Transportation Research Part C 72 (2016) 168–181

Contents lists available at ScienceDirect

Transportation Research Part C

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

An efficient realization of deep learning for traffic data imputation

Yanjie Duan, Yisheng Lv*, Yu-Liang Liu, Fei-Yue Wang

The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 28 April 2016
Received in revised form 27 September 2016
Accepted 27 September 2016
Available online 1 October 2016

Keywords:
Traffic data imputation
Deep learning
Missing data

ABSTRACT

Traffic data provide the basis for both research and applications in transportation control, management, and evaluation, but real-world traffic data collected from loop detectors or other sensors often contain corrupted or missing data points which need to be imputed for traffic analysis. For this end, here we propose a deep learning model named denoising stacked autoencoders for traffic data imputation. We tested and evaluated the model performance with consideration of both temporal and spatial factors. Through these experiments and evaluation results, we developed an algorithm for efficient realization of deep learning for traffic data imputation by training the model hierarchically using the full set of data from all vehicle detector stations. Using data provided by Caltrans PeMS, we have shown that the mean absolute error of the proposed realization is under 10 veh/5-min, a better performance compared with other popular models: the history model, ARIMA model and BP neural network model. We further investigated why the deep leaning model works well for traffic data imputation by visualizing the features extracted by the first hidden layer. Clearly, this work has demonstrated the effectiveness as well as efficiency of deep learning in the field of traffic data imputation and analysis.

1. Introduction

Traffic data are essential and fundamental for both transportation research and applications (Wang, 2010; Nan et al., 2008). Many institutes have established systems (Zhong et al., 2005), e.g. Caltrans (California Department of Transportation) PeMS (Performance Measurement System) (PeMS, 2014), to store and analyze traffic data like flow, occupancy and speed. With traffic data, engineers and researchers can understand and assess the performance of traffic systems, discover existent transportation problems, and make better decisions (Chen and Bell, 2002). However, traffic data collected from real traffic systems often have corrupted or missing data points (Li et al., 2014; Smith and Babiceanu, 2004) due to detector and communication malfunctions, adversely affecting traffic management (Sharma et al., 2004). Thus traffic data imputation is required in traffic data collection and storage.

Generally, traffic data imputation is to estimate the corrupted or missing traffic data. Because of the necessity of traffic data imputation, the complex patterns of traffic data and the diversity of application scenarios, there have been many researchers investigating this problem using a wide range of methods (Fernndez-Moctezuma et al., 2007; Tang et al., 2015; Tan et al., 2013; Castrillon et al., 2012; Zhang and Liu, 2009). These methods are mainly classified into three categories: prediction, interpolation and statistical learning (Li et al., 2014). Prediction methods typically use historical data collected

* Corresponding author.
E-mail addresses: duanyanjie2012@ia.ac.cn (Y. Duan), yisheng.lv@ia.ac.cn (Y. Lv), liuyuliangxp@126.com (Y.-L. Liu), feiyue@ieee.org (F.-Y. Wang).

http://dx.doi.org/10.1016/j.trc.2016.09.015
0968-090X/© 2016 Elsevier Ltd. All rights reserved.

from one site to build a prediction model and predict the values of corrupted or missing data points for the same site (Zhong and Sharma, 2006; Li et al., 2015; Gan et al., 2015). The autoregressive integrated moving-average (ARIMA) model (Nihan, 1997) is one of the commonly used methods to predict the imputed data points one by one. Interpolation methods replace the corrupted or missing data points with history data points or neighboring data points. One straightforward model is to fill a corrupted or missing data point with a known historical data collected at the same site at the same daily time interval but in previous days (Allison, 2001). This model is also known as the history model. Models using neighboring data points to interpolate corrupted or missing data points often use the data of neighboring sites or neighboring states to estimate the values of corrupted or missing data points for the current site. In these models, each corrupted or missing data point is estimated by the average or the weighted average of the neighboring data (Chen et al., 2003; Al-Deek et al., 2004; Van Lint et al., 2005; Kim and Lovell, 2006). A typical one is k-NN (Liu et al., 2008; Chang et al., 2012), of which the key work is to determine the neighbors by an appropriate distance metric. Statistical learning methods often use the observed data to learn a scheme, then inference the corrupted or missing data points in an iterated fashion (Smith et al., 2003; Wang et al., 2008; Qu et al., 2009; Li et al., 2013). A classical method is the Markov Chain Monte Carlo (MCMC) multiple imputation method (Ni and Leonard, 2005; Ni et al., 2005; Steimetz and Brownstone, 2005; Farhan and Fwa, 2013). The basic idea of MCMC multiple imputation method is to treat a corrupted or missing data point's value as a parameter of interest, and estimate the parameter by drawing a series of samples of the parameter. That means the imputation of the corrupted or missing data point is a combination of multiple imputed values instead of only one value. Neural network (Zhong et al., 2004; Ming et al., 2004; Lv et al., 2015) is a promising method to obtain better imputation performance than traditional imputation methods given more observed data. In addition, some researchers use traffic simulation models such as DynaSmart (Mahmassani et al., 1992), DynaMIT (Ben-Akiva et al., 1998), Vissim (Fellendorf, 1994), Paramics (Cameron and Duncan, 1996), TransWorld (Wang, 2010) to perform traffic data imputation (Muralidharan et al., 2009; Muralidharan and Horowitz, 2009). Whatever method a specific traffic data imputation realization adopts, the key idea is utilizing the potential temporal and spatial information in the data.

