Int J Data Sci Anal (2016) 2:107–117
DOI 10.1007/s41060-016-0014-1

ORIGINAL ARTICLE

Classifying spatial trajectories using representation learning

Yuki Endo[1] · Hiroyuki Toda[1] · Kyosuke Nishida[1] · Jotaro Ikedo[1]

Received: 29 May 2016 / Accepted: 28 June 2016 / Published online: 12 July 2016

Abstract This paper addresses the problem of feature extraction for estimating users' transportation modes from their movement trajectories. Previous studies have adopted supervised learning approaches and used engineers' skills to find effective features for accurate estimation. However, such handcrafted features cannot always work well because human behaviors are diverse and trajectories include noise due to measurement error. To compensate for the shortcomings of handcrafted features, we propose a method that automatically extracts additional features using a deep neural network (DNN). In order that a DNN can easily handle input trajectories, our method converts a raw trajectory data structure into an image data structure while maintaining effective spatiotemporal information. A classification model is constructed in a supervised manner using both of the *deep* features and handcrafted features. We demonstrate the effectiveness of the proposed method through several experiments using two real datasets, such as accuracy comparisons with previous methods and feature visualization.

Keywords Movement trajectory · Representation learning · Deep learning · Transportation mode

This paper is an extension version of the PAKDD2016 Long Presentation paper "Deep Feature Extraction from Trajectories for Transportation Mode Estimation" [5].

✉ Yuki Endo
endo.yuki@lab.ntt.co.jp; endo-wop@hotmail.co.jp

1 NTT Service Evolution Laboratories, 1-1 Hikarinooka, Yokosuka-shi, Kanagawa-ken 239-0847, Japan

1 Introduction

Estimating users' contexts from their movement trajectories obtained from devices such as mobile phones with GPS is crucial for location-based services (e.g., Google Now[1] and Moves[2]) This paper focuses on a specific aspect of human movement, the transportation mode of individual users when they move. The ability to accurately determine the transportation mode on mobile devices will have a positive impact on many research and industrial fields, such as personalized navigation routing services [8] and geographic information retrieval [18]. According to previous studies [23–25], transportation mode estimation involves two steps: extraction of segments of the same transportation modes and estimation of transportation modes on each segment (see also Fig. 1a).

In estimating transportation modes, researchers have manually discovered effective features for supervised classification (e.g., movement distance, velocities, acceleration, and heading change rate [23–25]) using their skills. While this heuristic approach is basically important for discriminating between transportation modes, handcrafted features do not always work well because human behaviors are diverse, and movement trajectories also include various aspects. For example, movement distance and velocity, which are especially fundamental and effective features, depend on users' contexts even when they are using the same transportation mode. Such features are also susceptible to GPS measurement error which becomes larger especially in urban environments.

To compensate for the above shortcomings, we utilize additional features automatically extracted by *representation learning*. *Deep learning* [2,7] is a well-known example of

[1] http://www.google.com/landing/now/.
[2] https://play.google.com/store/apps/details?id=com.protogeo.moves.

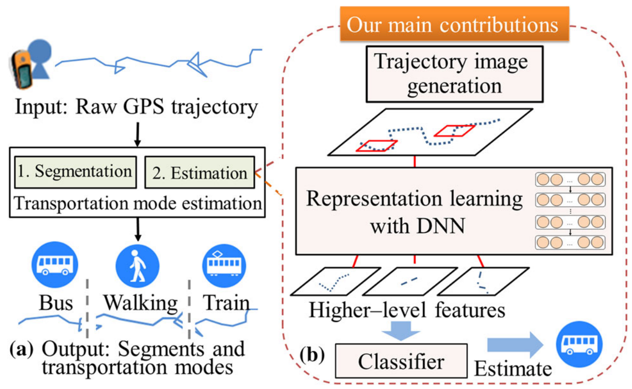

Input: Raw GPS trajectory
1. Segmentation
2. Estimation
Transportation mode estimation
Bus
Walking
Train
(a) Output: Segments and transportation modes
Our main contributions
Trajectory image generation
Representation learning with DNN
Higher–level features
(b)
Classifier
Estimate

Fig. 1 Transportation mode estimation and our contributions

this, which learns a deep neural network (DNN) model with multiple intermediate layers and can automatically extracts effective higher-level features for tasks from lower-level features of input data. Recently, this technique fundamentally improved performance in some fields including image recognition [9] and speech recognition [4].

The effectiveness of deep features for a task depends on an input data structure. For example, while raw pixel values are often used as input of a DNN for image data [7,9], spectrograms are calculated from raw signals for audio data so that a DNN can easily handle them [4]. These approaches cannot be directly adapted to the locational information which has a different data structure (a series of latitude, longitude, and timestamp) from image and audio data. Consequently, how to apply deep learning to locational information has not been properly studied.

We propose a method that extracts features from raw GPS trajectories using deep learning. Our key idea is to represent GPS trajectories as 2D image data structures (called *trajectory images*) and use these trajectory images as input of deep learning. This is based on the knowledge that deep learning works well in the field of image recognition. For example, a DNN can detect local semantic attributes of images, such as skin patterns and tail shapes of animals, as human can understand by looking them. This is because a DNN has a structure that approximates the operation of the neocortex of a human brain, which is associated with many cognitive abilities [1]. Our assumption is that a DNN can suitably detect particular attributes from the trajectory images: Movement trajectories inherently contain 2D spatial information that is more naturally perceivable for a human brain (i.e., a DNN) rather than simple latitude, longitude, and timestamp values.

We also propose a supervised framework for transportation mode estimation, which includes our feature extraction method from trajectory images. As illustrated in Fig. 1b, the framework first generates trajectory images from given GPS trajectory segments. After trajectory images are generated, higher-level features are extracted using a fully connected DNN with stacked denoising autoencoder (SDA) [21], which is a representative method of deep learning. Intuitively, higher-level features are obtained by appropriately filtering trajectory images for picking up discriminative parts of the images. Finally, transportation modes are estimated using a classifier that is learned from the higher-level features and transportation mode annotations.

Our main contributions are summarized as follows:

– We propose a method for generating informative trajectory images for deep learning from raw GPS trajectories (Sect. 3).
– We propose a supervised framework for trajectory classification including feature extraction from trajectory images using deep learning (Sect. 4).
– Extensive evaluations are provided to confirm the effectiveness of our method using two real datasets (Sect. 5).

