deep learning architectures), represented as $\mathbf{X}$, will need to be designed. The first step for generating $\mathbf{X}$ is to compute the motion characteristics of GPS points in each SE, which is described in the next section.

Problem 2 (Mode Detection). *Given the training data $\{(\mathbf{X}_i, y_i)\}_{i=1}^{n}$ for n samples of SE_i, the mode detection problem is defined as building the optimal classifier for estimating the transportation mode of a user's SE based on its features $\mathbf{X}$.*

The trained model is then deployed to estimate users' transport modes while traveling in transportation networks. For an unseen trip, a trip is first segmented into a set of SEs using the proposed trip segmentation technique. Then, the mode for each SE is estimated. Consecutive segments with the same detected modes are concatenated together.

3.2 Motion Characteristics of GPS Points

Several motion features can be computed for every GPS point based on their geographic coordinates and timestamps. The relative distance between two consecutive GPS points in a SE (e.g., p_1 and p_2), can be computed using the widely-used Vincenty's formula [25]. The Vincenty's

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

formula is a common and accurate method for computing the geographical distance between two points on the surface of a spheroid. The time interval between two successive GPS points is another motion quantity that can be simply computed. Having the relative distance and time interval, other fundamental kinematic motions including speed, acceleration, and jerk are computed to provide more information about a user's motion. Speed is the rate of change in distance that shows how fast a user is traveling. Acceleration is the rate of change in speed that shows how fast a user is changing their speed. Jerk, the rate of change in acceleration, is a significant factor in safety issues such as critical driver maneuvers and passengers' balance in public transportation vehicles [26]. Jerk has been used in mode detection models for the first time in [9]. These motion features for a GPS point p_1 are calculated based on the following equations:

$$RD_{p_1} = Vincenty\big(p_1[lat, lon], p_2[lat, lon]\big) \tag{1}$$

$$\Delta t_{p_1} = p_2[t] - p_1[t] \tag{2}$$

$$S_{p_1} = \frac{RD_{p_1}}{\Delta t_{p_1}} \tag{3}$$

$$A_{p_1} = \frac{S_{p_2} - S_{p_1}}{\Delta t_{p_1}} \tag{4}$$

$$J_{p_1} = \frac{A_{p_2} - A_{p_1}}{\Delta t_{p_1}}, \tag{5}$$

where RD_{p_1}, Δt_{p_1}, S_{p_1}, A_{p_1}, and J_{p_1} represent the relative distance, time interval, speed, acceleration/deceleration, and jerk of the point p_1, respectively. Analogously, the above-mentioned formulae are used to calculate the motion features of other GPS points in a SE.

The rate of change in the heading direction of different transportation modes varies. For example, cars and buses have to move only alongside existing streets while people with walk or bike modes alter their directions more frequently [4]. Bearing rate is a motion attribute for quantifying the heading change among modes. As depicted in Fig. 2, bearing measures the angle between the line connecting two successive points and a reference (e.g., the magnetic or true north). The bearing rate, which can be used as another motion feature, is the absolute difference between the bearings of two consecutive points.

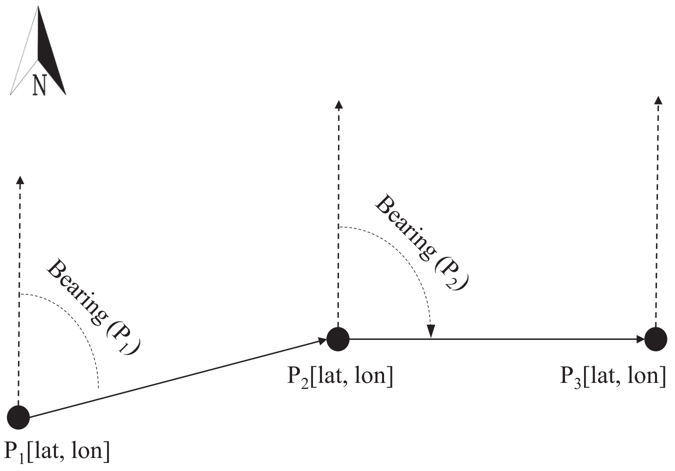

Fig. 2. Bearing between two consecutive GPS points. lat and lon represents the latitude and longitude of a GPS point.

4 The Proposed Framework

In this section, first, a two-step process is proposed to divide the GPS trajectory of a user's trip into the segments with only one transportation mode. Next, an efficient representation for each GPS segment is designed. The overall architecture of our semi-supervised framework is also explained in detail. Finally, an effective strategy for training our network is proposed.

4.1 Two-Step Trip Segmentation

Our proposed method for partitioning a user's trip into segments with a unique transportation mode consists of two parts. Considering the basic CNN requirement for having all input samples with a fixed size, all GPS trajectories need to be either truncated or padded to a fixed size for both offline learning and online inference at the end. Accordingly, we flip this requirement into the first step of our trip segmentation. Thus, a GPS trip is first uniformly partitioned into segments with a fixed number of GPS points, denoted as M. Our observation indicates that a majority of GPS segments contains one or two transportation modes after this uniform-size segmentation, which dramatically improves the performance of the overall segmentation process. In the online mode detection, consecutive segments with the same predicted transportation mode are merged together.

However, the uniform-size segmentation in the first step does not guarantee that every fixed-size GPS segment contains only one transportation, which calls for the second segmentation step. Assuming the fixed-size GPS segment $SE = [p_1, \ldots, p_M]$ as a signal, we aim to detect a number of change points, K, with their positions, $\mathbf{CP} = [CP_1, \ldots, CP_K]$, where the statistical properties of the signal change. Note that every change point belongs to $SE = [p_1, \ldots, p_M]$. To this end, SE is first converted into a multivariate time series $\{\mathbf{Y}_t\} \in \mathbb{R}^{(M \times d)}$, where d is the number of motion features introduced in Section 3.2. Our observation indicates using only speed and acceleration features for creating $\{\mathbf{Y}_t\}$ results in a better performance.

A common approach for detecting the change points from a time-series signal is to solve an optimization problem with a cost function that measures the goodness-of-fit of the sub-signals [27]. Unlike the pattern-matching techniques [28], an optimization-based model does not require a set of predefined pattern templates. Furthermore, an optimization-based model can be formalized without any assumption on how the signal has been generated, unlike the model-based approaches that rely on several assumptions including independent observations and a specific probability distribution (e.g., multivariate normal) on the signal. Note that the model-based approaches (e.g., Bayesian and Markov Random Field approaches) form another important track in the literature of change point detection.