With the increasing quantity of traffic data, imputing corrupted or missing data points automatically and efficiently has become essential and critical. For this end, here we propose a deep learning model named denoising stacked autoencoders (DSAE) for traffic data imputation. This model treats traffic data containing observed normal data points, corrupted or missing data points as a corrupted vector, and transforms traffic data imputation into clean data recovering or corrupted data denoising. We firstly evaluate the impact of temporal and spatial factors on traffic data imputation, and then propose a hierarchically training algorithm to efficiently build the model. This algorithm can take advantage of inherent information embedded in large scale data collected from a traffic network and achieve better performance.

The rest of this paper is organized as the following: Section 2 describes the denoising stacked autoencoders. Section 3 presents the realization schemes for traffic data imputation based on DSAE, and discusses the model performance on solving random corruption. Section 4 further explores the model performance on solving continuous corruption. Section 5 concludes this paper.

2. Methodology

Deep learning has been successfully applied in image classification, speech recognition, natural language processing and computer gaming (LeCun et al., 2015; Silver et al., 2016; Wang et al., 2016). In this paper, we propose a novel deep learning model named denoising stacked autoencoders (DSAE) and would like to see how effective DSAE are at imputing traffic flow data. DSAE has two basic blocks including autoencoders (AEs) (Bengio et al., 2007) and denoising autoencoders (DAE) (Vincent et al., 2008). An AE can extract features from original input data. AEs can be stacked to form a deep network to obtain an abstract representation of the input in a gradual feature extraction way. A DAE is a stochastic version of the AE and can capture statistical dependencies between the inputs. Also DAEs can be connected to form a deep network named stacked denoising autoencoders (SDAE) (Vincent et al., 2010). SDAE has been proved to be able to learn inherent features and correlations in data and extract useful higher level representations. Both DAE and SDAE have the ability of cleaning or denoising data. Here, for the purpose of traffic data imputation we combine a DAE playing the role of corrupted traffic data denoising with stacked AEs helping extract features from high dimension traffic data to build a deep learning model named DSAE which retains the advantages of both DAE and stacked AEs. DSAE recovers data through feature extraction and statistical dependency learning. The following paragraphs introduce AE and DAE, and illustrate the construction of a DSAE.

2.1. AE

An AE is structured by its encoder part and decoder part as shown in Fig. 1. The encoder $f\theta$ maps an input vector $\mathbf{x}$ into hidden representation $\mathbf{h}$, i.e. $\mathbf{h} = f_\theta(\mathbf{x})$. f_θ is typically a nonlinear transformation in the following form:

$$f_\theta(\mathbf{x}) = s(\mathbf{x}\mathbf{W}^T + \mathbf{b}), \tag{1}$$

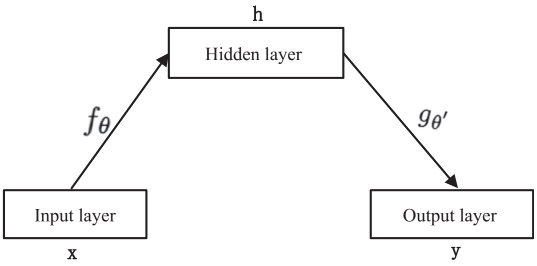

Fig. 1. The structure of an AE.

where θ represents parameters containing **W** and **b**. **W** is the weight matrix between the hidden layer and the input layer. **b** is the bias vector of the hidden layer. The decoder $g_{\theta'}$ maps the hidden representation **h** back to a reconstructed vector **y** of the input vector **x**, i.e. $\mathbf{y} = g_{\theta'}(\mathbf{h})$. $g_{\theta'}$ is also typically a nonlinear transformation in the following form:

$$g_{\theta'}(\mathbf{h}) = s(\mathbf{h}\mathbf{W}'^{T} + \mathbf{b}'), \tag{2}$$

where θ' represents parameters containing $\mathbf{W}'$ and $\mathbf{b}'$. $\mathbf{W}'$ is the weight matrix between the output layer and the hidden layer. $\mathbf{b}'$ is the bias vector of the output layer. In Eqs. (1) and (2), s is called the activation function.

