Taylor & Francis
Taylor & Francis Group

Journal of Intelligent Transportation Systems
Technology, Planning, and Operations

ISSN: 1547-2450 (Print) 1547-2442 (Online) Journal homepage: https://www.tandfonline.com/loi/gits20

Deep Architecture for Citywide Travel Time Estimation Incorporating Contextual Information

Kun Tang, Shuyan Chen, Aemal J. Khattak & Yingjiu Pan

To cite this article: Kun Tang, Shuyan Chen, Aemal J. Khattak & Yingjiu Pan (2019): Deep Architecture for Citywide Travel Time Estimation Incorporating Contextual Information, Journal of Intelligent Transportation Systems, DOI: 10.1080/15472450.2019.1617141

To link to this article: https://doi.org/10.1080/15472450.2019.1617141

JOURNAL OF INTELLIGENT TRANSPORTATION SYSTEMS
https://doi.org/10.1080/15472450.2019.1617141

Deep Architecture for Citywide Travel Time Estimation Incorporating Contextual Information

Kun Tang[a,b,c], Shuyan Chen[c], Aemal J. Khattak[d], and Yingjiu Pan[c]

[a]Jiangsu Key Laboratory of Urban ITS, Southeast University, Nanjing, China; [b]Jiangsu Province Collaborative Innovation Center of Modern Urban Traffic Technologies, Southeast University, Nanjing, China; [c]School of Transportation, Southeast University, Nanjing, China; [d]Department of Civil Engineering, Nebraska Transportation Center, University of Nebraska-Lincoln, Lincoln, NE, USA

ABSTRACT

To meet the growing demand of accurate and reliable travel time information in intelligent transportation systems, this a develops a deep architecture incorporating contextual information to estimate travel time in urban road network from a citywide perspective. First, several categories of features that affect travel time significantly are analyzed and extracted. On this basis, a deep architecture, which utilizes sparse denoising auto-encoders as building blocks, is proposed to learn the feature representations for travel time estimation. To train the deep architecture successfully, a greedy layer-wise semi-unsupervised learning algorithm is devised. The proposed approach inherently incorporates both the geographical features and contextual features, and accounts for the spatial correlation of adjacent road segments. It is a deep architecture with powerful modeling capabilities for the complex nonlinear phenomena in transportation. The information contained in the huge amount of unlabeled data are fully extracted and utilized. The feature representations for estimation are adaptively learned layer by layer from the input with an unsupervised fashion. The proposed model is applied to the real case study of the road network in Beijing, China, based on the large-scale GPS trajectories collected from a sample of taxicabs. Empirical results on extensive experiments demonstrate that the novel deep architecture provides a promising and robust approach for citywide travel time estimation, and outperforms the competing methods.

ARTICLE HISTORY
Received 27 September 2017
Revised 6 May 2019
Accepted 7 May 2019

KEYWORDS
Contextual features; data-driven; deep learning; sparse denoising auto-encoder; urban road network

1. Introduction

Accurate and reliable network-wide travel time information in urban context is of significant importance at many levels of transportation planning and management (Ma, Koutsopoulos, Ferreira, & Mesbah, 2017; Tang, Chen, & Liu, 2018; Zheng, Li, van Zuylen, Liu, & Yang, 2018). It is potential to help individuals make trip related decisions more accurately, reveal congestion-prone locations in road network, and enable transportation agencies to improve transport efficiency (Jenelius & Koutsopoulos, 2013; Bharti, Sekhar, & Chandra, 2018). Urban travel time estimation aims to provide travel time information in urban context. It is a fundamental component for the successful deployment of the Intelligent Transportation Systems (ITSs), especially the Advanced Traveler Information Systems (ATISs). Travel time in urban context is impacted by various endogenous and exogenous factors, like the stochastic arrivals and departures at signalized intersections, unexpected traffic incidents and weather conditions, etc. These impact factors are inherent uncertain (Tang et al., 2018). As a result, travel time estimation in urban road network is a challenging subject (Zheng & van Zuylen, 2013).

The deployment of ITSs is bringing a once data-starved transportation field into the era of data-ubiquitous and the development of a plethora of data-driven methods for investigating transportation phenomena (Chen, Liang, & Chu, 2018; Tang et al., 2018). Amongst them, deep learning models, which automatically discovers the latent knowledge in datasets using a general-purpose learning paradigm, has gained academic and industrial attention (Siripanpornchana, Panichpapiboon, & Chaovalit, 2017; Duan, Lv, & Wang, 2016). Compared to shallow learning architectures, deep networks are capable of approximating more complex nonlinear functions through learning layers of feature representation. Thus it has the ability to deal with large variations and inherent uncertainties of travel time. Deep learning

CONTACT Shuyan Chen panyingjiu@seu.edu.cn School of Transportation, Southeast University, Sipailou 2, Nanjing, Jiangsu, China.
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/gits.

has received success in many research domains, especially in computer vision and natural language processing (Ma, Yu, Wang, & Wang, 2015). In recent years, it has also been proved to be a promising approach for transportation research, such as traffic flow forecasting (Tang, Chen, & Khattak, 2018b; Yu, Wu, Wang, Wang, &Ma, 2017), traffic speed prediction (Cui, Ke, & Wang, 2018) and traffic condition learning (Cui, Henrickson, Ke, & Wang, 2018; Ke, Li, Cui, & Wang, 2018).

Aiming at providing traffic information to improve the transportation efficiency in road network, this article develops a deep architecture, i.e., *stacked sparse denoising auto-encoders* (SSDAE), to estimate travel time on road segments in urban road network from a network-level perspective. The proposed approach inherently incorporates both the geographical features and contextual features, and accounts for the spatial correlation of adjacent road segments. It is a deep architecture, which possesses the powerful capability to model complex nonlinear phenomenon in transportation. Since the information contained in the huge amount of unlabeled data are fully extracted and utilized and the feature representations for estimation are adaptively learned layer by layer, the novel model is expected to be a promising approach for travel time estimation in urban context. The salient contributions of this research are as follows:

- Contextual information incorporation. We incorporate contextual features around road segments with geographical and spatial correlation, both of which have an impact on travel time, to estimate travel time collaboratively;
- Stacked sparse denoising auto-encoder architecture. We develop a semi-supervised deep architecture, which utilizes sparse denoising auto-encoders as building blocks, to learn layers of travel time feature representations for estimation;
- Layer-wise training algorithm. We devise a greedy layer-wise semi-supervised learning algorithm, which is comprised of three phases: unsupervised feature learning, supervised travel time estimation and global fine-tuning, to train the developed deep architecture;
- Evaluation. We evaluate the proposed deep architecture with a case study in the citywide road network of Beijing, China. Empirical results on extensive experiments demonstrate the superiorities of the proposed approach.

The remainder of the article is organized as follows. Section 2 presents a review of literature on urban travel time estimation. Section 3 describes the proposed deep architecture along with the description of the training algorithm. Section 4 comprises of the case study experimental results on the citywide road network in Beijing, China. The last section provides the research conclusions and a discussion of future research potentials.

2. Literature review

To meet the growing need of traffic information in ITSs, a number of well-designed approaches have been developed to assist in travel time analysis (Delhome, Billot, & El Faouzi, 2017; Sanaullah, Quddus, & Enoch, 2016). They involve various techniques of different disciplines. According to the taxonomy proposed in (Van Lint & Van Hinsbergen, 2012), the existing models in published literatures can be broadly classified into four categories based on the type of the method utilized, namely Naïve, parametric, nonparametric, and hybrid.

- Naïve approaches such as historical average, refer to the models that rely only on the used data or exact physical relationships (e.g., distance = speed × time). These approaches are totally independent of model assumptions and parameters. Although the accuracy is generally low (Habtemichael & Cetin, 2016), they are widely used in practice due to the easy implementation and low computational effort. Naïve approaches are usually used as a baseline approach against which other methods are compared. For example, using the historical average as a baseline method, Nikovski et al. presented an experimental comparison of several methods for travel time estimation (Nikovski, Nishiuma, Goto, & Kumazawa, 2005).
- Parametric approaches (Ma et al., 2017; Zhan, Hasan, Ukkusuri, & Kamga, 2013) such as analytical models and macroscopic models, formulate travel time using mathematical or statistical equations. Since these approaches are described with a set of model parameters, they are vulnerable to suffer from the incompatible problem of model assumptions. When traffic conditions fluctuate remarkably or road settings are complex, these approaches were proven to perform relatively poorly. For example, based on the low frequency trajectories collected from global position system (GPS) probe vehicles, Jenelius and Koutsopoulos proposed a statistical model consisting of network model and observation model to estimate travel

times for urban road network (Jenelius & Koutsopoulos, 2013). Westgate et al. developed a parametric model to estimate the distribution of travel time for ambulances between any locations in road network, using the historical trips data that may be sparse in time and coverage (Westgate, Woodard, Matteson, & Henderson, 2016). The model depends on the path traveled and explanatory variables such as the time of day and day of week. Model parameters were estimated by a Bayesian formulation and Markov chain Monte Carlo method.