Thus, the second step of our trip segmentation process is formalized as solving the following objective function:

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

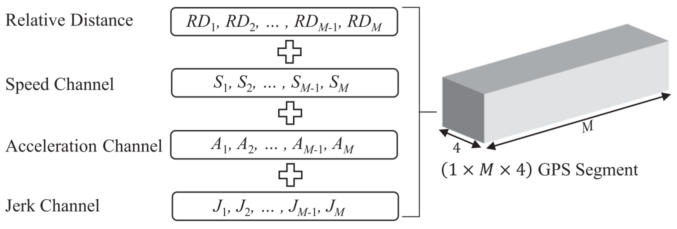
Relative Distance
$RD_1, RD_2, \dots, RD_{M-1}, RD_M$
Speed Channel
$S_1, S_2, \dots, S_{M-1}, S_M$
Acceleration Channel
$A_1, A_2, \dots, A_{M-1}, A_M$
Jerk Channel
$J_1, J_2, \dots, J_{M-1}, J_M$
4
M
(1 × M × 4) GPS Segment

Fig. 3. A 4-channel representation for a GPS segment with a shape of $(1 \times M \times 4)$.

$$C(\mathrm{CP}) = \sum_{i=1}^{K+1} [c(\mathbf{Y}_{CP_{i-1}+1:CP_i})] + \gamma f(K), \tag{6}$$

where $CP_0 = 0$ and $CP_{K+1} = M$. c is a cost function that measures the homogeneity in the sub-segments. We choose the mean-shift model, as one of the most studied cost function used in the change point detection literature [27]. For a GPS sub-segment $\{\mathbf{Y}_t\}_I$ on a time interval I between two consecutive change points, the mean-shifts is defined as follows:

$$c(\mathbf{Y}_I) = \sum_{t \in I} ||\mathbf{Y}_t - \bar{\mathbf{Y}}||_2^2, \tag{7}$$

where $\bar{\mathbf{Y}}$ is the mean of $\{\mathbf{Y}_t\}_{t \in I}$.

Since the number of change points K is unknown in our problem (i.e., a trip might be performed with various number of transportation modes), a penalty function $f(K)$ is used in Eq. (6) to constrain the number of change points. We choose a linear penalty function as γK. The penalty level γ makes a balance between the decrease in the cost function when more change points are allowed. Accordingly, solving the optimization problem in Eq. (6) is dependent on the choice of the penalty level γ, not a pre-defined number of change points.

A wide range of search algorithms have been proposed to solve the discrete change point detection optimization problem, formulated in Eq. (6) [27]. Our choice of the search algorithm is the Pruned Exact Linear Time (PELT) method, which has been recently proposed in [29]. PELT leverages the advantages of alternative search algorithms (e.g., binary segmentation and dynamic programming), which are exact solution and low computational complexity achieved through a combination of optimal partitioning and pruning. Under the assumption that change points are spread throughout the signal rather than confined to one portion, the PELT algorithm discards many points along the signal through its pruning step. This results in dramatically reducing the computational cost to on average $\mathcal{O}(n)$ while keeping the ability to detect the optimal segmentation. The detailed algorithm can be found in [29].

If PELT finds a change point in a GPS SE, the SE is partitioned into two or more sub-segments. After converting each SE into a 4-channel tensor, as described in the following section, all short sub-segments are padded with zero values to have a fixed-size M before feeding into our SECA model.

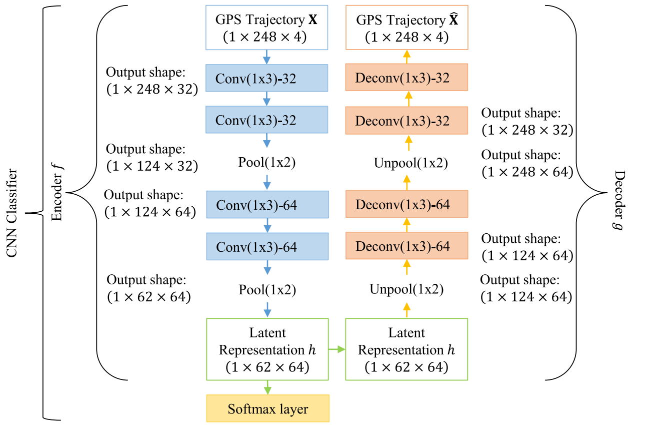
CNN Classifier
Encoder f
Decoder g
GPS Trajectory X (1 × 248 × 4)
GPS Trajectory X̂ (1 × 248 × 4)
Output shape: (1 × 248 × 32)
Conv(1x3)-32
Conv(1x3)-32
Deconv(1x3)-32
Deconv(1x3)-32
Output shape: (1 × 248 × 32)
Output shape: (1 × 124 × 32)
Pool(1x2)
Unpool(1x2)
Output shape: (1 × 248 × 64)
Output shape: (1 × 124 × 64)
Conv(1x3)-64
Conv(1x3)-64
Deconv(1x3)-64
Deconv(1x3)-64
Output shape: (1 × 124 × 64)
Output shape: (1 × 62 × 64)
Pool(1x2)
Unpool(1x2)
Output shape: (1 × 124 × 64)
Latent Representation h (1 × 62 × 64)
Latent Representation h (1 × 62 × 64)
Softmax layer

Fig. 4. The architecture of our semi-supervised framework, which consists of the convolutional-deconvolutional autoencoder and CNN classifier. The layers' parameters are represented by "(filter size)-(number of filters)" for Conv. and Deconv. layers, and "(pooling size)" for pooling and unpooling layers. The "Output shape" denotes the output size of the corresponding layer, which is shown only when the output size changes.

4.2 New Representation for Raw GPS Segments

Since the core component of our proposed framework is a convolutional network, we will need to first convert GPS segments into a format that is not only compatible with CNN architectures but is also efficient in representing the fundamental motion characteristics of a moving object. As explained earlier, the motion features utilized in this study are relative distance (RD), speed (S), acceleration (A), and jerk (J). For every type of motion feature, a sequence can be created by placing the corresponding value for every GPS point of a SE in chronological order, where the feature value is computed using Eqs. (1), (2), (3) , (4), and (5). Such a sequence can be seen as a 1-d channel. Stacking these channels turns a raw GPS SE into a 4-channel tensor. In Section 5, we demonstrate the superiority of such a configuration compared to other possible feature combinations.

By padding the short segments with zero values, all GPS SEs are mapped into a 4-channel tensor with the shape of $(1 \times M \times 4)$. We select M as the median size of all GPS segments obtained based on the *true* change points, which results in the best final performance. Fig. 3 illustrates the 4-channel arrangement for a GPS SE. Note that our proposed layout utilizes the information of all GPS points and allows the very algorithm to extract the efficient features for the mode detection. This is in contrast to the feature-engineering methods that leverage the information of a limited subset of the GPS segment such as points with the maximum speed and acceleration. Finally, values of each channel are individually scaled into the range [0,1] using the min-max normalization.

4.3 Semi-Supervised Convolutional Autoencoder (SECA) Model

As can be seen in Fig. 4, our semi-supervised architecture combines two main components: (1) a CNN classifier, which takes in only the labeled trajectories, denoted as $\mathbf{X}_l \in \mathbb{R}^{(1 \times M \times 4)}$, and (2) Convolutional-deconvolutional AutoEncoder (Conv-AE), which takes in both labeled and unlabeled trajectories, denoted as $\mathbf{X}_{comb} = (\mathbf{X}_l + \mathbf{X}_u) \in \mathbb{R}^{(1 \times M \times 4)}$. $\mathbf{X}_l$ and $\mathbf{X}_u$ correspond to the 4-channel tensors of labeled and unlabeled GPS segments, respectively, which are created based on the procedure described in the previous section.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

4.3.1 Convolutional-Deconvolutional AutoEncoder

Autoendocer is an unsupervised learning technique that aims to learn an efficient latent representation by reconstructing the input at the output layer. Autoencoder consists of two parts: (1) the encoder function that maps the input data into the latent representation $h = f(\mathbf{X})$, and (2) the decoder function that reconstructs the original data from the latent representation $\hat{\mathbf{X}} = g(h)$. The latent representation h often contains more useful properties than the original input data $\mathbf{X}$ [30]. In our framework, the functions f and g are deep convolutional and deconvolutional networks, respectively.