Training an autoencoder is the process of minimizing the reconstruction error by solving the following optimization problem:

$$(\theta, \theta') = \arg\min_{(\theta,\theta')} L(X, Y), \tag{3}$$

where X is a set of input vector **x** and Y is the corresponding set of reconstructed vector **y**. The training algorithm is summarized in Algorithm 1. The hidden representation of a trained AE is seen as the feature of the input vector. The feature will be used to extract higher level representation in a DSAE.

Algorithm 1. Training an autoencoder

```
Require: data set $X = \{\mathbf{x}\}, \mathbf{x} \in \mathbb{R}^N$, the number of hidden nodes $n_h$, the iterations T
Initialization: initialize the weights and biases randomly: $\mathbf{W}(n_h \times N), \mathbf{W}'(N \times n_h), \mathbf{b}(1 \times n_h), \mathbf{b}'(1 \times N)$;
for $i$ = 1 to T do
    Perform forward propagation to compute $Y$
    Compute output error: $X - Y$
    Perform backward propagation to compute $\Delta\theta, \Delta\theta'$
    Update $\theta, \theta' : \theta = \theta + \Delta\theta, \theta' = \theta' + \Delta\theta'$
end for
```

2.2. DAE

A DAE is a variant of an AE. The structure of a DAE is shown in Fig. 2 and it is very similar to an AE except the input part of the network, which takes a corrupted version of the original input feeding into the input layer. That means DAE is trained to reconstruct the input from a corrupted version of it. The corrupted version $\widetilde{\mathbf{x}}$ of an input vector **x** is obtained by a stochastic mapping:

$$\widetilde{\mathbf{x}} \sim q_{\mathscr{D}}(\widetilde{\mathbf{x}}|\mathbf{x}). \tag{4}$$

Then $\widetilde{\mathbf{x}}$ is fed to the input layer. Through encoding and decoding according to Eqs. (1) and (2), the reconstructed vector **y** is obtained. To reconstruct the original clean input, the reconstruction error is measured between **y** and **x** instead of $\widetilde{\mathbf{x}}$. Thus

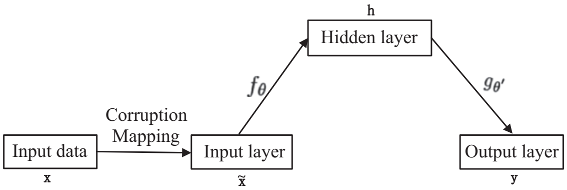

Fig. 2. The structure of a DAE.

training a DAE is the process of minimizing the reconstruction error through solving the optimization problem similar to Eq. (3). The hidden representation of a well-trained DAE is seen as a good and useful representation because it can recover the corresponding clean input based on a corrupted version of it. Besides, for a new corrupted input fed to the input layer, a well-trained DAE can recover its corresponding clean version.

2.3. DSAE

A DSAE is structured with a DAE at the bottom and stacked AEs in the middle layers, shown in Fig. 3. The bottom DAE is aimed to recover the clean input from a corrupted version of it while the internal stacked AEs can help to extract features from the input layer. For a corrupted input vector $\widetilde{\mathbf{x}}$ fed to the input layer, it is mapped to the reconstructed vector $\mathbf{y}$ through the following transformation:

$$\mathbf{h}_l = \begin{cases} s(\widetilde{\mathbf{x}}\mathbf{W}_1^T + \mathbf{b}_1) & l = 1 \\ s(\mathbf{h}_{l-1}\mathbf{W}_l^T + \mathbf{b}_l) & L \geqslant l > 1 \end{cases}, \tag{5}$$

$$\mathbf{y} = s(\mathbf{h}_{L+1}\mathbf{W}_{L+1}^T + \mathbf{b}_{L+1}), \tag{6}$$

where $\mathbf{h}_l$ is the hidden representation of the lth hidden layer, $\mathbf{W}_l$ is the weight matrix between the lth hidden layer and the layer below it, $\mathbf{b}_l$ is the bias vector of the lth hidden layer, $\mathbf{W}_{L+1}$ is the weight matrix between the output layer and the Lth hidden layer, $\mathbf{b}_{L+1}$ is the bias vector of the output layer.