- Nonparametric approaches (Fan, Su, Nien, Tsai, & Cheng, 2017; Qiao, Haghani, Shao, & Liu, 2016) such as artificial neural network (ANN)-based methods and data-analysis techniques, provide travel time information in road network based on the algorithms of machine learning and data mining. These approaches are mostly data-driven. They discover knowledge inside the data by the data itself. As they are free of model assumptions and the uncertainty involved in model parameter estimation, such approaches are generally more effective in many transportation applications. For example, Zheng and Zuylen developed a three-layer neural network model to estimate the complete link travel time, based on the sparse GPS data generated by probe vehicles (Zheng & Van Zuylen, 2013). Rahmani proposed a nonparametric method to estimate the travel time distribution of route, using low-frequency floating car data (FCD) (Rahmani, Jenelius, & Koutsopoulos, 2015). To address the bias problems of FCD such as incomplete, a number of procedures of reducing impact were involved in this model. By distributed implementation, the model is computationally efficient, scalable and can be employed for real-time applications.
- Hybrid approaches (Allstrion et al., 2016; Zhan et al., 2016) such as multi-model fusion and ensemble learning, refer to the models that fuze the aforementioned approaches at levels of model structures or decisions. On account of incorporating the advantages of multiple models, hybrid methods are generally more effective than the approaches that learn a model alone. For example, by combing a well-established theory of traffic flow through signalized intersections and a machine learning framework, Hofleitner et al. proposed a hybrid modeling framework to estimate and predict travel time using streaming GPS probe data (Hofleitner, Herring, & Bayen, 2012). Kumar et al. presented a hybrid model to estimate the travel time for bus (Kumar, Vanajakshi, & Subramanian, 2017). A data mining method k-Nearest Neighbor (k-NN) was employed to identify the significant inputs. The identified inputs were then used by a hybrid model, which combined the exponential smoothing technique and Kalman Filtering (KF) approach. Based on the latest measured data, the model parameters were dynamically estimated and updated.

While a variety of theories and models have been developed for travel time estimation in urban context over the decades, the existing approaches possess the following limitations. First, shallow architectures are extensively employed (Lv, Duan, Kang, Li, & Wang, 2015). Taking the ANN-based models as an example, due to a lack of successful training scheme for multiple hidden layers, they are conventionally designed with only one hidden layer. It limits the descriptive power of these models greatly. Second, to achieve good performance, hand-engineered features are usually essential for most approaches. It requires prior knowledge of specific domains for feature extraction and selection. These artificial features are often tedious and error-prone. Third, the existing approaches use only labeled data and have no ability to take full advantage of unlabeled data. It leads to poor model performance, especially in cases when the amount of labeled data is small. And finally, most approaches, especially descriptive model, focus only on a few of selected features that have great impact on travel time. In fact, travel time is also affected by various contextual features around the road, such as the distribution of schools and transportations. Taking these impact factors into consideration has the potential to help improve travel time estimation.

Most recently, Ke et al. models the traffic dynamics on different lanes as different channels of a spatial-temporal image, and presented a convolutional neural network (CNN) based deep learning framework to learn multilane traffic flow parameters (Ke et al., 2018). Cui et al. proposed a deep stacked bidirectional and unidirectional long short-term memory (LSTM) model to predict network-wide traffic speed (Cui, Ke et al., 2018). This is the first time that bidirectional LSTM has been exploited to measure the backward dependency of traffic data. On this basis, Cui et al. developed a traffic graph convolutional LSTM (TGC-LSTM) deep learning model for network-wide traffic state prediction (Cui, Henrickson et al., 2018). It defines the traffic graph convolution based on the

Table 1. Geographical features.

Features	Specification
Length	The length of road segment, measured by meters.
Lane	The number of Lanes on road segment, integers from 1 to 8.
Speed	The maximum speed allowed for road segment (km/h).
Direction	The direction of road segment, 0 means one-way, and 1 means bidirectional.
Level	The level of road segment, e.g., high level road like freeway and low level road like branch road, etc., integers from 0 to 9.
Tortuosity	The tortuosity of road segment, i.e., the property of being twisted. It is defined as $r_{\text{target}}.length / d$, where d is the straight-line distance between the two terminals of road segment, as shown in Figure 1, real value from 1 to infinity.
Connection	The number of neighbors (e.g., the target road segment r_{target} has 6 neighbors, as shown in Figure 2) connected to the two terminals of the road segment, integers from 0 to infinity.

physical network topology and learns the traffic network as a graph. Through the adoption of the graph convolution and LSTM layers, both the temporal and spatial correlations can be effectively captured. These studies provide great inspiration for modeling the spatial–temporal correlation of traffic data by deep learning models. However, they either need to explicitly model the traffic graph based on the whole actual road network or rely on a huge amount of historical data for model learning. These requirements may be difficult to meet in practical applications, especially in large-scale urban road networks. Besides, they rarely take into account the characteristics of the road itself and the contextual features around the road so that the impact of these features on travel time may not be captured.

To address these challenges, this article develops a deep architecture to estimate travel time in urban context from a network-level perspective. The proposed approach incorporates both the geographical features and contextual features, as well as the spatial correlation features. It is a deep architecture with powerful descriptive capability, and adaptively learns the feature representations for estimation from input data layer by layer. By means of the unsupervised feature learning, the information contained in the unlabeled data, which is generally easier to obtain than labeled data, can be fully extracted and utilized. As a result, the proposed approach has the potential to perform better than shallow networks, especially when the amount of data with label is small.

3. Methodology

In this section, the factors that impact travel time significantly are analyzed and extracted. Then, the problem of travel time estimation is formulated using a data-driven machine learning paradigm. Finally, the proposed deep architecture is elaborated, and its training algorithm is presented.

3.1. Features extraction

3.1.1. Geographical features

Travel time on road segment is highly correlated with the basic road characteristics of the segment, such as the length of the road segment. Several geographical features of road segment are extracted from the road network, as summarized in Table 1. Geographical features capture the similarity between different road segments in geographic spaces, and theoretically, road segments with similar context should have similar travel time.

3.1.2. Contextual features

In addition to the geographical features, travel time is also greatly affected by the contextual features around the road segment, such as the schools on the roadside. To this end, a variety of contextual features around road segment are also extracted, as summarized in Table 2. It is mainly comprised of the *points of interests* (POIs) at both terminals of the road segment, as shown in. There may be some other features such as traffic condition and signal timings, have an impact on travel time. These features are considered optionally in this article since on one hand, the data regarding these factors are not always available on urban road networks, especially when modeling travel time from a citywide perspective. On the other hand, we are trying to develop a model as generic as possible, which is capable of estimating travel time accurately, using only the information extracted from the easily accessible data (Zheng & van Zuylen, 2013).

3.1.3. Neighborhood spatial correlation

Taking the nature of road network into account, some spatial correlation features of the road segment are

Table 2. Contextual features.

Features	Specification
School	The number of schools.
Workplace	The number of companies, offices, and other work places.
Finance	The number of banks, ATMs, and other financial institutions.
Business	The number of malls, markets, and other shopping places.
Catering	The number of restaurants, and other food shops.
Service	The number of fueling, gas, and other vehicle service stations
Scenery	The number of scenic spots and parks.
Residence	The number of hotels, inns, and other residence places.
Transportation	The number of bus, metro stations, and other transportations.
Living	The number of KTVs, gymnasiums and other entertainments, and living services.
Sum	Amount of all considered POIs around the road segment.

also extracted to model travel time from a network perspective. The general idea is that the travel time on a road segment is not only determined by the characteristics of the road segment itself, but also affected by the features of its adjacent segments. Specifically, given a target road segment r_{target}, we extract the geographical and contextual features of both itself and some of its upstream segment $r^i_{\text{upstream}}$and downstream segments $r^i_{\text{downstream}}$. All of these features are then incorporated as the input of the model to estimate the travel time collaboratively. An illustration of 3-neighborhoods spatial correlation is illustrated in Figure 2. Note that, we consider only the upstream and downstream road segments that are straight to the target road segment, since they have relatively greater impact on travel time compared with other adjacent segments.

3.2. Machine learning model for travel time estimation

On the basis of the extracted features, a road segment can be described using a feature vector, which is comprised of the geographical, contextual and spatial correlation features. Specifically, given a target road segment i, the feature vector on the road segment is given by:

$$\mathbf{x}(i) = \left[x_1(i), x_2(i), ..., x_d(i)\right] = \left[\mathbf{g}(i-m), \mathbf{c}(i-m), ..., \mathbf{g}(i), \mathbf{c}(i), ..., \mathbf{g}(i+m), \mathbf{c}(i+m)\right] \quad (1)$$

where $\mathbf{g}(i)$ is the geographical features as introduced in Table 1. It is defined as:

$$\mathbf{g}(i) = \left[g_1(i), g_2(i), ..., g_7(i)\right] \quad (2)$$

$\mathbf{c}(i)$ is the contextual features as introduced in Table 2, and therefore defined as:

$$\mathbf{c}(i) = \left[c_1(i), c_2(i), ..., c_{11}(i)\right] \quad (3)$$

m is the number of considered upstream (or downstream) segments, $(i-m)$ and $(i+m)$ denote the m-th upstream and downstream segments, respectively. Therefore, the dimension of the whole feature vector is:

$$d = (2m+1) \times (7+11) = 36 \times m + 18 \quad (4)$$

In this fashion, given a set of road segments with travel time labels, the objective of travel time estimation is to learn a model that fits travel time and features best. Based on the learned model $\mathcal{F}$, the travel time of a new road segment can be estimated as:

$$t = \mathcal{F}(\mathbf{x}(i)) \quad (5)$$

There are a number of methods to solve this problem, such as the back propagation neural network (BPNN). However, most of them are supervised learning approach. These approaches has only the ability to make use of the data with label. On one hand, the labeled data, i.e., travel time observations, on road segments are extremely difficult and expensive to obtain, especially when modeling from a citywide perspective. On the other hand, the useful information contained in the data without label, which is easier to collected, cannot be extracted and utilized. As a result, the estimation performance of these models is limited, especially in the case of lacking labeled data. To this end, we propose in the next section a *stacked sparse denoising auto-encoders* to take full advantage of unlabeled data, leading to a powerful deep estimation architecture.