As shown in Fig. 4, the encoder function f consists of two sets of layers, where each set has two convolutional layers followed by a max pooling layer. The input to the encoder is $\mathbf{X}_{comb}$. Since the spatial size of $\mathbf{X}_{comb}$ is small in our application, a small filter size (1×3) is used for all convolutional layers while stride is equal to 1. The number of filters starts from 32 in the first set of convolutional layers, and then increase by a factor of 2 (i.e., 64) for the second set. The padding setting of convolutional layers is configured so as to preserve the spatial dimension after each convolution operation. The filter size of the max-pooling layer is (1×2) with the stride 2. According to the mentioned settings, the spatial size reduces by a half size of the previous layer only after max-pooling layers and remains unchanged after convolutional layers. The convolved neurons are activated by the Rectified Linear Unit ($ReLU$) function. The tensor h is the output of the last layer in the encoder part. Note that we do not use a fully-connected layer as the last layer to force the latent representation h into a vector form. Based on our experimental observation, collapsing the latent representation h with 3 dimensions into a 1-dimension vector deteriorates the performance of the model. The tensor h in Fig. 4, for instance, has the shape size $h \in \mathbb{R}^{(1 \times 62 \times 64)}$.

The decoder function g has the same number of layers as the encoder and performs the inverse operations (i.e., unpooling and deconvolutional) so as to generate an output with the same size of the input in the corresponding layer in the encoder part. For example, in Fig. 4, the tensor $h \in \mathbb{R}^{(1 \times 62 \times 64)}$ is first passed into the unpooling layer, which generates a tensor with the size $(1 \times 124 \times 64)$. Afterwards, the output feature map from the unpooling layer is fed into a deconvolutional layer, which results in a tensor with the same shape $(1 \times 124 \times 64)$. Except for the last layer, the activation function for all deconvolutional layers is $ReLU$. Proceeding with the same operations, the last deconvolutional layer produces an output with the same shape of the original input, denoted as $\hat{\mathbf{X}}_{comb} \in \mathbb{R}^{(1 \times M \times 4)}$. Since the input layer $\mathbf{X}_{comb}$ has been normalized into the range $[0, 1]$, the $sigmoid$ function is deployed as the activation function of the last deconvolutional layer.

As $\mathbf{X}_{comb}$ and $\hat{\mathbf{X}}_{comb}$ are composed of continuous-valued features, we use the squared euclidean distance as the loss function for the Conv-AE. Accordingly, the reconstruction error for every SE is computed as follows:

$$l^{Conv-AE} = \sum_i (\hat{x}_i - x_i)^2, \tag{8}$$

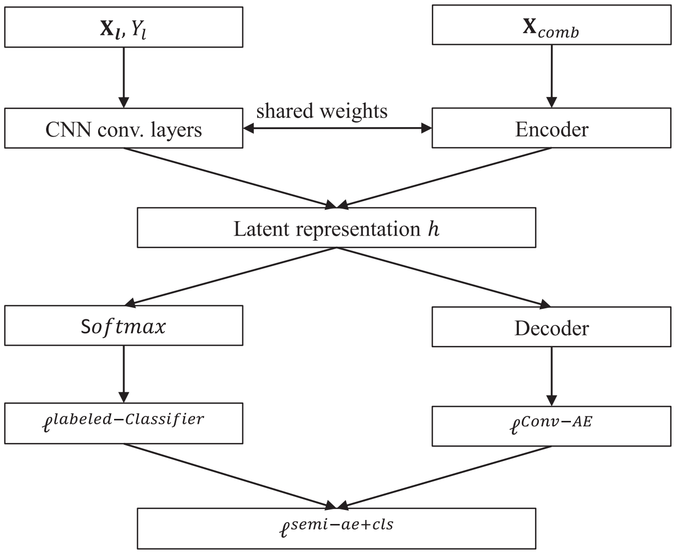

Fig. 5. Flow for jointly training the supervised and unsupervised components of the proposed SECA model, depicted in Fig. 4.

where $\hat{x}_i$ and x_i are the corresponding elements of the matrices $\hat{\mathbf{X}}_{comb}$ and $\mathbf{X}_{comb}$, respectively. The above error is averaged across the training batch in each iteration.

4.3.2 CNN-based Classifier

Our CNN classifier contains a stack of convolutional layers with one fully-connected layer. The convolutional part is exactly the same as the encoder function, yet receives $\mathbf{X}_l$ as the input layer. Therefore, the flattened latent representation h is directly fed into a $softmax$ layer to generate a probability distribution over the transportation labels for the GPS segment $\mathbf{X}_l$, denoted as $P_l = \{p_{l,1}, \ldots, p_{l,K}\}$, where K is the number of transportation modes. Note that using some fully connected layers between the last convolutional layer and the $softmax$ layer (i.e., deploying a multi-layer perceptron) does not improve the performance of our network. The widely accepted categorical cross-entropy is used as the loss function for the CNN classifier. The loss function for every labeled SE is formulated as follows:

$$l^{labeled-classifier} = -\sum_{i=1}^{K} y_{l,i} \log (p_{l,i}), \tag{9}$$

where $y_{l,i} \in \mathbf{Y}_l$ is a binary indicator which is equal to 1 if the class i is the true transportation label for the sample $\mathbf{X}_l$ and 0, otherwise. $\mathbf{Y}_l$ is the true label for $\mathbf{X}_l$, represented as one-hot encoding. Analogous to the Conv-AE, the cross entropy loss is averaged across the training batch in each iteration.

4.3.3 Model Training

Our main training strategy is to simultaneously train the Conv-AE and CNN classifier. The rationale behind this joint training strategy is to extract useful information from the underlying distribution of the input data through the Conv-AE, meanwhile enhancing the discrimination ability of the architecture using the classifier. As shown in Fig. 5, the encoder part of Conv-AE and the convolutional part of the CNN classifier, which have the same structure, need to share the same weights. As a consequence, in every weights

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

update, the latent representation matrix h obtained by the encoder function is equivalent to the output of the last pooling layer in the classifier. The unsupervised and supervised components of our proposed network are jointly learned by minimizing the following total loss function, which is a linear combination of Eqs. (8) and (9):

$$l^{semi-ae+cls} = \alpha l^{Conv-AE} + \beta l^{labeled-classifier}, \tag{10}$$

where α and β are the model hyperparameters that make a balance between the relative importance of the two losses in Eqs. (8) and (9). The appropriate schedule for varying values of α and β over training epochs is an integral part of our model learning. Details for tuning these two hyperparameters are elaborated in the next section. Note that we use Adam optimizer as the optimization technique [31], Glorot uniform initializer for initializing the layers weights [20], and a dropout regularization with the dropout ratio 0.5 before the $softmax$ layer to overcome the overfitting problem [21].