2 Related work

GPS trajectory mining An overview of trajectory data mining is outlined in a survey [22]. In particular, there have been many studies on trajectory mining tasks such as user activity estimation [6,13], transportation mode estimation [11,14,15,23–25], and movement destination estimation [17]. Several methods [6,13] use not only GPS trajectories as features, but also body temperature, heart rate, humidity, and light intensity obtained from other sensors, and construct a model for predicting user activities such as walking, running, cycling, and rowing. While these methods can estimate various user activities, users need to carry many devices. Estimating a user's context with few sensors is ideal to lighten his/her burden. Therefore, using sensor information other than GPS trajectories is out of the scope of this paper.

Liao et al. [11], Patterson et al. [14], and Shah et al. [15] reported on methods for estimating transportation modes, such as walking, bus, and car, using only GPS trajectories as sensor data. However, their methods require external information including a street map. Static information, such as a street map, might not be applied to the task because structures of cities dynamically change over time. We therefore do not target methods that require external information.

For an approach that does not use external information, Zheng et al. [25] proposed a method that can estimate transportation modes using only GPS trajectories. They describe a method for segmenting GPS trajectories by detecting change points of transportation modes on the basis of velocity and acceleration. Transportation modes are then estimated from features of segments using a classifier. Zheng et al. first presented basic features such as moving distance, velocity, and acceleration [23]. They also introduced advanced features including velocity change rate (VCR), stop rate (SR), and heading change rate (HCR), which achieved more accurate estimation [24]. While their method uses handcrafted features, our method tackles the problem of automatically extracting effective features from trajectory images.

Deep learning One of the major goals of deep learning is to obtain effective higher-level features from signal-level input using a DNN. For example, while traditional approaches for image recognition use handcrafted features such as scale-invariant feature transform [19], a DNN can automatically extract effective features from raw image pixels. In fact, it has been reported supervised learning with deep features can achieve high recognition accuracy [7].

Although a DNN has high expressiveness, learning a DNN model efficiently using conventional approaches is difficult due to a vanishing gradient problem. Specifically, back-propagation used to optimize a DNN does not sufficiently propagate a reconstruction error to deep layers, and the error vanishes midway through an intermediate layer. To solve this problem, greedy layer-wise training was proposed [2,7], and it has allowed the topic of deep learning to gain significant attention. This technique pre-trains parameters of intermediate layers layer by layer in an unsupervised manner before fine-tuning for the entire network. This enables error information to be efficiently propagated to deep layers and consequently improved performance in many tasks.

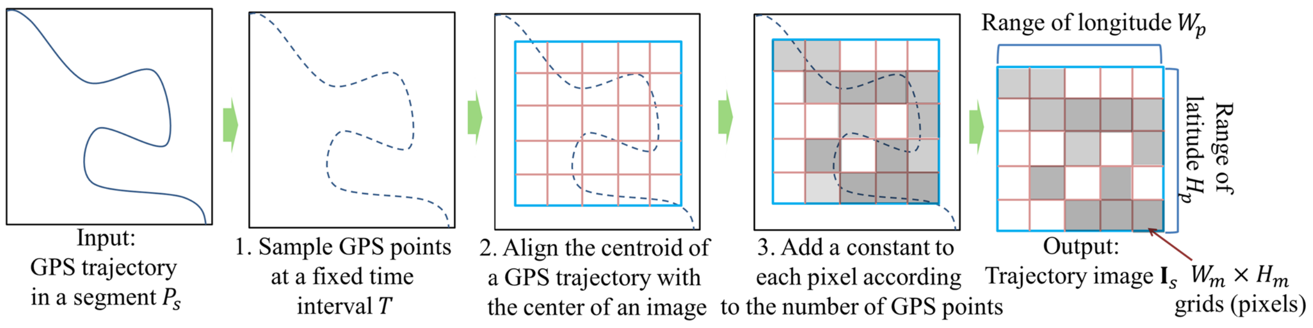
There are several techniques for deep learning such as deep belief nets (DBN) [7], deep Boltzmann machine (DBM) [3], and SDA [21] for pre-training. These and other techniques are outlined in a survey [3] that can be referred to for more information. In this paper, we adopt fully connected DNN with SDA for transportation mode estimation for the first time and demonstrate its effectiveness.

3 Trajectory image generation

There are several difficulties for generating informative trajectory images so that DNNs can discriminate between transportation modes. First, most of the DNNs must fix the dimensions of input vectors. That is, input images must be the same size when the pixel values are directly used as the input vectors. However, different-sized images are obtained by simply clipping an entire GPS segment when a spatial length of one pixel is fixed. The reason is that topographic ranges of the GPS segments differ, especially depending on transportation modes (walking is often narrow while a train is broad). Although one straightforward approach to solve this problem is to resize different-sized images to the same size, distance information in a trajectory is lost since each scale differs. Second, DNNs require sufficient as well as informative training data to improve its performance. If images are of high resolution (number of pixels is large), detailed movement can be obtained; however, the trajectory pixels (nonzero pixels) in the images become sparse, and such sparse images degrade the generalization capability of a DNN. As a result, many trajectory images are required in order to overcome this sparsity problem. If images are of low resolution (number of pixels is small), the sparsity problem is alleviated, but much information of GPS points corresponding to the same pixel is lost.

Based on the above, our trajectory image generation method consists of two steps: (1) determining the *target range* of a segment that is converted into a fixed size image and (2) determining the *number and value of pixels* of the image. For the first step, we simply clip a certain area from each segment. To do this, we define a rectangle region for clipping by ranges of latitude and longitude. Although information outside the defined region is lost, we verified that this method outperforms the resizing method through our experiments because distance information in a trajectory is preserved. For the second step, we use stay time to determine pixel values; i.e., the longer a user stays in the same pixel (a rectangular region), the higher the pixel value becomes. This manner can maintain temporal information of a segment with a small number of pixels and thus can alleviate the sparsity problem rather than using large binary images that maintain the details of movements.