3.3. Deep travel time estimation architecture

3.3.1. Sparse denoising auto-encoder

To get better performance for a given learning model, one of the most reliable way is to train the model with more data. This has led to the aphorism in field of machine learning, "sometimes it's not who has the best algorithm that wins, it's who has the most data". Researchers have spared no effort to get more labeled data, but it is an extremely difficult and expensive task. If there exists a model being able to exploit unlabeled data that can be easily obtained, we can then train it with more data and achieve an improved

model performance. Auto-encoder (Vincent, Larochelle, Bengio, &Manzagol, 2008) is exactly such an algorithm that learns features from unlabeled data by self-taught learning, of which an illustration is shown in Figure 3. It is architecturally similar to the feedforward neural network and attempts to reconstruct the input by applying a backpropagation algorithm.

Formally, supposing we have a set of unlabeled training samples $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^d$, $i = 1, 2, ..., N$, N is the number of samples, the auto-encoder first project the input vector $\mathbf{x}$ into a feature space, and learns a representation $\mathbf{y}$ at the hidden layer by a nonlinear mapping function f_μ, as follows:

$$\mathbf{y} = f_\mu(\mathbf{x}) = s(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \tag{6}$$

where $s(\bullet)$ is a deterministic activation function, for which the sigmoid function $s(x) = \frac{1}{1+\exp(-x)}$ is usually used. And $\mu = \{\mathbf{W}_1, \mathbf{b}_1\}$ is the set of parameters of the encoding function, $\mathbf{W}_1$ is the weight matrix, $\mathbf{b}_1$ is the bias vector. Afterwards, the hidden representation can be decoded back to the original space based on the similar transformation, as follows:

$$\hat{\mathbf{x}} = g_\nu(\mathbf{y}) = s(\mathbf{W}_2\mathbf{y} + \mathbf{b}_2) \tag{7}$$

where $\nu = \{\mathbf{W}_2, \mathbf{b}_2\}$is the set of parameters of the decoding function, g_ν is a nonlinear mapping function similar to f_μ. Obviously, $\hat{\mathbf{x}}$ is an approximate reconstruction of input $\mathbf{x}$, and the objective of auto-encoder is to minimize the reconstruction error $\varepsilon(\mathbf{x}, \hat{\mathbf{x}})$ between $\mathbf{x}$ and $\hat{\mathbf{x}}$, as follows:

$$\underset{\mu,\nu}{\operatorname{argmin}} \left\{ \varepsilon(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}\|^2 + \lambda\left(\|\mu\|^2 + \|\nu\|^2\right) \right\} \tag{8}$$

The first term is the empirical loss between the inputs and their reconstructions, where $\| \bullet \|^2$ denotes the ℓ_2-norm. The second term is a regularization term that penalizes the magnitude of weights and biases, and helps prevent over-fitting input data, where λ controls the weight of this term.

One issue with classical auto-encoder is learning the identity function when the neurons number in the encoder layer is equal to or larger than the input layer (Lv et al., 2015). For this end, sparsity constraints are usually imposed on the activation neurons in the encoder layer, which leads to a sparse auto-encoder, i.e., SAE.SAE (Vincent, Larochelle, Bengio, & Manzagol, 2010) learns a sparse representation for the input by minimizing the reconstruction error with an extra sparsity penalty term, as follows,

$$\underset{\mu,\nu}{\operatorname{argmin}} \left\{ \varepsilon_{sparse}(\mathbf{x}, \hat{\mathbf{x}}) = \varepsilon(\mathbf{x}, \hat{\mathbf{x}}) + \beta\sum_{j=1}^{H} KL\left(\rho|\hat{\rho}_j\right) \right\} \tag{9}$$

The first term $\varepsilon(\mathbf{x}, \hat{\mathbf{x}})$ is the auto-encoder defined in Eq. (8). The second term is a regularization term controlling the sparsity, where β is the sparsity penalty weight, H the encoder neurons number, ρ the constant sparsity parameter, $\hat{\rho}_j = (1/N)\sum_{i=1}^{N} y_j^{(i)}$ the average activation for the j-th encoder neuron, and $KL(\bullet)$ the Kullback–Leibler (KL) divergence defined as

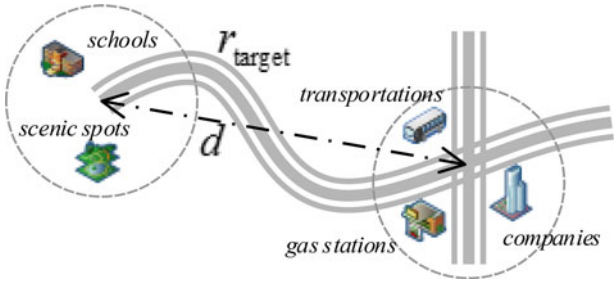

Figure 1. Features of a typical road segment.

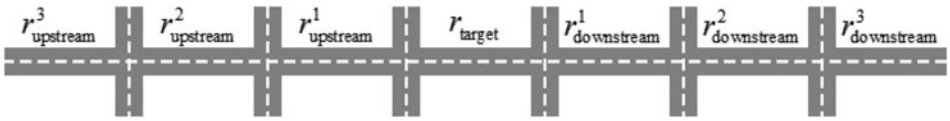

Figure 2. Illustration of neighborhood spatial correlation.

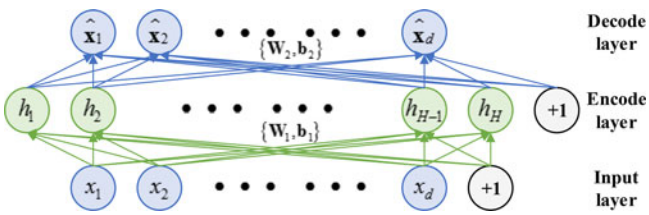

Figure 3. Illustration for auto-encoder.

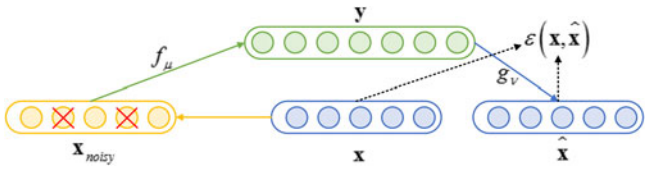

Figure 4. Architecture for denoising auto-encoder.

$$KL(\rho|\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{10}$$

In many cases, due to the unavoidable interference factors, such as transmission error, the input data is usually noisy or partially destroyed. To learn robust features from these noisy input, a sparse denoising auto-encoder, i.e., SDAE, is proposed in (Vincent et al., 2010). SDAE is an extension of SAE and attempts to reconstruct from the noisy inputs $\mathbf{x}_{noisy}$, as follows:

$$\mathbf{y} = f_\mu(\mathbf{x}_{noisy}) = s(\mathbf{W}_1\mathbf{x}_{noisy} + \mathbf{b}_1) \tag{11}$$

$$\hat{\mathbf{x}} = g_\nu(\mathbf{y}) = s(\mathbf{W}_2\mathbf{y} + \mathbf{b}_2) \tag{12}$$

where the noisy input $\mathbf{x}_{noisy}$is obtained by corrupting the initial input $\mathbf{x}$ by means of a stochastic mapping $\mathbf{x}_{noisy} = q_D(\mathbf{x}_{noisy}|\mathbf{x})$. Specifically, for each input $\mathbf{x}$, a number of randomly selected components are forced to 0, while the others are left untouched (Vincent et al., 2010). Even though $\hat{\mathbf{x}}$ is reconstructed from the $\mathbf{x}_{noisy}$ instead of $\mathbf{x}$, SDAE still minimizes the reconstruction error $\varepsilon_{sparse}(\mathbf{x}, \hat{\mathbf{x}})$ between $\mathbf{x}$ and $\hat{\mathbf{x}}$. The architecture of SDAE is illustrated in Figure 4, where neurons with red cross in $\mathbf{x}_{noisy}$ denote stochastically corrupted inputs.

3.3.2. Deep learning architecture

To discover patterns and knowledge from massive unlabeled data, an effective way is to construct a hierarchical architecture for deep learning. The primary advantage of deep architecture is that it can represent a significantly larger set of functions in a compact manner compared with shallow networks. A stacked sparse denoising auto-encoder, i.e., SSDAE, is a deep architecture stacked by multiple layers of sparse denoising auto-encoders hierarchically, in which the output from the previous layer is fed as input to the next layer. It extracts features from input data layer by layer. The low layer learns the minutiae, with the increase of the number of layers, we can get more and more abstract features, which can then be used for subsequent classification and regression tasks.

On the basis of the high level features learned by SSDAE, a predictor layer is added on the last layer of SSDAE to estimate travel time. The predictor layer is a supervised learning scheme and has a variety of choices, for example, logistic regression. In this fashion, the deep architecture for travel time estimation employed in this article is comprised of an SSDAE and a predictor layer, of which the whole structure is illustrated in Figure 5.