4.4 Parameter Tuning and Scheduling

Our proposed training procedure for scheduling α and β over training epochs consists of two steps:

- *Setting* $\alpha = 1$ *and* $\beta = 1$**:** At the first step, both the Conv-AE and CNN classifier are simultaneously trained while they have the same weights. Training continues through several epochs until the validation score drops down. Training in this step is stopped after two epochs with no further improvement. Indeed, the main goal in the first step is to obtain the best possible performance without making any trade-off between reconstruction and cross-entropy errors. The weights with the best validation score are restored for the next step.
- *Setting* $\alpha \in [1, 1.5]$ *and* $\beta = 0.1$**:** No further improvement is achieved by the previous setting, which mainly stems from the overfitting problem and/or getting stuck in local minima. Dramatically reducing the effect of the supervised component can act like a sharp perturbation and take the optimization out of local minima. The unsupervised weight can be kept fixed or marginally increased. In our application, increasing α up to 1.5 yields nearly the same performance. Continuing training with the new setting gives another chance to the optimization so as to move towards better local minima. The same as the first step, the training is stopped when the validation score is not improved after two consecutive epochs. The optimal weights are restored for classifying the test set.

In addition to implementing two steps for training, our proposed schedule for varying balancing parameters differs from similar joint training strategies in the following three main aspects.

- Neither fixed values nor annealing strategies are used over training epochs. In the annealing strategy, the value of α (or β) gradually decreases during the training to transit the focus towards the supervised (or unsupervised) task in the last training iterations [22].
- Unlike other studies [22], [23], the effect of supervised task, rather than unsupervised component, is significantly reduced in the last training iterations.
- Unlike other semi-supervised systems with joint training schemes [22], [23], [32], balancing parameters are considered for both unsupervised and supervised components.

5 Experimental Results

In this section, we will evaluate the performance of our two-step trip segmentation and SECA model on large-scale GPS trajectory data. First the dataset along with the data cleaning and preparation steps are discussed. Then, various supervised and semi-supervised baseline models that are used for performance comparison are described. The prediction performance of our model is evaluated using widely-used classification metrics. The proposed training strategy and the overall model architecture are also evaluated against several alternative approaches.

5.1 Experimental Setup

5.1.1 Dataset Description and Data Pre-Processing

The proposed model is examined and validated on the GPS trajectories collected by 182 users in the GeoLife project. The raw dataset contains 17,621 trajectories with a total distance of 1,292,951 kilometers and a total duration of 50,176 hours [33]. To the best of our knowledge, this is the only public dataset with GPS trajectories that have also been annotated with transportation modes. 69 users have labeled their trajectories with transportation modes while the remaining users left their trajectories unlabeled. It is worth noting that not all trajectories of those 69 users were annotated, and the trajectories for which the annotation was missing were considered to be unlabeled data. Although many kinds of transport modes have been labeled by the users, only transport modes that constitute significant portion of the dataset are considered for our analysis. Our transportation mode list is Y = {walk, bike, bus, driving, train}. The time-interval threshold for dividing a user's GPS trajectory T into trips and the maximum number of GPS points in a SE (i.e., M) are set to 20 minutes and 248, respectively.

Furthermore, we identify and remove erroneous GPS points that have been generated due to errors in sources such as satellite or receiver clocks. Every GPS SE is filtered by the following data processing steps:

- A GPS point with the timestamp greater than its next GPS point is identified and discarded.
- For labeled trajectories, a GPS point whose speed and/or acceleration do not fall within a certain and realistic range of its transportation mode, provided in Table 1, is identified and discarded.
- For unlabeled trajectories, due to lack of knowledge on transport modes, any GPS point of a segment that its speed and/or acceleration fall 1.5 times (or more) the interquartile range either above the third quartile or below the first quartile is identified and discarded.
- After removing the unrealistic GPS points, a segment with (1) the number of GPS points, (2) the total

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Number of Labeled GPS SE for Each Transportation Mode, Number of Unlabeled GPS SE, as Well as the Maximum Speed and Acceleration Associated with Each Transportation Mode

Mode	No. of SE	Max. $S(m/s)$	Max. $A(m/s^2)$
Walk	6640	7	3
Bike	3808	12	3
Bus	6051	34	2
Driving	4323	50	10
Train	3287	34	3
Total Labeled	24109	NA	NA
Unlabeled	72506	NA	NA

NA: Not Applicable.

distance, or (3) total duration less than specified thresholds are identified and discarded. In this study, these thresholds are set to 20, 150 meters and 1 minute, respectively.

The maximum allowable speed and acceleration pertaining to each mode are provided in Table 1, which have been defined using several reliable online sources and the engineering justification (e.g., existing speed limits, current vehicle/human's power).

Using the above-mentioned settings and the true change points, the distribution of the 4-channel labeled SE among various modes and the number of 4-channel unlabeled SE are listed in Table 1.

5.1.2 Baseline Methods

We compare the performance of the proposed model with two sets of baseline methods: (1) supervised algorithms, and (2) semi-supervised algorithms.

With respect to the *supervised group*, widely used standard supervised algorithms in the literature of transportation mode detection are deployed for comparison, including K-Nearest Neighbors (KNN), RBF-based Support Vector Machine (SVM), Decision Tree (DT), and Multilayer Perceptron (MLP). The hand-crafted features introduced in [3], [4] are passed into these supervised algorithms, as the most acceptable manual GPS trajectories' attributes available in the literature. These features include the GPS segment's total distance, mean speed, expectation of speed, variance of speed, top three speeds, top three accelerations, heading change rate, stop rate, and speed change rate. After calculating motion features of every $p \in SE$ using Eqs. (1), (2), (3), (4), and (5), these manual-designed features can be simply computed for a GPS SE using the definitions provided in [3], [4].

In addition, two supervised deep learning models are also used as baselines: (1) CNN classifier with the same settings as in the proposed model, (2) Recurrent Neural Networks (RNN) with the long short-term memory (LSTM) module. The number of repeating modules is equal to the length of the 4-channel tensor (i.e., M) while the current input for each module is a feature vector corresponding to every GPS point p, where the feature vector contains RD_p, S_p, A_p, and J_p. According to the hyperparameter tuning analysis, the combination of one LSTM layer with 50 units in each LSTM module yields the best RNN performance. RNN is an important baseline since it has widely been used for modeling trajectory data in recent years [6], [34].

With regards to the *semi-supervised group*, two distinct baselines are used: Semi-Two-Steps and Semi-Pseudo-Label, which are categorized as two-step and joint training techniques, respectively, as described in Section 2.

- *Semi-Two-Steps:* First, the Conv-AE is trained on both labeled and unlabeled trajectories. Then, the labeled data are transformed to the latent representation using the encoder part. In the second step, the transformed data are trained using a standard supervised algorithm, which is a logistic regression (i.e., the $softmax$ layer) in our case. The loss functions for the Conv-AE and logistic regression are given in Eqs. (8) and (9), respectively.
- *Semi-Pseudo-Label*: Two CNN classifiers with the same structures and shared layers are simultaneously trained in a supervised fashion, one on labeled and the other on unlabeled data. Pseudo label, Y_{ul}, is the predicted probability distribution over labels for an unlabeled sample $\mathbf{X}_{ul}$, using the updated weights from the previous training iteration. The overall loss function for this strategy is defined as follows:

$$l^{semi-pseudo} = \alpha l^{pseudo-classifier} + \beta l^{labeled-classifier}$$
$$l^{pseudo-classifier} = -\sum_{i=1}^{K} y_{ul,i} \log\left(p_{ul,i}\right), \tag{11}$$

where $y_{ul,i} \in Y_{ul}$ and $p_{ul,i} \in P_{ul}$ are the predicted probability for the class i based on the updated weights in the previous and current training iteration, respectively. Analogous to our SECA model, α and β are the balancing parameters. Further details about this approach is available in [24].