An overview of trajectory image generation is shown in Fig. 2, and a specific procedure is described in Algorithm 1. We first define some terms used in our method. We refer to



Input: GPS trajectory in a segment P_s
1. Sample GPS points at a fixed time interval T
2. Align the centroid of a GPS trajectory with the center of an image
3. Add a constant to each pixel according to the number of GPS points
Range of longitude W_p
Range of latitude H_p
Output: Trajectory image $\mathbf{I}_s$
$W_m \times H_m$ grids (pixels)

Fig. 2 Overview of trajectory image generation

Algorithm 1 TRAJECTORYIMAGEGENERATION

```
Input: $P_s = (p^{(i)})_{i=1}^{N_s}$, $T$, $W_p$, $H_p$, $W_m$, $H_m$
Output: $\mathbf{I}_s$
Initialize: $\mathbf{I}_s \in \mathbb{R}^{W_m \times H_m} \leftarrow \mathbf{0}$
 1: $P'_s = \text{SAMPLINGPOINTS}(P_s, T)$
 2: $center_{lng} \leftarrow \frac{1}{|P'_s|} \sum_{j=1}^{|P'_s|} p^{(j)}.lng$
 3: $center_{lat} \leftarrow \frac{1}{|P'_s|} \sum_{j=1}^{|P'_s|} p^{(j)}.lat$
 4: $min_{lng} \leftarrow \min_j p^{(j)}.lng$
 5: $min_{lat} \leftarrow \min_j p^{(j)}.lat$
 6: $offset_x \leftarrow \lfloor \frac{W_m}{2} \rfloor - \lfloor (center_{lng} - min_{lng}) \frac{W_m}{W_p} \rfloor$
 7: $offset_y \leftarrow \lfloor \frac{H_m}{2} \rfloor - \lfloor (center_{lat} - min_{lat}) \frac{H_m}{H_p} \rfloor$
 8: FOR $j$ in 1 to $|P'_s|$ DO
 9:     $x \leftarrow \left\lfloor (p^{(j)}.lng - min_{lng}) \frac{W_m}{W_p} \right\rfloor + offset_x$
10:     $y \leftarrow \left\lfloor (p^{(j)}.lat - min_{lat}) \frac{H_m}{H_p} \right\rfloor + offset_y$
11:     IF $0 \le x < W_m$ AND $0 \le y < H_m$ THEN
12:         $\mathbf{I}_s(x, y) \leftarrow \mathbf{I}_s(x, y) + 1$
13:     ENDIF
14: ENDFOR
```

each data point given a positioning system as a *GPS point*. Given segment s as input, let $P_s = (p^{(i)})_{i=1}^{N_s}$ be a sequence of continuous GPS points, where N_s denotes the number of GPS points in the segment. Let $p^{(i)}$ represent the i-th GPS point and each GPS point be represented as a three-tuple $p^{(i)} = (lat, lng, t)$; latitude lat, longitude lng, and timestamp t. Let W_p and H_p denote ranges of longitude and latitude for pixelizing trajectories, respectively, whereas W_m and H_m denote width and height of images, respectively. Let T be a time interval for sampling GPS points from input GPS trajectories of P_s. $\mathbf{I}_s \in \mathbb{R}^{W_m \times H_m}$ denotes a generated trajectory image that has one-channel value (intensity) per pixel like a grayscale image.

To extract a trajectory image from a GPS trajectory in a segment, we first evenly sample GPS points from P_s. The GPS points in each segment are not always positioned at a fixed time interval due to differences in GPS sensors and signal quality. If sequential GPS points positioned at different time intervals are converted into trajectory images, short time intervals result in a long stay in one pixel even if a user stays in the pixel for a short time. We therefore sample GPS points from P_s at T intervals on the basis of timestamps $p^{(i)}.t$. If the next GPS point is not obtained after just T, we sample the nearest time GPS point. As a result, we obtain a sequence of the sampled GPS points and denote it as P'_s.

In the next step, the target range of a GPS trajectory in a segment is determined. For all segments, we compute the centroid of the sampled GPS points of P'_s using $p^{(i)}.lat$ and $p^{(i)}.lng$, and then align the centroid with the center of a trajectory image to unify the basic geographic coordinates. We define a clipped region as a rectangular area measuring W_p and H_p. The rectangular area is divided into $W_m \times H_m$ grids, and each grid corresponds to each pixel of the trajectory image. The number of pixels $W_m \times H_m$ is searched for using grid search, and the range of grid search is empirically determined as explained in the evaluation section. In Algorithm 1, offsets for aligning the centroid of a GPS trajectory with the center of an image are calculated on the basis of the centroid and the southwest-most GPS point in a segment.

Finally, GPS points of P'_s are then plotted when the GPS points exist in a defined grid. In fact, as shown in Algorithm 1, we calculate image coordinates corresponding to each GPS point using the offsets calculated previously and then plot GPS points on the images if those points are in the specified image size. When plotting GPS points, we add a constant $c = 1$ to the corresponding pixel to express the stay time in the pixel. After plotting all GPS points of P'_s on the segment s, trajectory image $\mathbf{I}_s$ is obtained.

Figure 3 shows several examples of trajectory images extracted from real GPS trajectories. The intensity of pixels indicates a user's stay time: the brighter the color, the longer the stay time. These trajectory images show that the images store distance information by clipping in the same range and represent time information through the pixel values. For instance, pixels near the center of the walking images become bright since the moving distance of walking is relatively short and the user stays in the same pixels for a long time. On the other hand, the images of bus and subway include rectilinear lines that are geographically widespread. Although there are such easy-to-understand features in the images, it is time-consuming and difficult to discover all features and quantify them. We therefore extract effective features from trajectory images using deep learning in the next section.

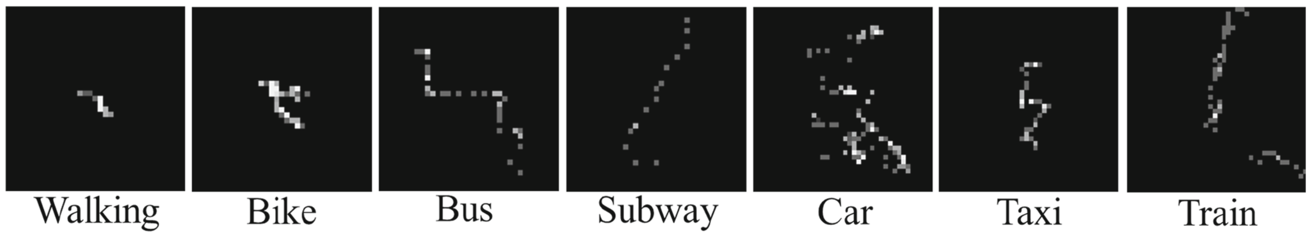
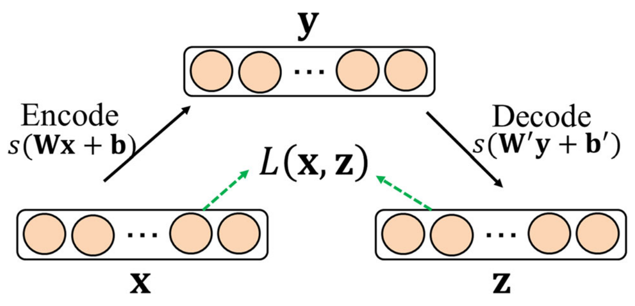
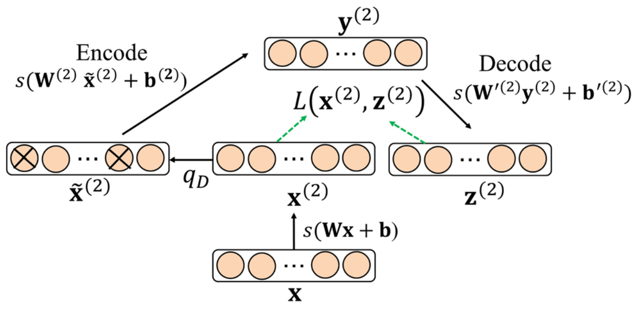