3.3.3. Training algorithm

Similar to neural network, deep architecture is traditionally trained by applying gradient-based backpropagation (BP) algorithm. It is difficult to achieve fine performance by training the deep architecture using BP solely. An alternative way to obtain good parameters for a deep architecture is to use the greedy layer-wise training, which works by training each layer of the deep architecture in turn. It is a semi-supervised learning scheme that is capable of training deep network successfully, as Hilton and Bengio presented in (Hinton, Osindero, & Teh, 2006) and (Bengio, Lamblin, Popovici, & Larochelle, 2007). Following the idea of greedy layer-wise pretraining, the training procedures of the proposed deep architecture is summarized in Figure 6. It is comprised of three major phases, as follows:

- Phase 1: layer-wise unsupervised learning in a bottom-top fashion. Learning the parameters of each layer in SSDAE individually while freezing parameters for the remainder of the network, using an unsupervised learning algorithm. The input is the all attribute data $\mathbf{X}_{unlabel}$ and $\mathbf{X}_{label}$, and the output is the pretrained weight matrices and bias vectors of the encoders $\{\mathbf{W}_1^{(l)}, \mathbf{b}_1^{(l)}\}$, for all $l = 1, 2, \ldots, L$.
- Phase 2: supervised learning for the last layer. Learning the parameters of the last layer, i.e., the predictor, to estimate travel time using a supervised learning algorithm. The input is the attribute data with label $\mathbf{X}_{label}$, the label data $\mathbf{T}$, and the pretrained SSDAE (i.e., $\{\mathbf{W}_1^{(l)}, \mathbf{b}_1^{(l)}\}$, for all $l = 1, 2, \ldots, L$) in Phase 1. The output is the pretrained weight matrix and bias vector of the predictor $\{\mathbf{W}_p, \mathbf{b}_p\}$.
- Phase 3: global fine-tuning in a top-bottom fashion. Fine-tuning the parameters of the whole deep network to improve performance by applying

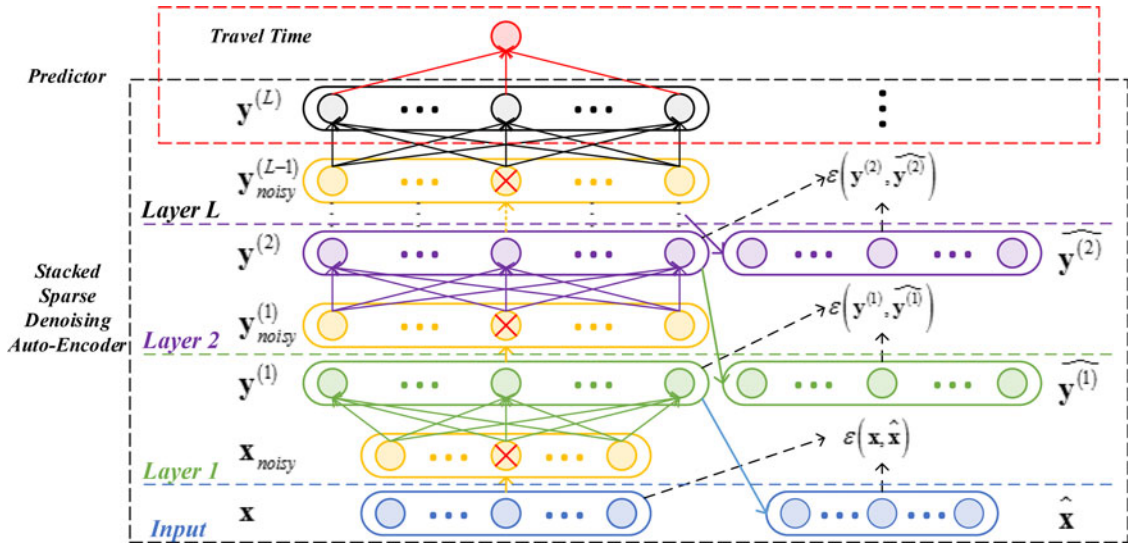

Figure 5. Deep learning architecture for travel time estimation.

backpropagation algorithm. The input is the attribute data with label $\mathbf{X}_{label}$, the label data $\mathbf{T}$, the pretrained $\{\mathbf{W}_1^{(l)}, \mathbf{b}_1^{(l)}\}$, for all $l = 1, 2, \ldots, L$) and the pretrained $\{\mathbf{W}_p, \mathbf{b}_p\}$. The output is optimized parameters of the whole deep network $\{\mathbf{W}_1^{(l)}, \mathbf{b}_1^{(l)}\}$ and $\{\mathbf{W}_p, \mathbf{b}_p\}$, for all $l = 1, 2, \ldots, L$.

4. Case study

4.1. Data

The proposed deep travel time estimation architecture is applied to the urban road network of Beijing, China. The citywide road network is comprised of 148,110 nodes and 196,307 road segments, and covers a 40×50 km rectangle spatial area. The geographical features (see Table 1) were extracted from the digital road network. The contextual features (see Table 2) were extracted from the points of interests (POIs) dataset in Beijing, of which the 273,165 POIs were classified into 195 categories. Here, the top 11 categories that affect travel time strongly were chosen, based on the correlation analysis between features and travel time. Travel times were extracted from the large-scale GPS trajectories generated by over 32,670 taxicabs driving in Beijing from Sep. 1 to Oct. 31, 2013, based on the map-matching algorithm introduced in (Yuan, Zheng, Zhang, Xie, & Sun, 2010).Since traffic conditions in morning peak hours are of the most concern, we divide the time period 7:00–9:00 am into four 30-min time slots. The lessons learned from the experiences in references (Wang, Zheng, & Xue, 2014) and (Zheng, Li, van Zuylen, Liu, &Yang, 2015) demonstrate that time slot with 30-min is a good description of traffic condition, and achieves the best tradeoff between performance and efficiency. For each time slot, we compute the average travel time of different road segments. As the number of involved vehicles is limited and a taxicab can only travel a few road segments in a time slot, many road segments may have no travel time observations.

In practice, especially when using GPS trajectories, the experimental data is inevitably interfered by kinds of noises, e.g., equipment transmission error and map-matching error. In the implementations, we use a mixture of Gaussians to estimate the least likely data, and label it as anomalous. Specifically, we fit 3 multivariable Gaussians to the data in the logarithmic space of road length and travel time, and identify the lowest 1% likelihood data with "outliers", based on the 3-sigma rule (almost 99% of values lie within three standard deviations away from the mean) of Gaussian distribution. As shown in Figure 7, red crosses represent the most likely outliers, blue crosses denote the outliers identified by mixture of Gaussian, yellow curves are the contours of mixture Gaussian, and green plus signs are the normal data after anomaly filtering. Experiments are then conducted based on these normal data.

The results of correlation analysis between extracted features and travel time are presented in Figure 8, where each subplot corresponds to a time slot. The horizontal axis represents the number of features and the vertical axis represents the value of (Pearson) correlation coefficient between the feature and travel time. As can be seen from the figure, both the geographical features and contextual features show clear correlation with travel time. The trends of correlation between features and travel time in different

Algorithm: Training the deep learning architecture for travel time estimation

Input: Attribute data of all road segments $\mathbf{X}_{unlabel}$, attribute data with label $\mathbf{X}_{label}$, label data $\mathbf{T}$, and desired number of layers L.

Output: Encoder weight matrices and bias vectors $\left\{\mathbf{W}_1^{(l)}, \mathbf{b}_1^{(l)}\right\}$ of each layer, for all $l = 1, 2, \cdots, L$, and predictor weight matrix and bias vector $\left\{\mathbf{W}_p, \mathbf{b}_p\right\}$.

Phase 1: Unsupervised learning for SSDAE, solely based on $\mathbf{X}_{unlabel}$.

For each layer $l = 1, 2, \cdots, L$:

1) Set the training parameters for the layer, including the number of neurons in the layer, weight regularization coefficient λ, sparsity penalty coefficient ρ, sparsity regularization coefficient β, and noise ratio η;
2) For each sample in the output $\mathbf{y}^{(l-1)}$ of the layer $l-1$, add stochastic noise to construct $\mathbf{y}_{noisy}^{(l-1)}$. Note that, for the first layer, i.e., $l = 1$, $\mathbf{y}^{(0)}$ corresponds to the samples in input data $\mathbf{X}_{unlabel}$;
3) Initialize $\left\{\mathbf{W}_1^{(l)}, \mathbf{b}_1^{(l)}\right\}$ with small random values, and encode $\mathbf{y}_{noisy}^{(l-1)}$ into $\mathbf{y}^{(l)}$;
4) Initialize $\left\{\mathbf{W}_2^{(l)}, \mathbf{b}_2^{(l)}\right\}$ with small random values, and decode $\mathbf{y}^{(l)}$ into $\mathbf{y}^{(l-1)}$;
5) Find the encoding parameters $\left\{\mathbf{W}_1^{(l)}, \mathbf{b}_1^{(l)}\right\}$ by minimizing the reconstruction error $\varepsilon\left(\mathbf{y}^{(l-1)}, \mathbf{y}^{(l-1)}\right)$;

Phase 2: Supervise learning for predictor, based on $\mathbf{X}_{label}$.

6) Compute the output $\mathbf{y}^{(L)}$ of the SSDAE for the labelled training data $\mathbf{X}_{label}$, based on the SSDAE trained in Phase 1;
7) Initialize predictor weight matrix and bias vector $\left\{\mathbf{W}_p, \mathbf{b}_p\right\}$ with small random values;
8) Compute the travel time prediction $\mathbf{T}_{pre}$;
9) Find the parameters $\left\{\mathbf{W}_p, \mathbf{b}_p\right\}$ by minimizing the prediction error between the true travel times $\mathbf{T}$ and their predictions $\mathbf{T}_{pre}$;

Phase 3: Global fine-tuning the deep network, based on $\mathbf{X}_{label}$.

10) Initialize the parameters of the whole deep network with the weights and biases computed in Phase 1 for the SSDAE and Phase 2 for the predictor.
11) Compute the travel time prediction $\mathbf{T}_{pre}$ for the labelled training data using the whole deep network;
12) Fine-tune the parameters of the whole deep network at the same time using the gradient-based backpropagation algorithm, and find the optimal parameters $\left\{\mathbf{W}_1^{(l)}, \mathbf{b}_1^{(l)}\right\}$, for all $l = 1, 2, \cdots, L$, and $\left\{\mathbf{W}_p, \mathbf{b}_p\right\}$ by minimizing the prediction error between $\mathbf{T}$ and $\mathbf{T}_{pre}$.