5.1.3 Performance Evaluation

The performance of our proposed trip segmentation is measured using precision and recall metrics, which are defined as below in the context of change point detection:

$$Precision = \frac{TP}{|\widehat{CP}|} \text{ and } Recall = \frac{TP}{|CP|},$$

where $|\widehat{CP}|$ and $|CP|$ are the number of predicted and true change points related to a GPS SE. TP is the number of true positives. A true change point is considered as a true positive if a predicted change point is found within a certain margin from the true change point. We set the margin to 20 GPS points, which is the same as the minimum of number allowed GPS points in a SE.

The performance of our proposed model is measured using two common classification metrics:

- *Accuracy* - it is computed as the fraction of SEs in the test set that are correctly classified.
- *Weighted F-measure* - F-measure for every transportation mode is defined as the harmonic average of its precision and recall in the test set, as shown below:

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

Weighted F-measure is the weighted average of F-measure for each mode, in which the corresponding

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Comparison of Accuracy Values for Different Supervised and Semi-Supervised Models with Varying Amounts of Labeled Data

	Proportion of labeled SE in the training data				
Model	10%	25%	50%	75%	100%
Supervised-KNN	0.469 (±0.015)	0.508 (±0.012)	0.549 (±0.014)	0.567 (±0.015)	0.579 (±0.015)
Supervised-SVM	0.417 (±0.006)	0.460 (±0.005)	0.470 (±0.006)	0.517 (±0.009)	0.532 (±0.010)
Supervised-DT	**0.661 (±0.014)**	0.672 (±0.013)	0.678 (±0.017)	0.689 (±0.012)	0.694 (±0.014)
Supervised-MLP	0.274 (±0.093)	0.309 (±0.107)	0.331 (±0.036)	0.347 (±0.069)	0.354 (±0.084)
Supervised-CNN	0.568 (±0.044)	0.617 (±0.042)	0.687 (±0.021)	0.719 (±0.019)	0.741 (±0.024)
Supervised-RNN	0.425 (±0.032)	0.428(±0.028)	0.431 (±0.027)	0.458 (±0.027)	0.461 (±0.039)
Semi-Two-Steps	0.544 (±0.019)	0.562 (±0.035)	0.588 (±0.016)	0.600 (±0.012)	0.605 (±0.015)
Semi-Pseudo-Label	0.589 (±0.020)	0.663 (±0.023)	0.707 (±0.021)	0.733 (±0.021)	0.754 (±0.018)
SECA (ours)	0.629 (±0.010)	**0.693 (±0.019)**	**0.732 (±0.017)**	**0.750 (±0.016)**	**0.768 (±0.016)**

All unlabeled data are used for training.

TABLE 3
Comparison of Weighted F-Measure Values for Various Supervised and Semi-Supervised Models with Varying Amounts of Labeled Data

	Proportion of labeled SE in the training data				
Model	10%	25%	50%	75%	100%
Supervised-KNN	0.440 (±0.017)	0.488 (±0.017)	0.531 (±0.017)	0.551 (±0.017)	0.564 (±0.017)
Supervised-SVM	0.337 (±0.003)	0.385 (±0.008)	0.429 (±0.013)	0.455 (±0.017)	0.476 (±0.018)
Supervised-DT	**0.662 (±0.014)**	0.672 (±0.014)	0.678 (±0.017)	0.689 (±0.012)	0.695 (±0.014)
Supervised-MLP	0.194 (±0.090)	0.227 (±0.129)	0.269 (±0.043)	0.252 (±0.064)	0.266 (±0.094)
Supervised-CNN	0.533 (±0.063)	0.560 (±0.044)	0.678 (±0.022)	0.710 (±0.020)	0.734 (±0.026)
Supervised-RNN	0.318 (±0.044)	0.325 (±0.039)	0.336 (±0.032)	0.358 (±0.034)	0.369 (±0.032)
Semi-Two-Steps	0.512 (±0.026)	0.541 (±0.038)	0.574 (±0.016)	0.584 (±0.014)	0.589 (±0.019)
Semi-Pseudo-Label	0.582 (±0.020)	0.654 (±0.025)	0.701 (±0.022)	0.728 (±0.021)	0.749 (±0.019)
SECA (ours)	0.615 (±0.011)	**0.683 (±0.019)**	**0.725 (±0.017)**	**0.745 (±0.017)**	**0.764 (±0.017)**

All unlabeled data are used for training.

weight for each mode is the proportion of SE from that mode in the test set. For each transportation mode y, precision shows the fraction of the true SEs with the label y among all SEs that are classified as y, while recall implies the ability of the model to correctly identify SEs with the true label y.

In all our experiments, models are trained and tested using stratified 5-fold cross-validation and average values (along with the standard deviations) of the results on all 5-folds are reported. Note that the stratified 5-fold cross-validation is only applied to labeled data. However, unless stated, the entire unlabeled data are used for training in semi-supervised models. Using stratified sampling, 10 percent of the labeled training data in each fold is selected as the validation set for the early-stopping procedure used within the deep learning-based models. All the described data processing and models are implemented within Python programming environment using `ruptures` for the trip segmentation, `TensorFlow` for deep learning models, and `scikit-learn` for classical supervised algorithms. All experiments are run on a computer with a single GPU. The source codes related to all data processing and models utilized in this study are available at https://github.com/sinadabiri/Deep-Semi-Supervised-GPS-Transport-Mode

5.2 Performance Comparison Results

First, the performance of our SECA model is assessed without taking the segmentation process into account. In other words, the labeled GPS SEs are created based on the true change points rather than the predicted ones. Next, our proposed trip segmentation is evaluated. Finally, the impact of the segmentation process on the overall performance is evaluated.

5.2.1 SECA Evaluation

Tables 2 and 3 provide the performance results of our SECA model and baseline methods in terms of accuracy and weighted F-measure, respectively. Every model is trained using various amounts of labeled data so as to investigate the effectiveness of the models when different amounts of labeled data is used.