Fig. 3 Examples of trajectory images extracted from real GPS trajectories. *Brighter color* means longer stay time



$\mathbf{y}$
Encode
$s(\mathbf{Wx}+\mathbf{b})$
Decode
$s(\mathbf{W}'\mathbf{y}+\mathbf{b}')$
$L(\mathbf{x},\mathbf{z})$
$\mathbf{x}$
$\mathbf{z}$

Fig. 4 Structure of AE. Parameters are learned so that reconstruction error $L(\mathbf{x}, \mathbf{z})$ between input vector $\mathbf{x}$ and encoded and decoded vector $\mathbf{z}$ is minimized



$\mathbf{y}^{(2)}$
Encode
$s(\mathbf{W}^{(2)}\tilde{\mathbf{x}}^{(2)}+\mathbf{b}^{(2)})$
Decode
$s(\mathbf{W}'^{(2)}\mathbf{y}^{(2)}+\mathbf{b}'^{(2)})$
$L(\mathbf{x}^{(2)},\mathbf{z}^{(2)})$
$\tilde{\mathbf{x}}^{(2)}$
q_D
$\mathbf{x}^{(2)}$
$\mathbf{z}^{(2)}$
$s(\mathbf{Wx}+\mathbf{b})$
$\mathbf{x}$

Fig. 5 Structure of SDA. Intermediate representations at deepest layer at current stack of DAEs are used as input for new DAE when new layer is added. In this example, first intermediate representation $\mathbf{x}^{(2)}$ at second layer is used as input of newly arranged DAE, and weighting matrix $\mathbf{W}^{(2)}$ and bias term $\mathbf{b}^{(2)}$ are learned (where lower indices indicate layer number). Then, $\mathbf{y}^{(2)}$ is used as input of DAE at third layer (i.e., $\mathbf{x}^{(3)}$). This process is repeated until the number of layers reaches the given value

4 Deep feature extraction and classification

In order to avoid the vanishing gradient problem, we pre-train a DNN using SDA [21] in an unsupervised manner only from trajectory images before fine-tuning the DNN in a supervised manner. In this section, we first briefly review basic techniques of SDA, autoencoder (AE) [2], denoising autoencoder (DAE) [20], and SDA. Finally, we explain the detailed settings for applying the deep learning algorithm to trajectory images.

Autoencoder (AE) AE is a feed-forward neural network that learns identity mapping so that an input vector $\mathbf{x}$ is equal to an encoded and decoded vector $\mathbf{z}$ of the input vector (Fig. 4). Encoding nonlinearly transforms an input vector $\mathbf{x}$ and obtains intermediate representation $\mathbf{y}$ as follows:

$$\mathbf{y} = s(\mathbf{Wx} + \mathbf{b}), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{n_h \times n_x}$ and $\mathbf{b} \in \mathbb{R}^{n_h}$ are, respectively, the weighting matrix and bias term that transform vectors in n_x-dimensional space into vectors in n_h-dimensional space. The term $s(\cdot)$ is an activation function such as sigmoid or tanh. In a similar manner, decoding nonlinearly transforms an intermediate representation as follows:

$$\mathbf{z} = s(\mathbf{W}'\mathbf{y} + \mathbf{b}'), \tag{2}$$

where $\mathbf{W}' \in \mathbb{R}^{n_x \times n_h}$ and $\mathbf{b}' \in \mathbb{R}^{n_x}$ are, respectively, the weighting matrix and bias term that transform vectors in n_h-dimensional space into vectors in n_x-dimensional space. We can also use a constraint $\mathbf{W}' = \mathbf{W}^T$ called tied weights that has the effect of circumventing over-fitting by reducing the degrees of freedom of parameters. The weighting matrices and bias terms are estimated on the basis of a reconstruction error. If we assume $P(\mathbf{x}|\mathbf{z})$ is a Gaussian distribution, the reconstruction error is defined as the mean squared error:

$$L(\mathbf{x}, \mathbf{z}) = ||\mathbf{x} - \mathbf{z}||^2. \tag{3}$$

If we assume $P(\mathbf{x}|\mathbf{z})$ is a binomial distribution, we can use the cross entropy error as the reconstruction error. In this paper, we use the mean squared error because our inputs are real-valued vectors calculated from trajectory images and the co-domain of each element is not limited.

Denoising autoencoder (DAE) DAE is similar to AE except for that it adds noise to an input vector, and this enables the generalization capability of a DNN model to be improved. Specifically, on the basis of a distribution $q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x})$, noise is added to each element of an input vector, and a corrupted version of an input vector $\tilde{\mathbf{x}}$ is obtained. Parameters are estimated on the basis of the reconstruction error between the original input vector and a vector obtained by encoding and decoding noisy input $\tilde{\mathbf{x}}$.

Stacked denoising autoencoder (SDA) SDA is composed of multiple DAEs (Fig. 5) and used to initialize a DNN. For training with SDA, unsupervised training with DAE is done in a greedy layer-wise fashion. That is, when a new layer is added, the intermediate representations at the deepest layer at a current stack of DAEs are used as input for a new DAE. This pre-training helps to avoid the vanishing gradient problem for a DNN. After pre-training with SDA, we can use features obtained from the deepest layer of the pre-trained DNN as input for constructing classification models. To adjust the parameters of the entire network, fine-tuning is also applied to the pre-trained DNN with annotation information.