Figure 6. Training algorithm for deep travel time estimation.

time slots are similar. In general, the correlation between geographical features (feature number 1–7) and travel time is greater than that of contextual features (feature number 8–18) in all time slots. The length of road segment (number 1) is positively correlated with travel time while the speed (number 2) of road segment is negatively correlated with travel time. It is reasonable that the travel time of a road segment

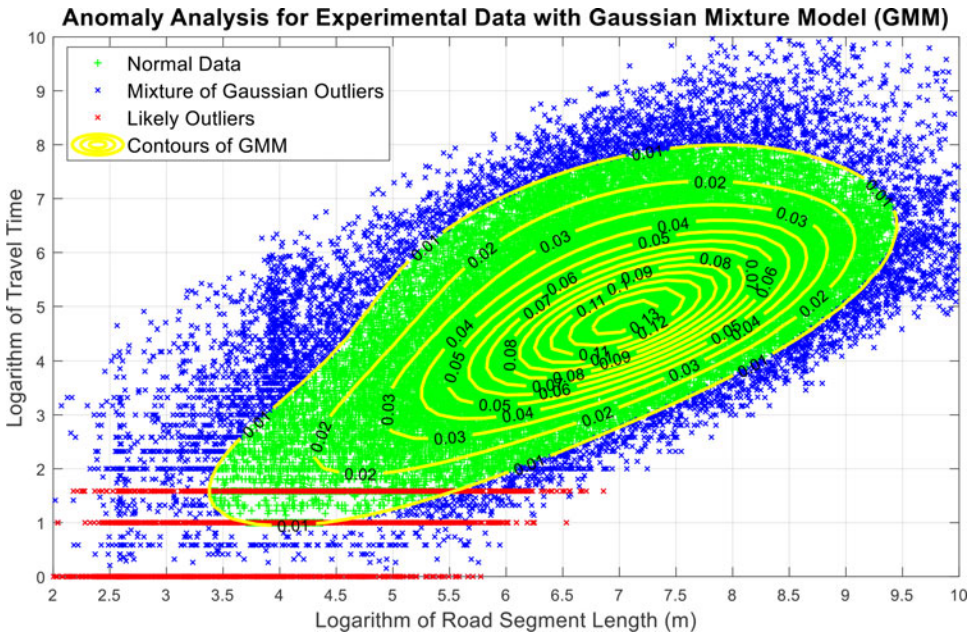

Figure 7. Experimental data distribution.

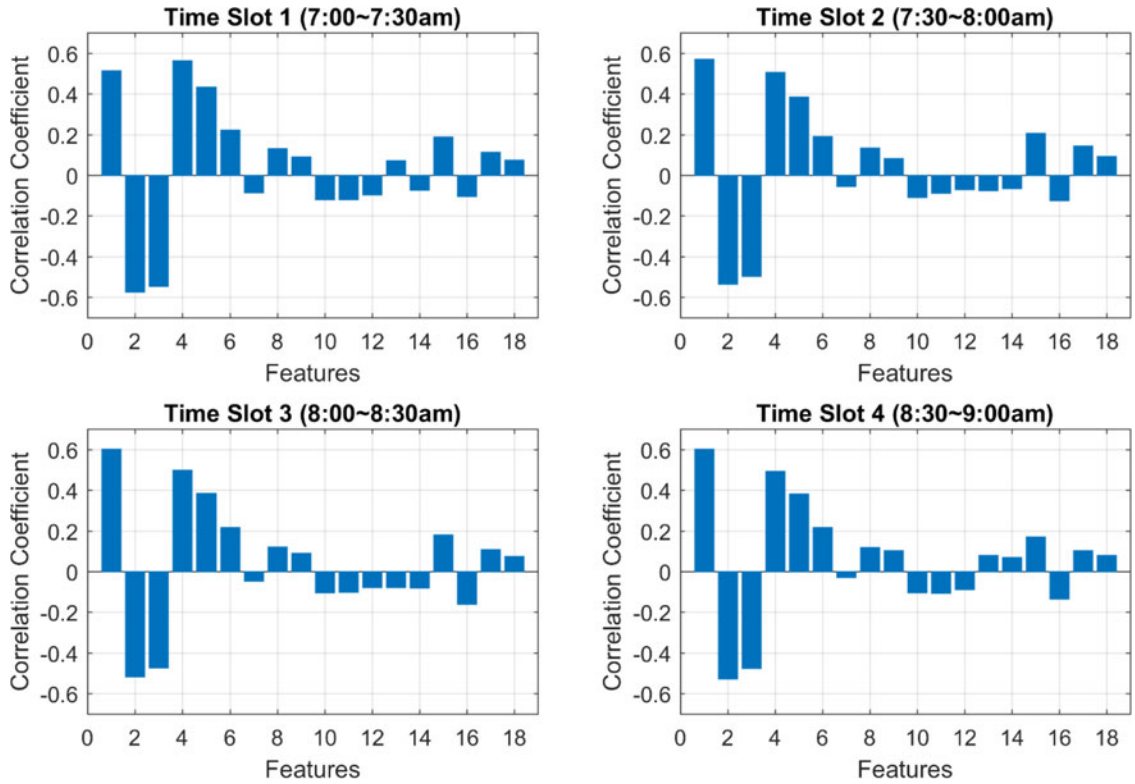

Figure 8. Correlation between features and travel time.

increases with the length while decreases with speed. Amongst the contextual features, the residences (number 15) around road segment show the largest correlation between travel times. It implies that the more the number of residences around the road segment, the larger the traffic demand. As a result, it takes a longer time to traverse the road segment.

Unsupervised learning is used to learn the feature representations in a greedy layer-wised manner, using the attribute data with and without label. Learned feature representations are then input to the predictor layer, i.e., the last layer, for travel time estimation. The predictor layer of the proposed model are trained and fine-tuned based on supervised learning using the

Table 3. Model parameters.

Model	Hidden layers	Model structure	λ	β	ρ	η
BPNN	1	[126 1000 1]				
SAE1	2	[126 100 100 1]	0.001			
SAE2	3	[126 100 100 100 1]	0.001			
SSAE	3	[126 300 300 300 1]	0.001	0.005	0.1	
SSDAE	3	[126 300 300 300 1]	0.001	0.005	0.1	0.05

Table 4. Performance comparison of different models in different time slots.

	Time slot 1			Time slot 2			Time slot 3			Time slot 4		
Model	MAE	MAPE	MSE	MAE	MAPE	MSE	MAE	MAPE	MSE	MAE	MAPE	MSE
BPNN	17.9	0.353	579	18.3	0.339	577	19.1	0.325	560	17.7	0.341	529
SAE1	15.7	0.325	561	16.2	0.323	548	15.2	0.313	528	14.3	0.320	499
SAE2	14.6	0.313	542	14.9	0.309	533	14.6	0.306	524	13.7	0.299	492
SSAE	13.9	0.302	528	14.2	0.297	520	13.7	0.293	508	12.8	0.285	489
SSDAE	13.2	0.284	527	14.0	0.289	513	13.5	0.282	499	12.6	0.278	479

data with label. Since unsupervised learning is only used for features learning and model performance can only be measured by the data with label (observed travel time), the evaluation of model performance is independent of unsupervised learning. As a result, only supervised learning is discussed in this evaluation-related section. To evaluate the performance of the proposed model, we randomly partition the road segments having travel time observation into training set, validation set and test set according to the ratio of 3:1:1. Training set is used for supervised training and validation set is used to determine whether training should be terminated. After training, travel times in the test set can be estimated based on the trained model to evaluate model performance.

4.2. Model specification

To demonstrate the advantages of the proposed model, we compare it with the following four methods.

- Back Propagation Neural Network (BPNN). The traditional artificial neural trained solely based on backpropagation algorithm;
- Stacked Auto-Encoder without unlabeled data (SAE1). The deep architecture stacked by several layer of auto-encoders, of which the hidden layers are pretrained using only the attribute data of the road segments with travel time label;
- Stacked Auto-Encoder with unlabeled data (SAE2). In contrast to SAE1, the hidden layers in SAE 2 are pretrained using the attribute data of all road segments;
- Stacked Sparse Auto-Encoder (SSAE). In contrast to SAE2, the unsupervised layers of SSAE are sparse auto-encoders, which learns a sparse representation of the input.

For simplicity, we refer the proposed stacked sparse denoising auto-encoder model as SSDAE. Among these methods, except for the BPNN, the rest are all semi-supervised deep learning approaches. Comparing with BPNN, we demonstrate the advantages of deep architecture and unsupervised pretraining in SAE1. SAE2 shows the effectiveness of utilizing unlabeled data to improve estimation accuracy. SSAE reveals the contribution of sparse representation learning. And SSDAE justifies why a denoising treatment is needed in our approach.

To train the models, we have to specify some parameters that determine model structure, such as the layers number of the deep architecture and the neurons number in each layer. For the input layer, considering the tradeoff between effectiveness and efficiency, we set m in Eq. (1) to be 3 and put the attribute data of these road segments together to estimate travel time on the target road segment. Therefore, the dimension of input and output are $36 \times 3 + 18 = 126$ and 1, respectively. The proposed model accounts for the spatial correlation of travel time inherently. For other parameters, we search the best value from their specified parameter space. Specifically, we choose the layers number from $\{1, 2, 3, 4, 5\}$, the neurons number in each layer from $\{10, 30, 100, 300, 1000\}$, the weight penalty λ, sparsity penaltyβ, the sparsity coefficient ρ and the noise coefficient η from 0 to 1 with step 0.1. Based on the grid search with the objective of minimizing the estimation error (l_2-norm), we get the optimal parameters for each model, as shown in.