Table 2 clearly shows the superiority of our SECA model and its training strategy in comparison with other baselines. Except for 10 percent labeled data that DT works better, our semi-supervised model consistently outperforms other methods for all percentages of labeled data. With respect to supervised algorithms, it is apparent that only CNN and DT are competitive as the test accuracy for other traditional learning methods are considerably low. What is interesting about the result is the low-quality of MLP and RNN, which also indicates that employing deep learning architectures does not always result in a better performance compared to shallow structures. Comparison between CNN and RNN clearly shows that capturing the local correlation between adjacent GPS points by the convolution operation generates more efficient features in comparison with the attempt to

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

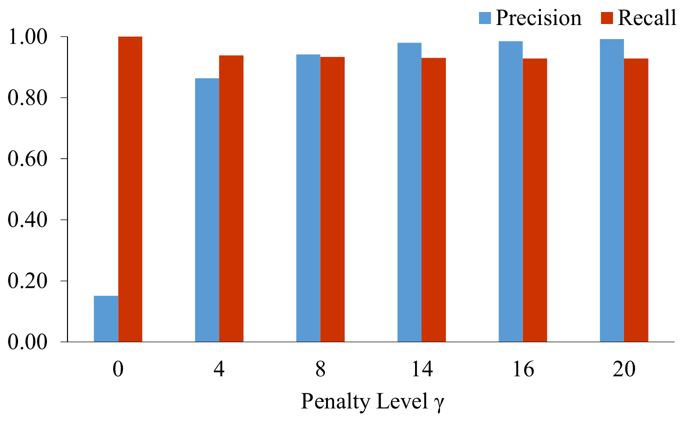

Fig. 6. Precision and recall values for the proposed trip segmentation process with different values of the penalty level γ.

learn the long-term dependency between all GPS points. In comparison between supervised and semi-supervised algorithms, the results indicate that semi-supervised techniques perform better than their supervised counterparts. Indeed, modeling the distribution of input data through unsupervised learning techniques with the help of unlabeled data can potentially ameliorate the generalization ability of the supervised task. Focusing on only semi-supervised algorithms, it is obvious that the prediction quality of joint training strategies (i.e., SECA and Semi-Pseudo-Label) are significantly higher than the Semi-Two-Steps. Such evidence confirms that, with aid of efficient initialization techniques, simultaneously training the reconstruction and classification abilities of a semi-supervised model yields a better performance compared to disjoint training strategies. For an overall comparison, our SECA model achieves on average 6.2 percent higher accuracy compared to other methods over different amounts labeled data, excluding the supervised algorithms (i.e., KNN, SVM, MLP, and RNN), which have not obtained competitive results in our problem.

Achieving high accuracy is only a positive starting point for having a reliable classifier. An effective and unambiguous way for evaluating the performance of a classifier is to use other measures such as precision, recall, and F-measure. As can be seen in Table 3, weighted F-measure for our SECA model is also superior to all other algorithms. This evidently confirms our findings and reasoning based on results in Table 2.

5.2.2 Trip Segmentation Evaluation

Fig. 6 shows the performance of our two-step trip segmentation process in terms of precision and recall for different values of the penalty level γ. The results clearly indicate the effectiveness of our proposed approach by achieving the very high values of 0.99 and 0.93 for precision and recall, respectively. As expected, absence of the penalty function in Eq. (6) causes overfitting since it allows the search algorithm choosing more and more change points in order to decrease the first component in the right side of Eq. (6). Predicting a high number of false change points results in a low precision value. This issue can be controlled by increasing the value of γ in order to constrain the number of predicted change points. Interestingly, while the precision value is dramatically improving (i.e., almost closing to 1) by raising the γ value, the recall value is dropping down very slightly. The rationale behind such a behavior is that the number of true change points in each GPS SE is very low after implementing the first step of the trip segmentation process. Our analysis indicates that more than 99 percent of the GPS SE contains less than 2 change points after the first step segmentation, which also implies the importance of the first step. Therefore, using larger values of γ forces the search algorithm to predict a few yet correct number of change points, which in turn results in high value of both precision and recall.

5.2.3 Overall Performance Evaluation

Table 4 shows the accuracy and F-measure results of our SECA model for three trip segmentation scenarios: (1) Trips are segmented based on the true change points. In other words, we assume the change points are already known, which leads to have the same result as in Tables 2 and 3, (2) Trips are uniformly partitioned into segments with the minimum number of GPS points, which is 20 in our setting. This scenario guarantees that all GPS SEs have only one transportation as SEs contains only the minimum allowed GPS points, (3) Trips are partitioned based on our proposed two-step trip segmentation process, which represents our whole mode detection framework (i.e., the SECA model followed by the results of the two-step trip segmentation). As can be seen from Table 4, the overall performance of our proposed framework is degraded by only 4 percent (on-average) compared to when the true change points are known. This implicitly indicates the acceptable performance of our trip segmentation. However, naively partitioning trips into uniform-size segments with the minimum number of GPS points (i.e., the second scenario) leads to lose many GPS information in a segment and in turn making mode detection more difficult for the SECA model. The performance metrics for the second scenario decreases by on average 10 percent compared to the first scenario, which is 6 percent less than the overall performance of our proposed framework.

TABLE 4
Comparison of Accuracy (Acc.) and F-Measure (F1) for our SECA Model while Different Trip Segmentation Scenarios Are Applied

	Proportion of labeled SE in the training data									
	10%		25%		50%		75%		100%	
Trip Segmentation	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
True Change Points	0.629	0.615	0.693	0.683	0.732	0.725	0.750	0.745	0.768	0.764
Uniform Size	0.536	0.514	0.584	0.572	0.630	0.623	0.646	0.637	0.671	0.664
Two-step Segmentation (ours)	0.600	0.589	0.650	0.641	0.680	0.673	0.704	0.700	0.721	0.717

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

TABLE 5
Comparison of Accuracy Values for Different Hyperparameter Schedules Along with Different Sizes of Labeled Data

#	Schedule for α and β		Proportion of labeled data in the training set				
	stage 1	stage 2	10%	25%	50%	75%	100%
1	$\alpha:1\rightarrow 0.1$ $\beta:1$	NA	0.603 (±0.018)	0.675 (±0.015)	0.709 (±0.008)	0.712 (±0.020)	0.747 (±0.022)
2	$\alpha:1$ $\beta:1\rightarrow 0.1$	NA	0.618 (±0.017)	0.670 (±0.024)	0.715 (±0.015)	0.721 (±0.017)	0.741 (±0.012)
3	$\alpha:1$ $\beta:1$	NA	0.625 (±0.018)	0.667 (±0.016)	0.716 (±0.015)	0.734 (±0.019)	0.745 (±0.022)
4	$\alpha:1$ $\beta:0$	$\alpha:1$ $\beta:1$	0.551 (±0.021)	0.564 (±0.214)	0.703 (±0.028)	0.715 (±0.023)	0.739 (±0.024)
5	$\alpha:0$ $\beta:1$	$\alpha:1$ $\beta:1$	0.590 (±0.023)	0.671 (±0.022)	0.711 (±0.025)	0.732 (±0.023)	0.755 (±0.022)
6 (ours)	$\alpha:1$ $\beta:1$	$\alpha:1$ $\beta:0.1$	**0.629 (±0.010)**	**0.693** (±0.019)	**0.732** (±0.017)	**0.750** (±0.016)	**0.768** (±0.016)

$1\rightarrow 0.1$: *Gradually decreasing from 1 to 0.1 over training iterations.*

TABLE 6
Comparison of Accuracy and Weighted F-Measure for Various Feature Combinations

Single Feature	Accuracy	F-measure	Feature Combination	Accuracy	F-measure
RD	0.529 (±0.119)	0.492 (±0.119)	$RD+S$	0.702 (±0.022)	0.691 (±0.023)
Δt	0.375 (±0.054)	0.352 (±0.087)	$RD+S+A$	0.763 (±0.016)	0.758 (±0.016)
S	0.702 (±0.021)	0.691 (±0.022)	$RD+\Delta t+S+A$	0.755 (±0.018)	0.750 (±0.022)
A	0.481 (±0.169)	0.415 (±0.022)	$RD+\Delta t+S+A+J+BR$	0.752 (±0.026)	0.748 (±0.026)
J	0.275 (±0.000)	0.119 (±0.000)	$RD+S+A+BR$	0.752 (±0.018)	0.741 (±0.018)
BR	0.295 (±0.019)	0.178 (±0.056)	$RD+S+A+J$	**0.768 (±0.016)**	**0.764** (±0.017)

5.3 Analysis and Discussion

In this section, we assess the quality of our proposed framework in several other aspects including the hyperparameter schedule, the data structure for GPS SEs, the SECA architecture, and the prediction capability for every transportation mode.