To use trajectory images as input of a fully connected DNN, we convert trajectory image matrices $\mathbf{I}_s$ into $W_m \times H_m$-dimensional vectors $\mathbf{x}_s$ by simply aligning each pixel value. The number of intermediate layers L of the DNN is determined by grid search as explained in the evaluation section. We use a sigmoid function $s(\cdot)$ as an activation function of each layer. To pre-train parameters (weighting matrices $\mathbf{W}^{(l)}$ and bias terms $\mathbf{b}^{(l)}$ at each intermediate layer l) of DNN using SDA [21], we use a minibatch L-BFGS method because of its effectiveness for classification problems [10]. After pre-training with SDA, supervised fine-tuning adjusts parameters of the entire DNN using annotated labels. For fine-tuning, an output sigmoid layer is added to the DNN, and parameters are updated using a stochastic gradient descent (SGD) method on the basis of the squared error between vectors on the output layer and binary vectors obtained from annotations. By using the learned DNN, higher-level features $\mathbf{x}^{(L+1)}$ are extracted from the deepest L intermediate layer of the DNN:

$$\mathbf{x}^{(l)} = \begin{cases} s(\mathbf{W}^{(l-1)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l-1)}) & (l > 1); \\ \mathbf{x}_s & (l = 1). \end{cases} \quad (4)$$

These image-based higher-level features are concatenated with the handcrafted features $\mathbf{x}_e$ (movement distance, mean velocity, etc.). Finally, we construct a classifier, such as logistic regression and support vector machine, using the concatenated features $[\mathbf{x}_{(L+1)}^T, \mathbf{x}_e^T]^T$ and annotated transportation mode labels.

5 Evaluation

5.1 Dataset

GeoLife (GL) We used a GeoLife dataset [23–25] published by Microsoft Research. The GPS trajectories in the dataset were basically positioned every 1–3 s, and 69 users annotated labels of transportation modes. We removed the data of users who have only ten annotations or fewer and used the data of 54 users for our experiments. Each annotation contains a transportation mode and beginning and end times of the transportation. In the experiments, we labeled each section of GPS trajectories between the beginning and end times with an annotation, and used these sections as a segment of the same transportation mode. Although there are 11 types of annotations, we used only seven (walking, bus, car, bike, taxi, subway, and train), because the other four are in too few trajectories, and 9043 segments were obtained.
Kanto trajectories (KT) To verify that our method works in other regions, we used other trajectory data collected in the Kanto area of Japan. The data contains 30 users' trajectories for 20 days obtained from a Nexus7 2012 with a GPS sensor. The trajectories were basically positioned every 3 s. Each trajectory was annotated with a label of the seven transportation modes (walking, bike, car, bus, taxi, motorcycle, and train). In this dataset, we additionally segmented each labeled segment at three-minute intervals, and 14,019 segments were obtained. This is because we assume the use of our method for a real-time application, which estimates transportation modes from sequential segments for a relatively short time window.

5.2 Compared methods

Feature extraction methods To evaluate our feature extraction method, we prepared the following baseline features and our features:

- Basic Features (BF) [23]: Ten-dimensional features such as velocity.
- BF + Advanced Features (AF) [24,25]: Thirteen-dimensional features including BF and advanced features (VCR, SR, HCR).
- BoVW (Bag of Visual Words): Image features extracted from trajectory images using Dense-SIFT [19].
- SDNN: Deep features extracted using a DNN from vectors simply consisting of a series of latitude, longitude, and movement time at each GPS point.
- IDNN: Deep features extracted using a DNN from trajectory images.
- BF + AF + IDNN: Features consisting of handcrafted ones (BF + AF) and deep ones of trajectory images (IDNN).

For SDNN, the dimensions of input vectors are fixed to be the same number as those of the trajectory images of IDNN. Since one GPS point consists of three-dimensional components (i.e., latitude, longitude, and movement time), when three times the number of GPS points in a segment is smaller than the fixed dimensions, the empty element of the vector is set to 0. When that value is larger than the fixed dimensions, the newer GPS points are discarded.
Classification methods To build a classifier for estimating transportation modes, supervised learning is done using the extracted features and transportation mode annotations. We compared three classification methods, logistic regression (LR), support vector machine (SVM), and decision tree (DT). The experiment showed that the effectiveness of the classification method differs according to the features. For BF and BF + AF, we used DT in the following experiments since DT obtains the highest accuracy [23–25]. For BoVW, we used SVM. For SDNN, IDNN, and BF + AF + IDNN, we used LR.

5.3 Evaluation method

As an evaluation metric, we use accuracy that is the ratio of segments of correctly estimated labels out of all segments. We used fivefold cross validation (CV) over users, that is, each dataset was divided into training segments of *80% users* and test segments of *20% users*, while previous studies [23–25] mentioned nothing about discriminating users. This is because the training data of the test users are not often obtained in a realistic scenario. The problem setting in our study is more difficult than the previous studies. This is because movement features depend on users due to their habits or environments, but their data cannot be trained, and we also handle more transportation modes than the previous studies.

For the GL dataset, we search for model parameters using grid search based on fivefold CV with training data (i.e., nested CV):

- For DT, the splitting criterion is selected from the Gini coefficient or entropy, and the maximum ratio of features used for classification is searched for from $\{0.1, 0.2, \ldots, 1.0\}$.
- For SVM, the rbf kernel is used, the trade-off parameter is searched for from $\{0.01, 0.1, 1, 10, 100\}$, and the kernel coefficient is searched for from $\{0.001, 0.01, 0.1, 1, 10\}$.
- For trajectory image generation, the interval of sampling GPS points T is searched for from $\{10, 30, 60, 120\}$ s, ranges of longitude W_p and latitude H_p from $\{0.01, 0.05, 0.1, 0.2\}$, and the image size $W_m \times H_m$ from $\{20 \times 20, 25 \times 25, 30 \times 30, 35 \times 35, 40 \times 40, 50 \times 50\}$.
- For the DNN, the number of intermediate layers L is searched for from $\{1, 2, \ldots, 5\}$ (often 3 performed best), the number of each layer's neurons from $\{10, 50, 100, 200\}$ (often 100 performed best). For fine-tuning, the learning rate is set to 0.1 and the number of epochs is searched for from $\{1, 2, \ldots, 15\}$.

For the KT dataset, we empirically set the parameters by referring to the parameters automatically determined for the GL dataset.

5.4 Performance of feature extraction

Overall performance Table 1 compares the accuracies of transportation mode estimation with our features and the other features. The bold font denotes the condition that yielded the highest accuracy. In the results for both datasets, the accuracy of IDNN is modestly higher than those of BF and BF + AF. This indicates that the features extracted from trajectory images using deep learning work at least similarly to the handcrafted features, without complicated features designing. IDNN also significantly outperformed BoVW, that is, deep learning is more effective than the common image feature extraction approach. In contrast, SDNN does not work well despite using deep learning. It implies that simply applying deep learning to almost raw trajectory data cannot extract effective features for this task. One possible reason is that raw trajectory data contain values that widely range in an absolute coordinate system. Finally, it can be seen that the proposed method with the handcrafted and deep features (i.e., BF + AF + IDNN) achieves the best performance among all the methods. In other words, our deep features make up for the deficiencies of the existing features.