Training a DSAE is again the process of minimizing the reconstruction error by solving the following optimization problem:

$$\Theta = \arg\min_{\Theta} L(X, Y), \tag{7}$$

where Θ represents the parameters including $\mathbf{W}_l$ and $\mathbf{b}_l (l = 1, 2, \ldots, L+1)$. Different from training an AE or a DAE, this process includes two training steps: pretraining and fine-tuning. The pretraining step trains the parameters of each AE while the fine-tuning step adjusts all the parameters in a DSAE. The process is summarized in Algorithm 2. After training, a DSAE can reconstruct the corresponding clean input vector from a corrupted version of it.

Algorithm 2. Training denoising stacked autoencoders

```
Require: data set $X = \{\mathbf{x}\}, \mathbf{x} \in \mathbb{R}^N$, the number of hidden layers $L$, the number of hidden nodes $n_l, l = 1, 2, \ldots, L$, the
  pretraining iterations T, the fine-tuning iterations $T'$
Stochastic mapping: map X into $\widetilde{X}, \widetilde{X} = \{\widetilde{\mathbf{x}}\}, \widetilde{\mathbf{x}} \in \mathbb{R}^N$;
Step 1: Pretraining
$\widetilde{X}$ is fed to the input layer;
Train the first AE using Algorithm 1;
for $l$ = 2 to L do
    $H_{l-1}$ is fed to the $l$th AE
    Train the $l$th AE using Algorithm 1
end for
Step 2: Fine-tuning
Initialization: initialize $\mathbf{W}_{L+1}(N \times n_L), \mathbf{b}(1 \times N)$ randomly;
for$i$ = 1 to $T'$ do
    Perform forward propagation to compute $Y$
    Compute output error: $X - Y$
    Perform backward propagation to compute $\Delta\Theta$
    Update $\Theta : \Theta = \Theta + \Delta\Theta$
end for
```

3. Realization for traffic data imputation

Traffic data imputation is to recover the values of corrupted or missing data points in a traffic data set. Reliable imputation needs the potential temporal and spatial information contained in traffic data. Deep network can represent inherent features and correlations in data, and has the potential to impute traffic data. This section discusses the performance of imputing traffic flow data with the deep learning model DSAE on the Caltrans PeMS (PeMS, 2014) database. Here, assume that all the corrupted or missing data points have been detected and marked with zero, which is the actual situation in Caltrans PeMS.

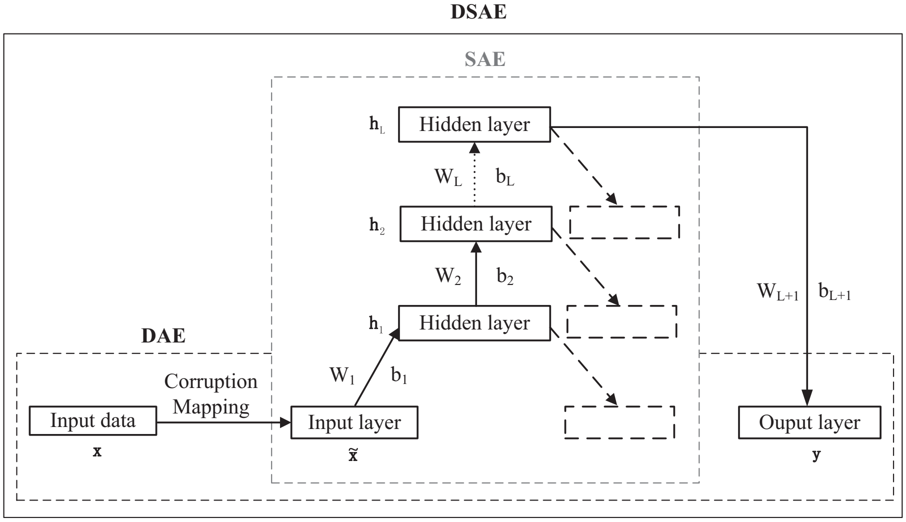

Fig. 3. The structure of a DSAE.

3.1. Data description

Traffic flow data provided by Caltrans PeMS (PeMS, 2014) are used for this study. Caltrans PeMS consists of more than 15,000 detector stations and collects traffic data every 30 s. Caltrans PeMS then aggregates the 30-s data into 5-min increments (Varaiya, 2001). We exploit the 5-min traffic flow data of Year 2013 from District 5: Central Coast. There are 153 vehicle detector stations (VDSs) in this district. However, we only use data collected from 147 VDSs because 6 VDSs involve null numbers due to some unknown reason and we remove them. Further, the data collected from the 147 VDSs on March 10th and September 17th in the year of 2013 consist of less than one day's time stamps. The two days are removed and there are 363 days. Thus we have a traffic data set including 147 VDSs each of which has 104,544 data points.