5.3.1 Balancing Parameters Schedule

We will now discuss the proposed schedule for tuning the balancing parameters α and β. Table 5 presents the test accuracy of the SECA model, according to several schedules for tuning hyperparameters in the loss function. In the schedule #1, the hyperparameter α gradually decreases from 1 to the minimum value 0.1 while the hyperparameter β is fixed to 1 during training. In fact, this schedule gradually shifts the focus solely on the supervised component over the training iterations. Schedule #2 is analogous to the schedule #1, yet with keeping the focus on the unsupervised task. Schedule #3 maintains the balancing parameters equal to 1 during the entire training process. Note that the schedules #1, #2, and #3 have only one stage and the training is stopped when no further improvement is achieved after two consecutive epochs. Schedules #4 and #5 have two stages. In the first stage, only one component (i.e., either supervised or unsupervised component) is trained until the early stopping criterion terminates the training process. Then, in the second stage, the training continues by reversing the focus to the part that has not been trained in the first stage. Training at this stage is stopped until no further improvement is achieved after two consecutive epochs. From Table 5, it can be seen that our proposed schedule for tuning balancing parameters improves the model performance more than other alternatives. The results reveal that deploying an effective tuning schedule can simply increase the model accuracy. It is worth noting that the schedules #1-#5 are the best examples with the highest accuracy among many other possible schedules.

Furthermore, the performance measures of Semi-Pseudo Label in Tables 2 and 3 supports the effectiveness of our proposed hyperparameter schedule. Note that Semi-Pseudo-Label is the most competitive technique to our approach and is jointly trained by varying its balancing parameters according to our proposed schedule.

5.3.2 Feature Analysis for GPS SE Representation

The quality of our proposed layout for representing motion features of a SE, shown in Fig. 3, is rigorously evaluated by tracking the effectiveness of various motion-feature combinations. Table 6 summarizes the performance comparisons when the input tensor is created using a single or a combination of features, as described in Section 3.2. The number of motion features determines the number of channels in the tensor. For example, an input tensor with only S information contains one channel rather than four. First, each feature type is independently examined in order to detect the most salient feature types, which in turn helps in constructing better feature combinations. As observed in Table 6, speed (S), relative distance (RD), and acceleration (A) are the most effective features when used in a standalone setting. The information obtained from tensors with single features leads to examine more reliable feature combinations. The combinations of the first two, the first three, and the first four important features (i.e., $RD+S$, $RD+S+A$, $RD+\Delta t+S+A$, respectively) are the most reasonable initial selections. A 6-channel tensor with all feature types is another selection in Table 6. Nonetheless, our configuration, that fuses relative distance, speed, acceleration, and jerk, attains the best performance compared to other potential good configurations. We also replace the jerk with bearing rate (i.e., $RD+S+A+BR$) so as to integrate the heading direction with kinematic information, but no further improvement is achieved. It is worth noting that Table 6 encapsulates the information for a few yet potential combinations that have attained the highest accuracy.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

TABLE 7
Evaluation of the Model Configuration of the Proposed SECA Method by Varying the Number of Convolutional Layers Across Different Amounts of Labeled SE in the Training Data

No.	% labeled SE in the training data					
Layers	10%	25%	50%	75%	100%	Average
2	0.621	0.680	0.723	0.741	0.751	0.703
4	0.629	0.693	0.732	0.750	0.768	**0.714**
6	0.629	0.693	0.732	0.750	0.759	0.712
8	0.629	0.698	0.725	0.743	0.762	0.711

TABLE 8
Confusion Matrix for our SECA Model Where Prec. and Rec. Correspond to Precision and Recall, Respectively

		Predicted Modes					
		Walk	Bike	Bus	Drive	Train	Rec.
True Modes	**Walk**	1269	43	16	0	0	0.96
	Bike	110	577	66	6	3	0.75
	Bus	173	41	927	49	20	0.76
	Drive	95	27	159	533	51	0.61
	Train	57	13	60	58	469	0.71
	Prec.	0.74	0.82	0.75	0.82	0.86	

5.3.3 Analysis of Model Architecture

The depth of neural networks is a key hyperparameter in deep architectures. Thus, the structure of our SECA model is evaluated in terms of depth by steadily increasing its depth through adding convolutional layers. Analogous to the SECA structure, depicted in Fig. 4, the number of filters starts with 32 for the first two convolutional layers and increases by a factor of 2 after adding every two convolutional layers. Every two convolutional layers are followed by a max-pooling layer. Other parameters (e.g., filter size) are fixed throughout the network. Table 7 shows the average accuracy values in 5-fold cross-validation for our SECA model with 2, 4, 6, and 8 convolutional layers by varying amounts of labeled SE. The last column is the average over all amounts of labeled data. It can be seen that increasing the number of layers up to the 4 layers enhances the model accuracy by around 1 percent, whereas adding additional layers does not result in a substantial improvement. Accordingly, we stop at a model with 4 convolutional layers (i.e., our proposed SECA model) so as to reduce the computation time.

5.3.4 Prediction Capability Per Transport Mode

As our last round of evaluation, we delve into the confusion matrix to analyze the high-level prediction ability of our SECA model for every transportation mode. Table 8 illustrates the confusion matrix, as well as precision and recall pertaining to each transportation mode for a test set. As expected, there is a high correlation between prediction quality and the number of available SE for a mode in the training set. As shown in Table 1, the walk and driving modes constitute the largest and smallest portions of the GPS SEs, which in turn results in the best and worst prediction performance for walk and driving modes, respectively. Nonetheless, the discriminating moving pattern of walk compared to others is another principal reason in achieving a perfect recall value for the walk mode. In addition to the lack of labeled driving SEs, the poor performance of the model in estimating the driving mode stems from several other factors including the possibility for driving in alternative routes with different speed limits, the presence of various types of drivers' behavior, and the flexibility of drivers in maneuvering. On the other hand, bus and train are the transit modes that must adhere to pre-defined routes and schedule, which leads to more predictable mobility behavior. Another interesting yet reasonable finding is that a large portion of false negative SEs for the driving mode is bus since bus is the most identical mode to car and taxi. Analogously, a majority of bike SEs has been falsely classified as walk as the moving pattern of bike is closer to walk compared to other modes. Such misclassifications calls for more labeled training SE so as to improve the discrimination ability of our SECA model.