Table 2 shows the results of other classification models in the GL dataset. Notably, the accuracies of LR for IDNN and BF + AF + IDNN were the highest among the three classifiers, and the values were significantly higher than those of BF, BF + AF, and BoVW. This demonstrates that fine-tuning with the output sigmoid layer works well because LR is based on a logistic sigmoid function. In other words, our method can appropriately exploit the expressiveness of the DNN.

Additionally, we analyzed accuracies of individual transportation modes in Fig. 6. As can be seen in the figure, the

Table 1 Performance comparison of transportation mode estimation

Features	GL dataset	KT dataset
BF	0.632 ± 0.025	0.771 ± 0.0040
BF + AF	0.648 ± 0.025	0.780 ± 0.0030
BoVW	0.602 ± 0.044	0.760 ± 0.015
SDNN	0.386 ± 0.014	0.474 ± 0.025
IDNN	0.663 ± 0.029	0.797 ± 0.0060
BF + AF + IDNN	$\mathbf{0.679 \pm 0.028}$	$\mathbf{0.832 \pm 0.0047}$

Table 2 Performance comparison of transportation mode estimation with each feature and three classifiers in GL dataset

Features	LR	SVM	DT
BF	0.458 ± 0.017	0.479 ± 0.013	0.632 ± 0.025
BF + AF	0.483 ± 0.021	0.524 ± 0.0088	0.648 ± 0.025
BoVW	0.579 ± 0.0055	0.602 ± 0.044	0.548 ± 0.0087
SDNN	0.386 ± 0.014	0.386 ± 0.014	0.362 ± 0.0096
IDNN	0.663 ± 0.029	0.649 ± 0.0053	0.626 ± 0.0045
BF + AF + IDNN	$\mathbf{0.679 \pm 0.028}$	0.660 ± 0.0058	0.659 ± 0.0028

Each bold font denotes the condition that yielded the highest accuracy on each experiment

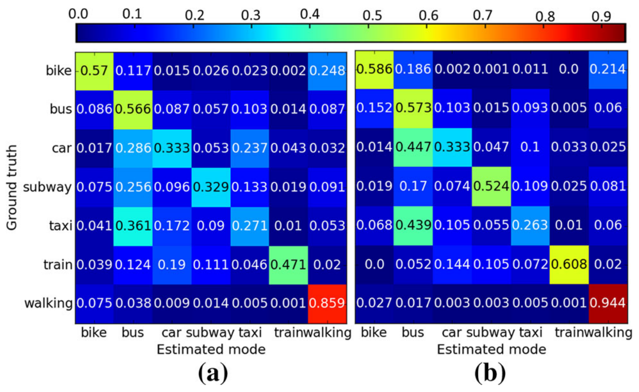


Fig. 6 Performance comparison on each transportation mode. **a** Confusion matrix of the previous method (BF + AF). **b** Confusion matrix of the proposed method (BF + AF + IDNN)

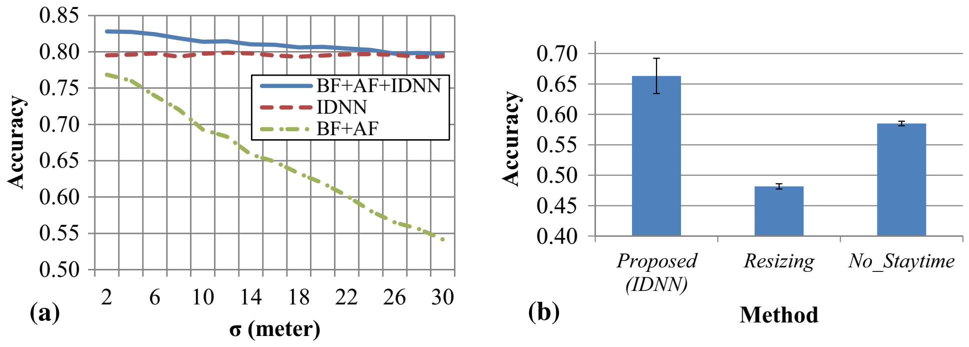


Fig. 7 **a** Accuracy with different noisy levels in KT dataset. **b** Performance comparison in GL dataset with each trajectory image generation method

deep image features (IDNN) improved the performance of estimating bike, subway, train, and walking. On the other hand, IDNN did not have significant effect on the accuracies of private car, bus, and taxi; that is, it was difficult to discriminate between these transportation modes because each mode is all a sort of automobile. However, IDNN improved the performance of discriminating between these automobiles and the other transportation modes. For example, IDNN increased the ratio of the car data that were estimated to one of these automobiles.

We implemented our method using Python.[3] The program was run on a PC equipped with a 2.66GHZ CPU and 48GB of memory. As for computational time in our method, training the DNN and classifier from the datasets took about 30 minutes, whereas estimation of transportation modes from a segment (including generating a trajectory image) took about 1 s.

Noise robustness We also evaluated our method's robustness against noise. For this purpose, we generated noisy trajectory data from the KT dataset. While the original dataset already contains some noise due to measurement error, the measurement can degenerate even more depending on the performance of a GPS sensor equipped in a mobile device and urban environments. For example, the KT dataset has about a 10-meter error on average according to the measurement accuracy reported from a function of Android OS. This value seems to be relatively low because we use devices with a relatively accurate GPS sensor (Nexus 7 2012), but all people do not have high-performance devices, and some people may also move in noisier environments. In fact, measurement accuracy may be worse than 100 meters in actual situations, while current positioning systems in smartphones are accurate to within 10 meters under ideal conditions [16]. We therefore evaluated noise robustness by adding some noise to the relatively clean trajectories in the KT dataset. The measurement is modeled as random Gaussian noise with zero mean and σ^2 variance [26].

Figure 7a shows the accuracy with different noisy levels in the KT dataset. In this experiment, we empirically fixed the DNN parameters for simplifying the experiment. The accuracy of BF + AF decreased with increasing noisy levels, whereas that of IDNN was barely affected by the noise. The accuracy of BF + AF + IDNN modestly decreased, but it only reached that of IDNN. This is because BF + AF does not work well when the noise level is high, but our DNN-based method is robust against measurement error.