Fig. 4(a) shows one week's traffic flow data collected at VDS 500010102. It is obvious that traffic flow data have different patterns on weekdays and at weekends, i.e. traffic data collected on normal weekdays usually have two peaks while traffic data collected at weekends usually have one peak. This pattern can help imputing traffic data, and many researchers treat these two kinds of patterns separately to estimate corrupted or missing traffic data. Simultaneously, there exists similarity and periodicity through days in the traffic flow series. Thus we divide traffic data into days and regard each day's traffic flow data as a vector **x**. Therefore the dimension of traffic data vector (5-min interval data vector of one day) is 288. Each traffic data vector has a property indicating whether it is a weekday or not. Through the observation of a large amount of traffic data, we find that some vectors take on different patterns from their official statements of weekday property. In order to solve this problem and distinguish weekdays from non-weekdays like weekends and holidays, we apply a clustering method to traffic data vectors. This procedure is summarized in Algorithm 3. Assume that vectors of the same date from different

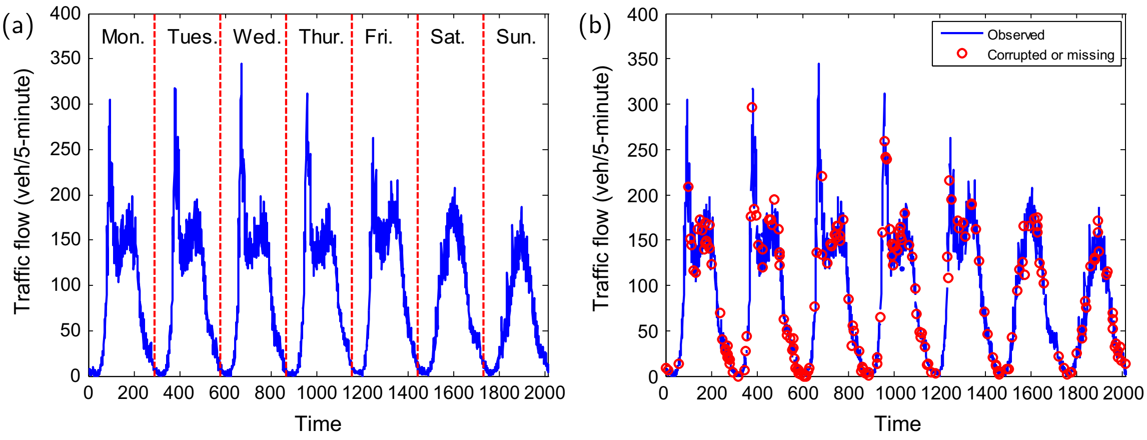

Fig. 4. An example of traffic flow series: (a) clean traffic flow series and (b) corrupted traffic flow series.

VDSs have the same weekday property. Thus we only need to apply Algorithm 3 to one VDS. The initial centroids in the algorithm are chosen as a typical weekday vector and a typical weekend vector. The clustering procedure results in two clusters. One cluster is traffic data vectors with weekday pattern and the other is traffic data vectors with non-weekday pattern. For a new corrupted vector $\tilde{\mathbf{x}}$, calculate its distances with the latest μ_1, μ_2 in the uncorrupted dimensions and assign it to the closer cluster. After clustering, we obtain 244 weekdays and 119 non-weekdays in our data set.

Algorithm 3. k-means clustering traffic data vectors

```
Input: Traffic data vectors $X = \{\mathbf{x}\}, \mathbf{x} \in \mathbb{R}^{288}$;
Output: k clusters;
1: Initialization: k = 2, k cluster centroids $\mu_1, \mu_2 \in \mathbb{R}^{288}$;
2: repeat
3:     Assignment step: Assign each traffic data item to the cluster whose centroid is nearest in Euclidean distance;
4:     Update step: Calculate the means of the new clusters to be the new centroids;
5: until Clusters do not change.
```

The goal of traffic data imputation is to recover the complete and clean data like Fig. 4(a) from its corrupted version like Fig. 4(b). A desired imputation method should fill in the corrupted or missing data points automatically using all the information contained in the observed data. In a traffic data system, there are many VDSs whose traffic data and their corrupted patterns are diverse and complex. Instead of building many different imputation models for VDSs, we prefer a more general model which can be applied to all the VDSs efficiently. Our deep learning method based on DSAE is a potential choice.

3.2. Model settings and evaluation criteria

The meta parameters of our model include activation functions, the number of layers, the number of nodes in each layer. Proper combination of these parameter values is important and the parameters are usually determined through experiences and experiments. Here, we did grid-search experiments to determine these parameters. Finally, the sigmoid function

$$s(x) = \frac{1}{1 + e^{-x}}$$

is selected as the activation function, and the deep network has 3 hidden layers with 144, 72, 144 nodes for each hidden layer, respectively. This set of meta parameters will be used by default in the following realizations.