6 Conclusion

We proposed a two-step trip segmentation and semi-supervised convolutional autoencoder (SECA) model for identifying transportation modes from GPS trajectory data. First, the raw GPS trajectory of a user's trip was partitioned into GPS segments with only one transportation mode. Next, every GPS segment was converted into an efficient 4-channel tensor so as to utilize the convolutional operation for automatically extracting high-level features rather than using hand-crafted features. The unsupervised and supervised components of the SECA model were simultaneously trained on both unlabeled and labeled GPS tensors while an optimal schedule was deployed for varying the balancing parameters between reconstruction and classification losses. Our extensive experiments demonstrated the superiority of the proposed trip segmentation process, the SECA model, the hyperparameter schedule, the representation for GPS trajectories, and the configuration of the model architecture compared to several baselines and alternatives.

As a future research direction, an end-to-end deep learning architecture can be designed to not only learn GPS representation from the raw GPS points using embedding approaches but also perform the trip-segmentation task through an initial layer of the deep learning model.

References

[1] L. Wu, B. Yang, and P. Jing, "Travel mode detection based on GPS raw data collected by smartphones: A systematic review of the existing methodologies," *Inf.*, vol. 7, no. 4, 2016, Art. no. 67.

[2] S. Dabiri and K. Heaslip, "Transport-domain applications of widely used data sources in the smart transportation: A survey," *arXiv:1803.10902*, 2018.

[3] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw GPS data for geographic applications on the web," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 247–256.

[4] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on GPS data," in *Proc. 10th Int. Conf. Ubiquitous Comput.*, 2008, pp. 312–321.

[5] 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, 2017, Art. no. 57.

[6] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3083–3090.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.

[7] 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.

[8] S. Dabiri and K. Heaslip, "Developing a twitter-based traffic event detection model using deep learning architectures," *Expert Syst. Appl.*, vol. 118, pp. 425–439, 2019.

[9] S. Dabiri and K. Heaslip, "Inferring transportation modes from GPS trajectories using a convolutional neural network," *Transp. Res. Part C: Emerging Technol.*, vol. 86, pp. 360–371, 2018.

[10] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Advances Neural Inf. Process. Syst.*, 2014, pp. 3581–3589.

[11] G. Li, C.-J. Chen, S.-Y. Huang, A.-J. Chou, X. Gou, W.-C. Peng, and C.-W. Yi, "Public transportation mode detection from cellular data," in *Proc. ACM Conf. Inf. Knowl. Manag.*, 2017, pp. 2499–2502.

[12] H. Menp, L. Andrei, and L. M. L. Jose, "Travel mode estimation for multi-modal journey planner," *Transp. Res. Part C: Emerging Technol.*, vol. 82, pp. 273–289, 2017.

[13] M. Rezaie, Z. Patterson, J. Y. Yu, and A. Yazdizadeh, "Semi-supervised travel mode detection from smartphone data," in *Proc. Int. Smart Cities Conf.*, 2017, pp. 1–8.

[14] Y. Endo, T. Hiroyuki, N. Kyosuke, and K. Akihisa, "Deep feature extraction from trajectories for transportation mode estimation," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2016, pp. 54–66.

[15] H. Wang, L. GaoJun, D. Jianyong, and Z. Lei, "Detecting transportation modes using deep neural network," *IEICE Trans. Inf. Syst.*, 2017, pp. 1132–1135.

[16] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 3546–3554.

[17] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using LSTM for region embeddings," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, 2016, pp. 526–534. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045447

[18] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2007, pp. 153–160.

[19] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, 2010.

[20] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statistics*, 2010, pp. 249–256.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] Y. Zhang, S. Dinghan, W. Guoyin, G. Zhe, H. Ricardo, and C. Lawrence, "Deconvolutional paragraph representation learning," in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 4172–4182.

[23] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2011, pp. 151–161.

[24] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. Workshop Challenges Representation Learn.*, 2013, vol. 3, Art. no. 2.

[25] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Survey Rev.*, vol. 23, no. 176, pp. 88–93, 1975.

[26] O. Bagdadi and A. Várhelyi, "Development of a method for detecting jerks in safety critical events," *Accident Anal. Prevention*, vol. 50, pp. 83–91, 2013.

[27] C. Truong, L. Oudre, and N. Vayatis, "A review of change point detection methods," *arXiv:1801.00718*, 2018.

[28] F.-L. Chung, T.-C. Fu, V. Ng, and R. W. Luk, "An evolutionary approach to pattern-based time series segmentation," *IEEE Trans. Evol. Comput.*, vol. 8, no. 5, pp. 471–489, Oct. 2004.

[29] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *J. Amer. Statistical Assoc.*, vol. 107, no. 500, pp. 1590–1598, 2012.

[30] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, vol. 1.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations, arXiv:1412.6980*, 2014.

[32] M. Ranzato and M. Szummer, "Semi-supervised learning of compact document representations with deep networks," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 792–799.

[33] Y. Zheng, F. Hao, X. Xing, M. Wei-Ying, and L. Quannan, "Geolife GPS trajectory dataset-user guide," https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide, 2011.

[34] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, "Trajectory clustering via deep representation learning," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 3880–3887.

Sina Dabiri received the MS degree in transportation engineering from the Sharif University of Technology, Tehran, Iran in 2012, the simultaneously MS degree from the Department of Computer Science and the PhD degree from the Department of Civil Engineering at Virginia Tech, in 2018. His research interests include machine learning and deep learning in intelligent transportation systems (ITS).

Chang-Tien Lu received the MS degree in computer science from the Georgia Institute of Technology, Atlanta, GA, in 1996, and the PhD degree in computer science from the University of Minnesota, Minneapolis, MN, in 2001. He is a professor of computer science and associate director of the Discovery Analytics Center at Virginia Tech. His research interests include spatial databases, data mining, urban computing, and intelligent transportation systems. He served as the vice chair of ACM SIGSPATIAL from 2011 to 2014. He is an ACM Distinguished Scientist.

Kevin Heaslip received the BS and MS degrees from Virginia Tech, and the PhD degree from the University of Massachusetts Amherst. He is an associate professor of civil engineering at Virginia Tech. He also serves as the associate director of the Information Systems Laboratory at the Hume Center for National Security and Technology. His research interests include the connections of transportation, smart cities, resilience, and cybersecurity. He is a registered professional engineer in New Hampshire with more than 15 years' experience in the transportation fields. He has also been a principal or co-principal investigator of projects funded at more than 18 million dollars.

Chandan K. Reddy received the MS degree from Michigan State University, and the PhD degree from Cornell University. He is an associate professor with the Department of Computer Science at Virginia Tech. His primary research interests include data mining and machine learning with applications to real-world problems. He has published more than 95 peer-reviewed articles in leading conferences and journals. He received several awards for his research work including the Best Application Paper Award at ACM SIGKDD conference in 2010, Best Poster Award at IEEE VAST conference in 2014, Best Student Paper Award at IEEE ICDM conference in 2016, and was a finalist of the INFORMS Franz Edelman Award Competition in 2011. He is an associate editor of the *ACM Transactions on Knowledge Discovery and Data Mining* and PC co-chair of ASONAM 2018. He is a senior member of the IEEE and life member of the ACM.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on June 01,2020 at 18:45:48 UTC from IEEE Xplore. Restrictions apply.