There are two reasons our method is robust against measurement error. First, noise is reduced in the process of trajectory image generation. For example, if W_p and H_p are 0.01, the images are generated in the range of about 1000 square meters. When the image size $W_m \times H_m$ is 40 × 40, one pixel represents 25 meter square. Therefore, noise of tens of meters has an insignificant effect on trajectory image generation. Second, the DNN can automatically detect features from trajectory images even if the data have some noise. In particular, the DNN with SDA learns a model to be able to reconstruct denoised data from noisy data.

5.5 Effectiveness of image generation method

We now discuss the effectiveness of our method at generating trajectory images. Our image generation method does not use the information of the GPS points that are (1) outside of the defined region and (2) not sampled at T intervals, and (3) detailed latitude and longitude values (discretization into pixels).

For the validation of the first point, as shown in Fig. 7b, we compared the proposed method (*Proposed*), which maintains the scale of trajectories and also stores the stay time in image pixels, with the following two methods. One method (*Resizing*) generates different-sized trajectory images by clipping an entire region in each segment where a spatial range of one pixel is fixed to a small constant. It then resizes the different-sized images to the same size (i.e., $W_m \times H_m$) using the nearest neighbor method [12]. The stay time information is stored in the same way as with *Proposed*. The other method (*No_Staytime*) assigns the same constant value to pixels in which multiple GPS points exist. The scale is maintained

[3] We used the scikit-learn library to implement classifiers. http://scikit-learn.org/stable/.

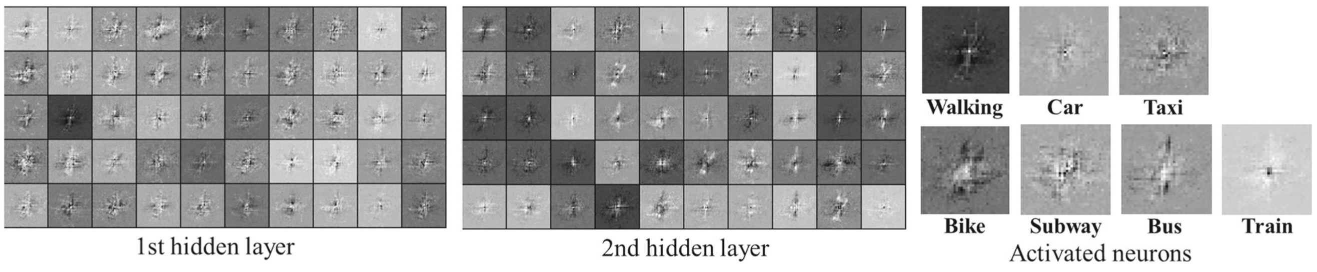


Fig. 8 Visualization results on state of activated neurons (weighting matrices) of learned DNN. Two *left* images show states of activated neurons on first and second intermediate layers, and seven *right* images show states of activated neurons strongly responding to data with labels of each transportation mode

in the same way as with *Proposed*. Obviously, *Proposed* performed best among the three methods, which suggests effectiveness of maintaining scale and storing stay time with our method.

Second, we evaluated our method at different sampling intervals from 10 to 120 s. We confirmed that smaller intervals (<60 s for the GL dataset) worsened the accuracy via the grid search (explained in Sect. 5.3). The GPS points in each segment are not always positioned at a fixed time interval. Therefore, the sampling method, which generates GPS points at more regular intervals, is effective for accurately maintaining the stay and velocity information of the trajectories in images, and results in accuracy improvement.

Third, as we mentioned in Sect. 5.4, we confirmed the discretization into pixels improved the robustness to spatial noises in GPS trajectories.

We concluded that our image generation method can extract important information of GPS trajectories and convert them into images effectively.

5.6 Feature visualization

We analyzed deep features by visualizing activity states of neurons on the learned DNN. In Fig. 8, the two left images show visualization results on states of activated neurons of each intermediate layer of the DNN. We can see that each layer acts as filters for extracting characteristic parts of trajectories such as moving range, moving interval, and distribution. The features also become more abstract as layers become deeper. The seven right images visualize the activity states of neurons that strongly respond to the data with each transportation mode. While it is difficult to understand all meanings of them by visualization, we can distinguish between walking, bike, and bus on the basis of moving range. Interestingly, we can see that the activity state of bus includes more dark regions than that of car. This is seemingly because buses are driven on specific roads unlike cars. These results verify that activated neurons differ depending on transportation modes and that deep learning for trajectory images can extract features that effectively distinguish between transportation modes.

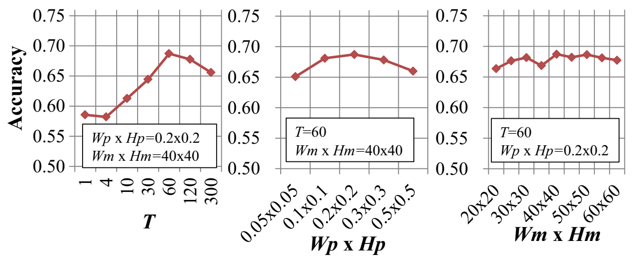


Fig. 9 Accuracy comparison depending on parameters of trajectory image generation (sampling interval T, longitude and latitude ranges $W_p \times H_p$, and images size $W_m \times H_m$)

5.7 Effect of parameters

Figure 9 compares accuracies for the GL dataset when we used different parameters of our trajectory image generation algorithm (sampling interval T, longitude and latitude ranges $W_p \times H_p$, and images size $W_m \times H_m$). When $T = 1$, the stay time was not appropriately detected and the accuracy was low because the difference in positioning intervals was not considered. In contrast, modestly increasing T enabled the difference in positioning intervals to be reduced and the accuracy to become higher. That is, our approach for sampling GPS points worked well. While too large $W_p \times H_p$ and small $W_m \times H_m$ cannot represent detailed movement in images, opposite parameter settings make trajectories sparse in images; consequently, the accuracies decrease. These results show that setting the appropriate parameters can yield better estimation performance.