4.3. Validation metrics

Three widely used validation metrics are employed to quantify estimation performance. These are *Mean Absolute Error* (MAE), *Mean Absolute Percentage Error* and *Mean Squared Error* (MSE), and defined as:

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|t_n - \hat{t_n}| \tag{13}$$

$$MAPE = 100 \times \frac{1}{N}\sum_{n=1}^{N}|\frac{t_n - \hat{t_n}}{t_n}| \tag{14}$$

$$MSE = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - \hat{t_n}\right)^2 \tag{15}$$

where t_n and $\hat{t_n}$ denote the n-th true and estimated

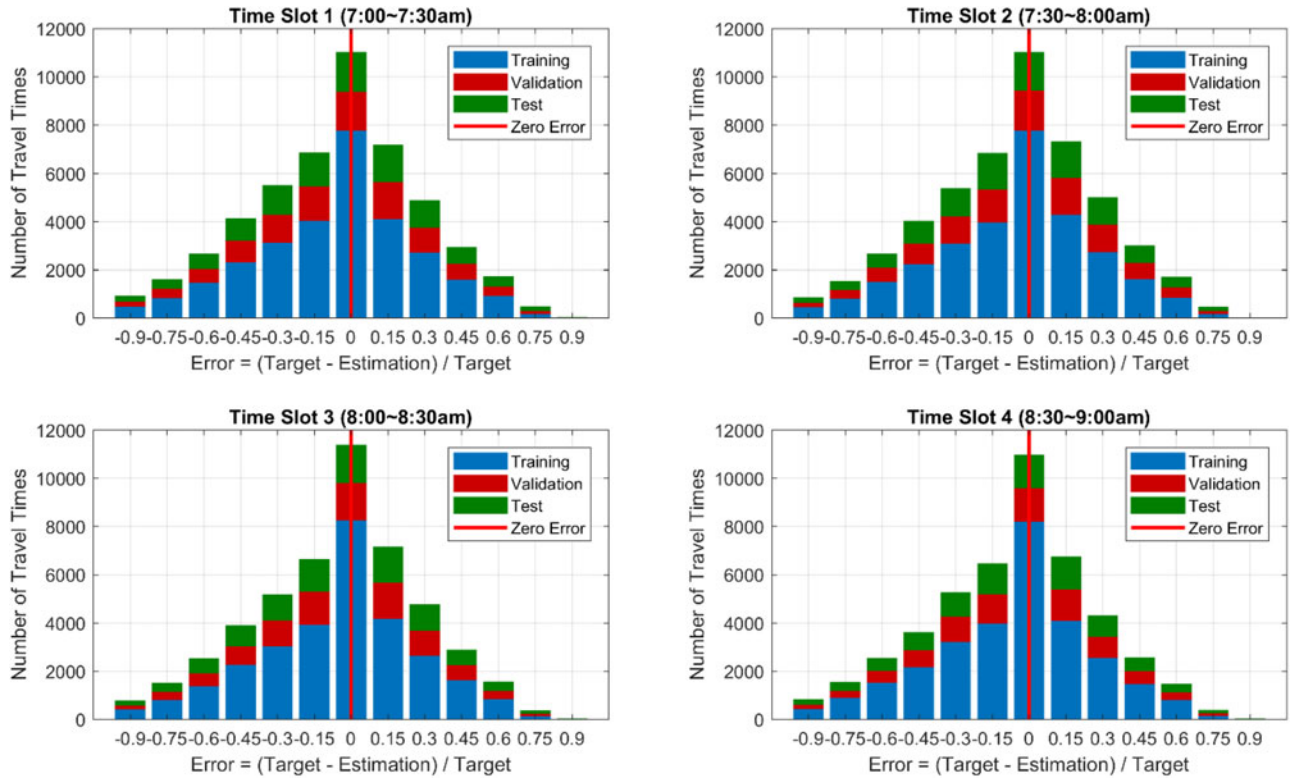

Figure 9. Estimation error distribution with histograms.

travel time, respectively, and N is the total number of estimations.

4.4. Results

4.4.1. Model results

Table 4 shows the estimation accuracies on the test dataset from different models in different time slots. As illustrated in this table, owing to the merits of deep architecture, SAE1 has a better estimation accuracy than the shallow model BPNN. Due to the contribution of making use of unlabeled data, SAE2 is more accurate than SAE1. Benefiting from the advantages of sparse feature learning, SSAE performs better than SAE2. And due to the denoising treatment, the proposed model SSDAE achieves the highest estimation accuracies amongst the competing methods. For example, compared to the MAE of the baseline method BPNN, the improvement of SSDAE is up to 4.7 s, 4.3 s, 5.6 s, and 5.1 s for time slot 1 to 4, respectively. The results clearly demonstrate the superiorities of the proposed model SSDAE.

To demonstrate the robustness of the proposed model, we provide the estimation error distribution of SSDAE in Figure 9. The results of different time slots are given in separate sub-graphs. For each sub-graph, x-axis is the estimation error in percentages of the actual travel time values, which is calculated by subtracting the estimated value from the true value and then divided by the true value (i.e., error = (target − estimation)/target). Y-axis is the number of travel time samples whose estimation error locate in corresponding error intervals. The blue, red, and green histograms represent the results of training, validation, and test data set, respectively. The red vertical line in the middle is the zero error line. It can be observed from the figure that, for all time slots, a large amount of travel times are estimated with percentage error around zero. For example, in time slot 1, the number of travel time samples with estimation error around zero is around 110,00, accounting for 22% (11000/50490) of the total samples. With the increase of the amplitude of the estimation error, the number of travel time samples with estimation error located in corresponding intervals becomes less and less. For all time slots, the number of travel times with estimation error in percentage of the actual value larger than 0.9 is nearly zero.

The proposed model is a learning-based approach, which means the parameters of the model need to be initialized. We further study the effect of initialization values on the estimation accuracy of different models. Figure 10 shows the boxplots of the MSE scores of different models and is grouped by time slot. Each boxplot with different color is associated with a model, and summarizes the MSE scores of estimation

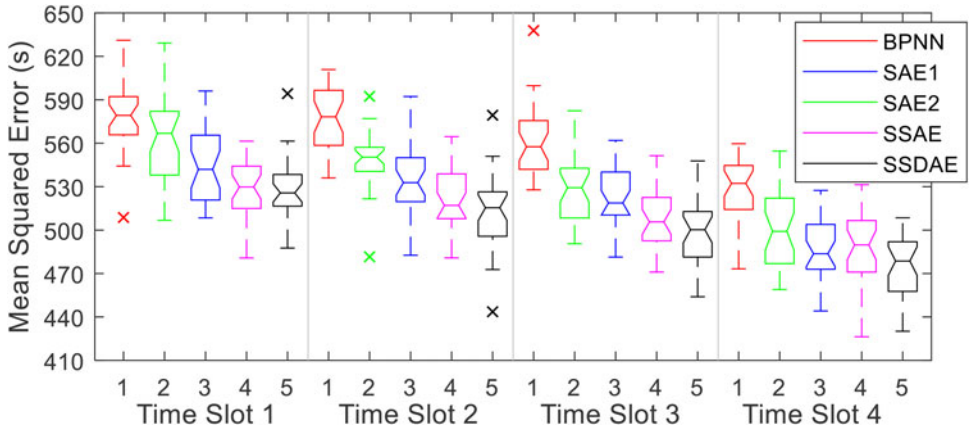

Figure 10. MSE accuracy of different models in different time slots.

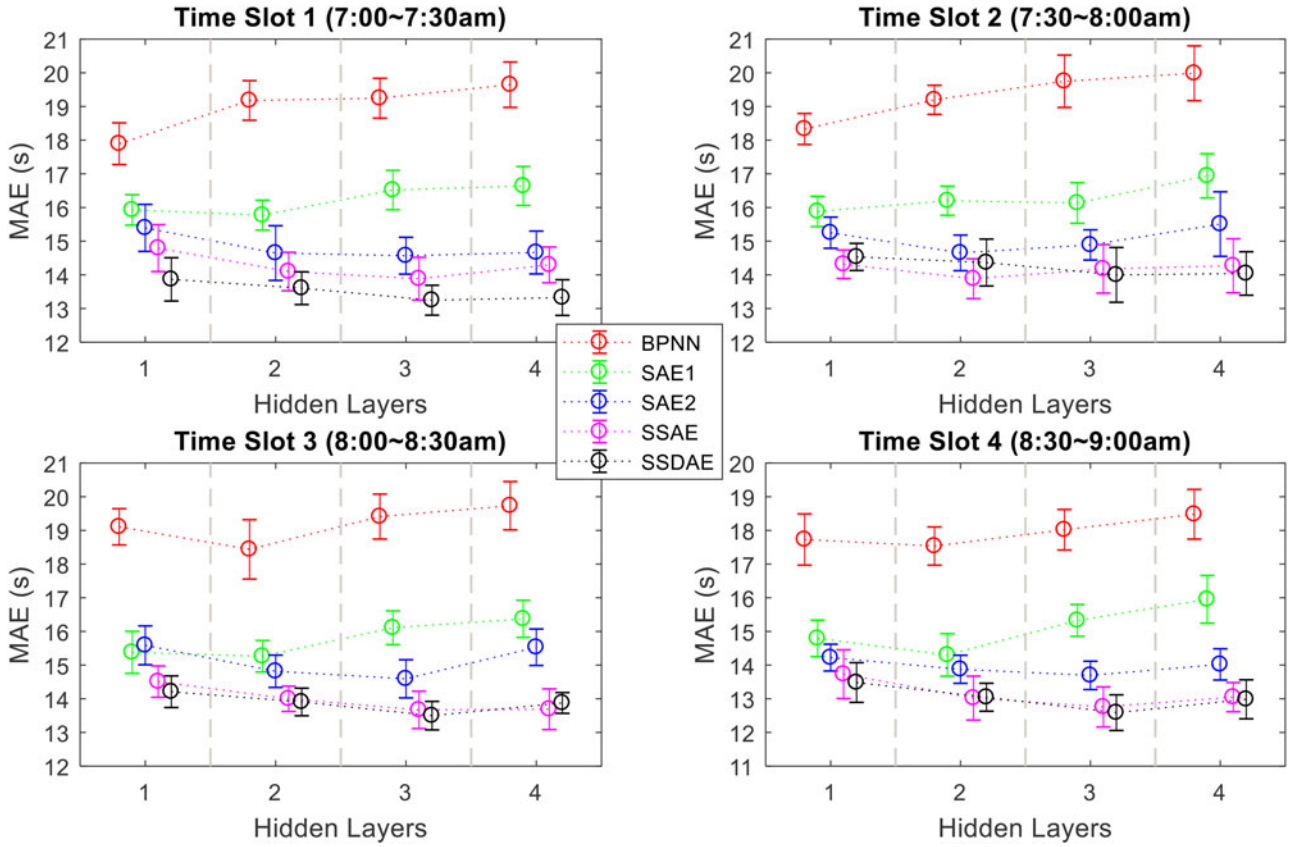

Figure 11. MAE accuracy of different models with different hidden layers.