In order to evaluate the performances of the proposed method, we adopt three criteria to measure the error of the imputed data. They are mean absolute error (MAE),

$$MAE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}|x_{ij} - y_{ij}|}{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}},$$

root mean square error (RMSE),

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}(x_{ij} - y_{ij})^2}{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}}},$$

and mean relative error (MRE),

$$MRE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}\frac{|x_{ij} - y_{ij}|}{x_{ij}}}{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}},$$

where M is the total number of test vectors, N is the dimension of each test vector, I is an indicator function,

$$I_{ij} = \begin{cases} 0 & \tilde{x}_{ij} \text{ is observed}, \\ 1 & \tilde{x}_{ij} \text{ is corrupted or missing}. \end{cases}$$

3.3. Impact of spatial and temporal factors

Spatial and temporal factors have impact on traffic data imputation. In this section we present how we find an optimal realization scheme of our model by considering spatial and temporal information to impute traffic data. In terms of spatial factors, we mainly consider whether data are from single VDS or multiple VDSs in adjacent locations. When it comes to temporal factors, we mainly consider the differences between weekdays and non-weekdays. In order to describe these realizations clearly, we classify the realizations from the perspective of spatial factors and consider the influence of temporal factors in each realization. For the convenience of evaluations, we select the test data from the same VDS 500010102. In the sequel we evaluate the performances of realizations starting from the simplest one and optimize it gradually. Here we focus on random corruption (RC) where the corrupted or missing data points are randomly distributed in corrupted vectors. Define RC rate as

$$\frac{\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}}{MN} \times 100\%.$$

3.3.1. Using data collected from single VDS

One simple realization of the deep learning model for traffic data imputation is to consider traffic data collected from single VDS. This is what we did in our previous work (Duan et al., 2014), where we just evaluated the imputation of traffic data from weekdays. In this paper, we further conduct two additional experiments to evaluate the performances of realizations considering temporal factors: data collected just on non-weekdays, and data collected both on weekdays and non-weekdays. The three experiments and the corresponding results of evaluation criteria are listed in Table 1.

The ratio between training data and test data is 4:1 which is the default ratio adopted in the rest experiments. With the default meta parameters, we train the model using the technique of early stopping (Erhan et al., 2010; Hinton, 2012). Table 1 shows the performance indexes on the test data. All the performances in Table 1 are tested under the RC rate of 30%, and the same RC rate will be used in the following tables by default. It is obvious that the realization considering both weekdays and non-weekdays performs best. The results indicate that the exploitation of data with different patterns on weekdays and non-weekdays can contribute to improve traffic data imputation. Furthermore, we conduct a series of experiments to test our model with data on weekdays and non-weekdays under different RC rates ranging from 5% to 50%. The results are shown in Fig. 5, where the MAE ranges from 10.5 to 13.2 veh/5-min, the RMSE ranges from 15.4 to 19.8 veh/5-min and the MRE ranges from 23.1% to 29.1%. It shows that all the errors follow the trend of increasing slowly along with the growth of RC rate. This trend meets our expectation and demonstrates the rationality of our deep learning model for traffic data imputation.

3.3.2. Using data collected from upstream and/or downstream VDSs

Traffic data from upstream and/or downstream VDSs may have some relationship with that from the current VDS. On researching the influences of upstream and downstream VDSs on the current VDS, we focus on two possible cases. One case is that the upstream and downstream VDSs also include corrupted or missing data points with the same RC rate of the current VDS. The other is that only the current VDS includes corrupted or missing data points. Indicated by the results above, the realization considering both weekdays and non-weekdays performs better. Thus we consider the data collected on weekdays and non-weekdays in both cases.

In the first case, we can combine vectors of traffic data collected on the same day from the current VDS, the upstream and/or downstream VDSs as the input of the deep network model. Thus the dimension of the input vector to the model is one or two times larger than just using data from the current VDS. Simultaneously, the size of the new training dataset remains the same. We define this input mode as an augment-vector mode. In this mode, the output target of our model is still the clean data of the current VDS, which is the only clean data we need for model training. Or we can augment the training dataset by including data from the current VDS, the upstream and/or downstream VDSs as a new dataset. Thus the size of the new training dataset is one or two times larger while the dimension of each sample in the dataset keeps the same. We define this input mode as an augment-dataset mode. In this mode, the output target of our model is the clean data vector corresponding to each input vector. And the clean data we need for model training are from the current VDS, the upstream and/or downstream VDSs. The performances of two input modes considering spatial factors are shown in Table 2. The results show that the performances of the realizations considering upstream and/or downstream VDSs including corrupted or missing data points are similar. These realization schemes have not reduced the imputation errors obviously relative to the last scheme in Table 1.