Table 3 compares accuracies for the GL dataset when we used different DNN parameters (number of intermediate layers, number of neurons on each layer, and corruption rate). The best performance was obtained when three intermediate layers and 100 neurons were used. This means that too many layers and neurons can increase the expressiveness of models, but cause an over-fitting problem and vice versa. For the corruption rate, 0.4 is the best because adding moderate

Table 3 Results with the various DNN parameters

# of layers	$\#of\,neurons$	Corr. rate	Accuracy
1	100	0.2	0.633 ± 0.068
2	100	0.2	0.652 ± 0.059
3	100	0.2	0.683 ± 0.059
4	100	0.2	0.677 ± 0.058
5	100	0.2	0.680 ± 0.065
3	10	0.2	0.607 ± 0.091
3	50	0.2	0.657 ± 0.067
3	100	0.2	0.683 ± 0.059
3	200	0.2	0.671 ± 0.069
3	100	0.0	0.669 ± 0.057
3	100	0.2	0.683 ± 0.059
3	**100**	**0.4**	**0.687 ± 0.051**
3	100	0.6	0.683 ± 0.053
3	100	0.8	0.673 ± 0.052

Too many layers and neurons can increase the expressiveness of models, but cause an over-fitting problem and vice versa, and moderate corruption rate is important to improve the generalization ability of the model
Each bold font denotes the condition that yielded the highest accuracy on each experiment

Table 4 Results with and without pre-training

Algorithm	Accuracy
Fine-tuning without pre-training	0.624 ± 0.033
Fine-tuning with pre-training	**0.679 ± 0.028**

Pre-training is important to efficiently optimize model parameters in deep layers
Each bold font denotes the condition that yielded the highest accuracy on each experiment

noise to input data improves the generalization ability of the model.

Table 4 compares accuracies with and without pre-training. Without pre-training, we used randomly initialized weighting matrices and bias terms for initialization for fine-tuning. As can be seen in the table, pre-training with IDNN greatly improved accuracy without pre-training and played an important role regarding deep learning.

6 Conclusion

We have proposed a method for extracting features from raw GPS trajectories for transportation mode estimation using deep learning. Given raw GPS trajectories, our method generates trajectory images and automatically extracts features from them using a fully connected DNN with SDA. From these features and conventional handcrafted features, trajectories are classified according to transportation modes using the supervised framework.

We have demonstrated that our method outperformed the existing methods that use only the handcrafted features. This is because the DNN can appropriately extract global features from trajectory images while it is difficult for existing approaches to manually discover all features and quantify them. For example, trajectory images of walking have often narrow movements, those of bus have broad movements on regular routes, and those of train have rectilinear movements on railways. Such global features are not susceptible to GPS noise, and thus, our method performed better especially in noisy environment.

We also note that our framework can easily use deep features in conjunction with other existing features, and it can replace the classifier with other classifiers.

While we used a fully connected DNN with SDA, which is a standard method of deep learning, a convolutional neural network (CNN) is known as a closely related approach to deep learning. Although a basic CNN was proposed before deep learning emerged, a recent approach based on CNN significantly improved performance of image recognition [9]. Several learning algorithms for DNNs were also proposed, such as dropout and maxout. Nevertheless, we demonstrated that our framework for transportation mode estimation attained the highest overall performance and significant improvement in noisy environment. It is hoped that our study will become a bridge between the recently advanced approaches of deep learning and trajectory mining.

References

1. Arel, I., Rose, D.C., Karnowski, T.P.: Deep machine learning—a new frontier in artificial intelligence research. IEEE Comput. Int. Mag. **5**(4), 13–18 (2010)
2. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: NIPS, pp. 153–160 (2006)
3. Bengio, Y.: Learning deep architectures for AI. FTML **2**(1), 1–127 (2009)
4. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. TASLP **20**(1), 30–42 (2012)
5. Endo, Y., Toda, H., Nishida, K., Kawanobe, A.: Deep feature extraction from trajectories for transportation mode estimation. In: Proceedings of PAKDD2016, pp. 54–66 (2016)
6. Ermes, M., Parkka, J., Mantyjarvi, J., Korhonen, I.: Detection of daily activities and sports with wearable sensors in controlled and

uncontrolled conditions. IEEE Trans. Inf. Technol. Biomed. **12**(1), 20–26 (2006)
7. Hinton, G.E., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
8. Hung, C.-C., Peng, W.C., Lee, W.C.: Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. VLDB J. **24**(2), 169–192 (2015)
9. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS, pp. 1106–1114 (2012)
10. Le, Q.V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Ng, A.Y.: On optimization methods for deep learning. In: ICML, pp. 265–272 (2011)
11. Liao, L., Fox, D., Kautz, H.: Learning and inferring transportation routines. In: AAAI'04, pp. 348–353 (2004)
12. Parker, J.A., Kenyon, R.V., Troxel, D.: Comparison of interpolating methods for image resampling. IEEE Trans. Med. Imaging **2**(1), 31–39 (1983)
13. Parkka, J., Ermes, M., Korpippa, P., Mantyjarvi, J., Peltola, J.: Activity classification using realistic data from wearable sensors. IEEE Trans. Inf. Technol. Biomed. **10**(1), 119–128 (2006)
14. Patterson, D., Liao, L., Fox, D., Kautz, H.: Inferring high-level behavior from low-level sensors. In: UbiComp, pp. 73–89 (2003)
15. Shah, R.C., Wan, C.-Y., Lu, H., Nachman, L.: Classifying the mode of transportation on mobile phones using GIS information. In: UbiComp, pp. 225–229 (2014)
16. Shaw, B., Shea, J., Sinha, S., Hogue, A.: Learning to rank for spatiotemporal search. In: WSDM, pp. 717–726 (2013)
17. Song, X., Zhang, Q., Sekimoto, Y., Shibasaki, R.: Prediction of human emergency behavior and their mobility following large-scale disaster. In: KDD, pp. 5–14 (2014)
18. Toda, H., Yasuda, N., Matsuura, Y., Kataoka, R.: Geographic information retrieval to suit immediate surroundings. In: GIS, pp. 452–455 (2009)
19. Vedaldi, A., Fulkerson, B.: Vlfeat: an open and portable library of computer vision algorithms. In: MM, pp. 1469–1472 (2010)
20. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P. A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of ICML'08, pp. 1096–1103 (2008)
21. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. JMLR **11**, 3371–3408 (2010)
22. Zheng, Y.: Trajectory data mining: an overview. ACM TIST **6**(3), 29 (2015)
23. Zheng, Y., Liu, L., Wang, L., Xie, X.: Learning transportation mode from raw GPS data for geographic applications on the web. In: WWW, pp. 247–256 (2008)
24. Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.-Y.: Understanding Mobility Based on GPS Data. In: Ubicomp, pp. 312–321 (2008)
25. Zheng, Y., Chen, Y., Li, Q., Xie, X., Ma, W.-Y.: Understanding transportation modes based on GPS data for web applications. TWEB **4**(1), 1–36 (2010)
26. Zheng, Y., Zhou, X. (eds.): Computing with Spatial Trajectories. Springer, Berlin (2011)