for 30 independent experiments with different initialization values. The box plots are generated using the "boxplot" command (with "notch" on) in the statistics toolbox of MATLAB 2017a. The median value is shown as a solid line inside the box, the 25th and 75th percentile values are indicated by the top and bottom edges of the notched box extending from the median value, and the outliers are shown as small crosses. For all time slots, although the values used to initialize the model are different, the estimation accuracies in terms of MSE derived with SSDAE only fluctuate in a reasonable range. What's more, the proposed model SSDAE distinctly performs better than other competing methods for all time slots. It is consistent with the results reported in Table 4, and clearly demonstrates the robustness of the proposed model.

4.4.2. Sensitive analysis

For deep model, one important issue is determination of the number of hidden layers of the architecture. To

evaluate how this hyper-parameter influences the estimation performance, we conduct experiments and study the estimation accuracy. Figure 11 provides estimation accuracy in terms of MAE varying with the number of hidden layers. Each subplot is associated with a time slot, and plots the MAE scores of different models with different hidden layers. It plots the error bars of MAE scores for 30 independent experiments, using the command "error bar" of MATLAB. The number of hidden layers is tested from 1 to 4, and the number of neurons in each layer is chosen by cross volition, as shown in Table 3. The best values of the number of hidden layers for models BPNN (red), SAE1 (green), SAE2 (blue), SSAE (magenta), and SSDAE (black) are 1, 2, 3, 3, and 3, respectively. When the layers number becomes larger (larger than the best value), the model performance will not further improve and on the contrary, it will decrease to some extent. Lessons learned from extensive practical experiences indicate that the number of hidden layers of a deep learning architecture should be neither too large nor too small. The results confirm these lessons.

As described in Section 3.1, we use three categories of features, including the geographical, contextual and spatial correlation features, as the input for the proposed deep architecture SSDAE. One of the remaining questions is whether all the input information plays a role in travel time estimation accuracy. We thus develop several models with different input information in the following subsection to explore the impact of inputs on the estimation results. Note that, the geographical features are the basic information of the target road segment, and therefore are always used as input data.

- Model 1: Using only the geographical features (see Table 1) of the target road segment as input, and the dimensionality of the input is just 7;
- Model 2: Using the geographical and contextual features (see Table 2) of the target road segment as input, and the dimensionality of the input is $7 + 11 = 18$;
- Model 3: Using the geographical and contextual features of the target road segment and the neighborhood spatial correlation features of the first upstream road segment (see Figure 2) as input, and the dimensionality of the input is just $18 \times 2 = 36$;
- Model 4: On the basis of model 3, adding the neighborhood spatial correlation features of the first downstream road segment to the input, and the dimension of the input is $18 \times 3 = 54$;
- Model 5: On the basis of model 4, adding the neighborhood spatial correlation features of the second upstream road segment to the input, and the dimension of the input is $18 \times 4 = 72$;
- Model 6: On the basis of model 5, adding the neighborhood spatial correlation features of the second downstream road segment to the input, and the dimension of the input is $18 \times 5 = 90$;
- Model 7: On the basis of model 6, adding the neighborhood spatial correlation features of the third upstream road segment to the input, and the dimension of the input is $18 \times 6 = 108$;
- Model 8: On the basis of model 7, adding the neighborhood spatial correlation features of the third downstream road segment to the input, and the dimension of the input is $18 \times 7 = 126$. It is actually the final model employed in this article.

To analyze the sensitivity of SSDAE to the input information, we study the estimation accuracy in terms of MAPE for the models 1 to 8 with different input data. Similar to Figure 10, Figure 12 displays the boxplots of MAPE scores of different models and are grouped by time slot. Each box is associated with the results of a model, and summarizes the MAPE scores of estimation for 30 independent experiments. The error bar of the MAPE is plotted on the right side of each box, using the MATLAB command "error bar". We can observe that, due to the introduction of certain features, the model estimation accuracy is interfered and decreases to some extent. However, on the whole, estimation accuracy of the model improves with increase input information. Compared with other models, both the median and mean values derived from Model 8 (with all features input) are much lower. It implies that: (1) increasing input information is helpful for the proposed model to obtain more accurate travel time estimates, (2) the proposed deep architecture is capable of extracting useful information from the noise data to improve model performance.

5. Conclusion

This article develops a deep architecture to estimate travel time on road segment in urban road network from a citywide perspective. First, we analyze the impact factors of travel time and extract several categories of features data, including the geographical features, contextual features and neighborhood spatial correlation features. On the basis of these features, we formulate the travel time estimation problem with a data-driven machine learning paradigm. Afterwards,

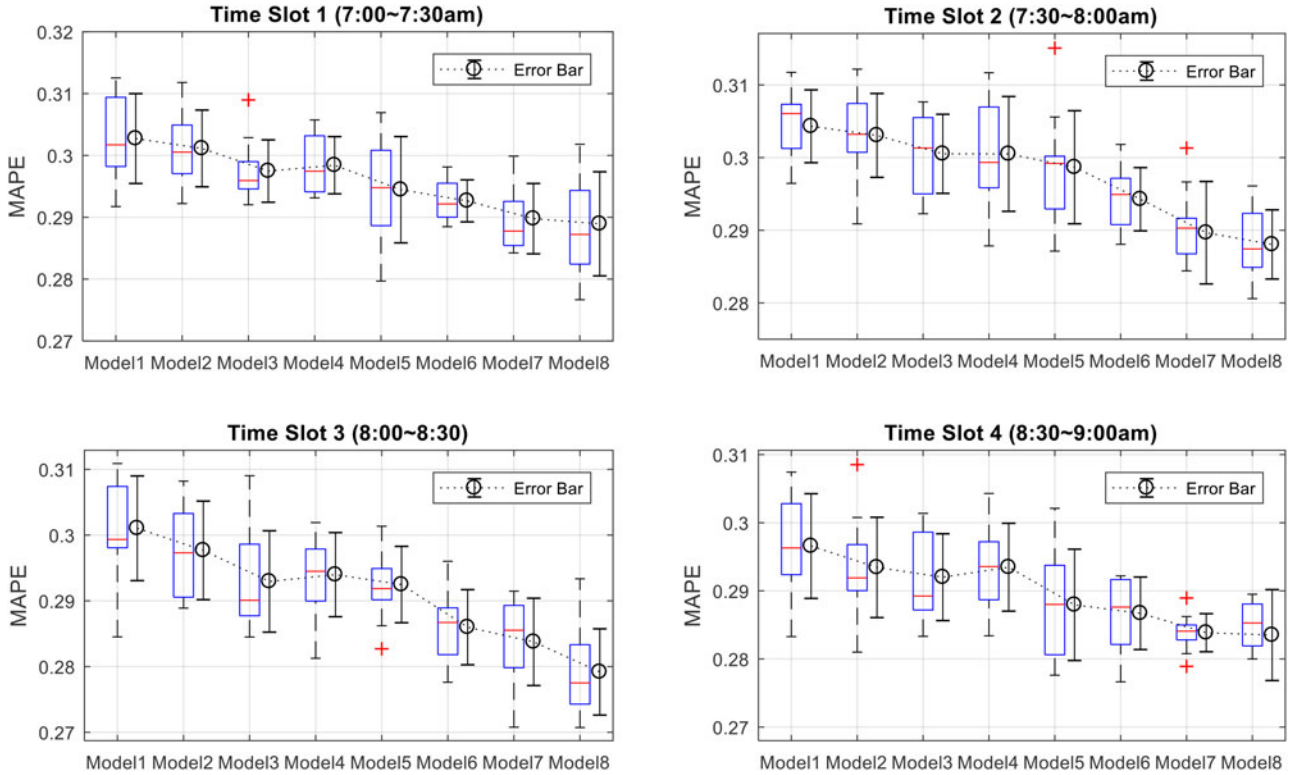

Figure 12. MAPE accuracy of SSDAE with different input information.

we construct a deep architecture using sparse denoising auto-encoders as building blocks to solve the travel time estimation problem. To train the deep architecture, a greedy layer-wise semi-unsupervised learning algorithm is devised. It makes full use of the unlabeled data to adaptively learn hierarchical feature representations from the input data for travel time estimation. The proposed approach inherently incorporates both the geographical features and contextual features, as well as the neighborhood spatial correlation features.