In the other case, the data from upstream and/or downstream VDSs are clean and needn't to be imputed. We combine vectors of traffic data collected on the same day from the current VDS, the upstream and/or downstream VDSs as the input of the proposed model. The combined input vector consists of the corrupted vector from the current VDS and the clean vectors from the upstream and/or downstream VDSs. Thus the dimension of the new input vector to the model is one or two

Table 1
Experiments set and performances of single VDS considering temporal factors.

Time type	MAE	RMSE	MRE
Weekdays	12.1	17.9	0.323
Non-weekdays	13.9	20.9	0.309
Weekdays and non-weekdays	**12.0**	**17.5**	**0.284**

Bold values are the best values in term of model performance in each column.

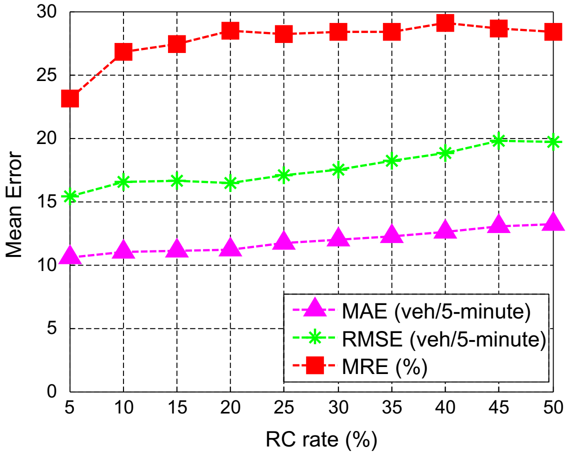

Fig. 5. Imputation performance considering weekdays and non-weekdays.

times larger than just using data from the current VDS. This input mode is also called an augment-vector mode similar to the above except that the input vector here contained complete information from upstream and/or downstream. In this case, the output target of our model is the clean data of the current VDS. The performances of the proposed model considering spatial factors are shown in Table 3. The results show that the performances of all the realizations considering complete upstream and/or downstream information are better than that using the current VDS only. The difference between Tables 2 and 3 indicates that clean data containing complete information from upstream and/or downstream contribute to single VDS's traffic data imputation while corrupted data are uncertain to do that.

3.3.3. Using data collected from all VDSs

Research (Krizhevsky et al., 2012; Donahue et al., 2014; Girshick et al., 2014) in deep learning has demonstrated that different objects' data can be used to train a deep network and extract features which will improve the recognition accuracy of a specific object. Therefore we try to use all VDSs instead of just neighboring VDSs to train our model. In this situation, the dataset's order of magnitude changes thus we reevaluate the influence of temporal factors. The experiments and their performances are listed in Table 4. The results show that all the realizations using data from all VDSs perform better than that just using data from single VDS or that using data from single VDS, its upstream and/or downstream VDSs for almost all the criteria. That indicates large amount of different traffic data patterns from different VDSs contribute to single VDS's traffic data imputation. Our experimental results agree with the research findings in deep learning that using data from other classes to train a model help obtain better classification results on data from the current class.

Table 2
Experiments set and performances of single VDS considering upstream and downstream incomplete information.

Input VDS(s)	Input mode	MAE	RMSE	MRE
Current	–	12.0	17.5	0.284
Upstream and current	Augment-vector	12.0	17.8	0.287
Upstream and current	Augment-dataset	11.7	17.4	0.279
Current and downstream	Augment-vector	**11.6**	**17.0**	0.285
Current and downstream	Augment-dataset	12.1	17.9	0.290
Upstream, current and downstream	Augment-vector	12.2	18.1	0.277
Upstream, current and downstream	Augment-dataset	12.2	18.0	**0.269**

Bold values are the best values in term of model performance in each column.

Table 3
Experiments set and performances of single VDS considering upstream and downstream complete information.

Input VDS(s)	Input mode	MAE	RMSE	MRE
Current	–	12.0	17.5	0.284
Upstream and current	Augment-vector	10.5	15.6	**0.257**
Current and downstream	Augment-vector	**10.5**	**15.6**	0.271
Upstream, current and downstream	Augment-vector	11.0	16.5	0.268

Bold values are the best values in term of model performance in each column.

Table 4
Experiments set and performances of Single VDS considering all VDSs and temporal factors.