The proposed deep architecture is applied in a case study on the citywide road network of Beijing, China, based on the GPS trajectories collected from a sample of taxicabs. Empirical results from extensive experiments demonstrate that the proposed deep architecture provides a promising and robust approach for network-wide travel time estimation, and outperforms competing methods considered in this research.

Future research is recommended in the following two directions. First, travel times of different time slots are estimated separately in this article. However, travel times on the same road segment in continuous time slots are temporally correlated with each other. Learning multiple estimation tasks collaboratively may make further performance improvement (Huang, Song, Hong, & Xie, 2014). In addition, we train a single model for the whole training data. However, on account of the large variation and inherent uncertainties of the travel time in urban context, some travel times (outliers) could not be estimated accurately by the trained model, because their true hypothesis (distribution) may be outside the hyperspace supported by the training data set (Zhou et al., 2017). Ensemble multiple models for data sets with different hypothesis may help to further improve the estimation accuracy.

Acknowledgment

We are grateful for the comments and suggestions from the editor and the reviewers who helped improve the article.

Funding

This work was sponsored by the National Science Foundation of China (NO. 61374195), the Fundamental Research Funds for the Central Universities and the Jiangsu Province University Graduate Student Research and Innovation Program (NO. KYLX_0180).

References

Allstrion, A., Ekstrom, J., Gundlegård, D., Ringdahl, R., Rydergren, C., Bayen, A. M., & Patire, A. D. (2016). A hybrid approach for short-term traffic state and travel time prediction on highways. *Transportation Research Record: Journal of the Transportation Research Board, 2554*, 60–68. doi:10.3141/2554-07

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, *19*(1), 153–160.

Bharti, A. K., Sekhar, C. R., & Chandra, S. (2018). Modeling travel time of car with varying demand on an urban mid-block section. *Journal of Intelligent Transportation Systems*, *22*(2), 99–105. doi:10.1080/15472450.2017.1408013

Chen, C. M., Liang, C. C., & Chu, C. P. (2018). Long-term travel time prediction using gradient boosting. *Journal of Intelligent Transportation Systems*, 1–16. doi:10.1080/15472450.2018.1542304

Cui, Z., Henrickson, K., Ke, R., & Wang, Y. (2018). High-order graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. arXiv preprint arXiv:1802.07007, (February).

Cui, Z., Ke, R., & Wang, Y. (2018). Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. arXiv Preprint arXiv:1801.02143.

Delhome, R., Billot, R., & El Faouzi, N.-E. (2017). Travel time statistical modeling with the Halphen distribution family. *Journal of Intelligent Transportation Systems*, 1–13.

Duan, Y., Lv, Y., & Wang, F.-Y. (2016). *Travel time prediction with LSTM neural network*. In 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 1053–1058.

Fan, S. K. S., Su, C. J., Nien, H. T., Tsai, P. F., & Cheng, C. Y. (2017). Using machine learning and big data approaches to predict travel time based on historical and real-time data from Taiwan electronic toll collection. *Soft Computing*, 1–12.

Habtemichael, F. G., & Cetin, M. (2016). Short-term traffic flow rate forecasting based on identifying similar traffic patterns. *Transportation Research Part C: Emerging Technologies*, *66*, 61–78.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554. doi:10.1162/neco.2006.18.7.1527

Hofleitner, A., Herring, R., & Bayen, A. (2012). Arterial travel time forecast with streaming data: A hybrid approach of flow modeling and machine learning. *Transportation Research Part B: Methodological*, *46*(9), 1097.

Huang, W., Song, G., Hong, H., & Xie, K. (2014). Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, *15*(5), 2191–2201. doi:10.1109/TITS.2014.2311123

Jenelius, E., & Koutsopoulos, H. N. (2013). Travel time estimation for urban road networks using low frequency probe vehicle data. *Transportation Research Part B: Methodological*, *53*, 64–81. doi:10.1016/j.trb.2013.03.008

Ke, R., Li, W., Cui, Z., & Wang, Y. H. (2018). *Multi-lane traffic pattern learning and forecasting using convolutional neural network*. In COTA International Symposium on Emerging Trend in Transportation.

Kumar, B. A., Vanajakshi, L., & Subramanian, S. C. (2017). A hybrid model based method for bus travel time estimation. *Journal of Intelligent Transportation Systems*, 1–17.

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. Y. (2015). Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, *16*(2), 865–873.

Ma, X., Yu, H., Wang, Y., & Wang, Y. (2015). Large-scale transportation network congestion evolution prediction using deep learning theory. *PLoS ONE*, *10*(3), e0119044.

Ma, Z., Koutsopoulos, H. N., Ferreira, L., & Mesbah, M. (2017). Estimation of trip travel time distribution using a generalized Markov chain approach. *Transportation Research Part C: Emerging Technologies*, *74*, 1–21. doi:10.1016/j.trc.2016.11.008

Nikovski, D., Nishiuma, N., Goto, Y., & Kumazawa, H. (2005). Univariate short-term prediction of road travel times. In *IEEE Conference on Intelligent Transportation Systems*, Proceedings, ITSC, pp. 1074–1079.

Qiao, W., Haghani, A., Shao, C.-F., & Liu, J. (2016). Freeway path travel time prediction based on heterogeneous traffic data through nonparametric model. *Journal of Intelligent Transportation Systems*, *20*(5), 438–448. doi:10.1080/15472450.2016.1149700

Rahmani, M., Jenelius, E., & Koutsopoulos, H. N. (2015). Non-parametric estimation of route travel time distributions from low-frequency floating car data. *Transportation Research Part C: Emerging Technologies*, *58*, 343–362. doi:10.1016/j.trc.2015.01.015

Sanaullah, I., Quddus, M., & Enoch, M. (2016). Developing travel time estimation methods using sparse GPS data. *Journal of Intelligent Transportation Systems*, *20*(6), 532–544. doi:10.1080/15472450.2016.1154764

Siripanpornchana, C., Panichpapiboon, S., & Chaovalit, P. (2017). Travel-time prediction with deep learning. IEEE Region 10 Annual International Conference, Proceedings/TENCON, pp. 1859–1862.

Tang, K., Chen, S., & Khattak, A. J. (2018a). Personalized travel time estimation for urban road networks: A tensor-based context-aware approach. *Expert Systems with Applications*, *103*, 118–132. doi:10.1016/j.eswa.2018.02.033

Tang, K., Chen, S., & Khattak, A. J. (2018b). A spatial–temporal multitask collaborative learning model for multistep traffic flow prediction. *Transportation Research Record: Journal of the Transportation Research Board*, *2672*(45), 1–13. doi:10.1177/0361198118790330

Tang, K., Chen, S., & Liu, Z. (2018). Citywide spatial-temporal travel time estimation using big and sparse trajectories. *IEEE Transactions on Intelligent Transportation Systems*, *99*, 1–12.

Tang, K., Chen, S., Liu, Z., & Khattak, A. J. (2018). A tensor-based Bayesian probabilistic model for citywide personalized travel time estimation. *Transportation Research Part C: Emerging Technologies*, *90*, 260–280. doi:10.1016/j.trc.2018.03.004

Van Lint, H., & Van Hinsbergen, C. (2012). Short-term traffic and travel time prediction models. *Artificial Intelligence Applications to Critical Transportation Issues*, 22–41.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). *Extracting and composing robust features with denoising autoencoders*. In Proceedings of the 25th

International Conference on Machine learning – ICML '08, pp. 1096–1103.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion pierre-antoine manzagol. *Journal of Machine Learning Research*, *11*, 3371–3408.

Wang, Y., Zheng, Y., & Xue, Y. (2014). *Travel time estimation of a path using sparse trajectories*. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD'14), (5), 25–34.

Westgate, B. S., Woodard, D. B., Matteson, D. S., & Henderson, S. G. (2016). Large-network travel time distribution estimation for ambulances. *European Journal of Operational Research*, *252*(1), 322–333. doi:10.1016/j.ejor.2016.01.004

Yu, H., Wu, Z., Wang, S., Wang, Y., & Ma, X. (2017). Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors*, *17*(7), 1501. doi:10.3390/s17071501

Yuan, J., Zheng, Y., Zhang, C., Xie, X., & Sun, G. Z. (2010). *An Interactive-Voting based Map Matching algorithm*. In Proceedings – IEEE International Conference on Mobile Data Management, pp. 43–52.

Zhan, X., Hasan, S., Ukkusuri, S. V., & Kamga, C. (2013). Urban link travel time estimation using large-scale taxi data with partial information. *Transportation Research Part C: Emerging Technologies*, *33*, 37–49.

Zhan, X., Ukkusuri, S. V., & Yang, C. (2016). A Bayesian mixture model for short-term average link travel time estimation using large-scale limited information trip-based data. *Automation in Construction*, *72*, 237–246. doi:10.1016/j.autcon.2015.12.007

Zheng, F., Li, J., van Zuylen, H., Liu, X., & Yang, H. (2018). Urban travel time reliability at different traffic conditions. *Journal of Intelligent Transportation Systems*, *22*(2), 106–120. doi:10.1080/15472450.2017.1412829

Zheng, F., & Van Zuylen, H. (2013). Urban link travel time estimation based on sparse probe vehicle data. *Transportation Research Part C: Emerging Technologies*, *31*, 145–157. doi:10.1016/j.trc.2012.04.007

Zheng, Y. (2015). Trajectory data mining: An overview. *ACM Transactions on Intelligent Systems and Technology*, *6*(3), 29. doi:10.1145/2743025

Zhou, T., Han, G., Xu, X., Lin, Z., Han, C., Huang, Y., & Qin, J. (2017). δ-agree AdaBoost stacked autoencoder for short-term traffic flow forecasting. *Neurocomputing*, *247*, 31–38. doi:10.1016/j.neucom.2017.03.049