Time type	Data source	MAE	RMSE	MRE
Weekdays	Single VDS	12.1	17.9	0.323
Weekdays	All VDSs	10.9	16.0	0.282
Non-weekdays	Single VDS	13.9	20.9	0.309
Non-weekdays	All VDSs	10.5	15.1	0.267
Weekdays and non-weekdays	Single VDS	12.0	17.5	0.284
Weekdays and non-weekdays	All VDSs	**10.2**	**14.9**	**0.265**

Bold values are the best values in term of model performance in each column.

3.4. Hierarchically training algorithm

For the purpose of realization scheme optimization, Tables 1–4 evaluate the impact of spatial and temporal factors on realization schemes. According to the results, using data from all VDSs to train the model achieves better performance totally speaking. Intuitively, the data from the current single VDS have more impact on its imputation performance while the above realizations treat all VDSs equally. To keep the advantage of using all VDSs and simultaneously reflect the different importance between the current single VDS and all the other VDSs, we design Algorithm 4 training our model hierarchically.

Algorithm 4. Hierarchically training DSAE for traffic data imputation

```
Require: training data from all VDSs in the current district $X_{all} = \{\mathbf{x}_{all}\}, \mathbf{x}_{all} \in \mathbb{R}^N$; training data from the current single
  VDS $X_{single} = \{\mathbf{x}_{single}\}, \mathbf{x}_{single} \in \mathbb{R}^N$;
Step 1 Train the model with $X_{all}$ using Algorithm 2;
 Step 2 Train the model obtained in the Step 1 with $X_{single}$;
Set the fine-tuning iterations $T''$;
Map $X_{single}$ into $\widetilde{X}_{single}, \widetilde{X}_{single} = \{\widetilde{\mathbf{x}}_{single}\}, \widetilde{\mathbf{x}}_{single} \in \mathbb{R}^N$;
for $i = 1$ to $T''$ do
  Perform forward propagation to compute $Y_{single}$
  Compute output error: $X_{single} - Y_{single}$
  Perform backward propagation to compute $\Delta\Theta$
  Update $\Theta : \Theta = \Theta + \Delta\Theta$
end for
```

As Algorithm 4 described, the first step is training the model using data from all VDSs just like the above realizations. The second step is training the model obtained in the first step using data from the current single VDS. Thus we get a model which has been trained twice and used the data from current VDS once more. To evaluate the performance of the algorithm, we conduct a series of experiments listed in Table 5. The results show that the algorithm performs better than that treating all VDSs equally for all criteria. Additionally, the realization considering data of both weekdays and non-weekdays is better. This scheme takes advantage of complex traffic data patterns of all VDSs as well as the greater importance of the current VDS's traffic data. It is the optimal one among all the realizations of the proposed DSAE model we have evaluated.

3.5. Model comparison

The realization schemes considering the impact of temporal and spatial factors have been evaluated and the optimal realization scheme has been proposed. In order to compare our model with other traffic data imputation models, we conduct series of experiments with the RC rate ranging from 5% to 50% and obtain the imputation results of the test data. The compared models are the history model (Allison, 2001), ARIMA model (Nihan, 1997) and BP (Back Propagation) neural network

Table 5
Experiments set and performances of Single VDS considering the difference between VDSs.

Time type	Training style	MAE	RMSE	MRE
Weekdays	All VDS	10.9	16.0	0.282
Weekdays	Hierarchically	10.2	15.3	0.232
Non-weekdays	All VDS	10.5	15.1	0.267
Non-weekdays	Hierarchically	9.7	14.2	0.233
Weekdays and Non-weekdays	All VDS	10.2	14.9	0.265
Weekdays and Non-weekdays	Hierarchically	**9.6**	**13.9**	**0.232**

Bold values are the best values in term of model performance in each column.

model (Zhong et al., 2004). Each model for comparison has been carefully tuned. For the history model, the window size of historical data is 5 days for weekdays and non-weekdays separately. For ARIMA model, we first remove the trend item then learn a ARMA(9,1) model for weekdays and non-weekdays separately. For the BP neural network model, the hidden layer structure is [144, 72, 144] and the model is also trained hierarchically, except the pretraining step specialized for deep learning, to achieve better performance. Fig. 6 shows the MAE, RMSE, MRE of the imputed data under different RC rates, respectively. Both the MAE and the RMSE of our model are the smallest under all RC rates. The MRE is smallest with most RC rates. Totally speaking our model performs better than all other models in comparison. Here, to compare the performances from the statistical aspect, we present the error distributions of the models under the RC rate of 30%. Both MAE and RMSE focus on the average distances between imputed data and real data while either distance can be represented by the absolute error. Thus we only need to concern about the distributions of absolute error and relative error. Fig. 7(a) shows that there are about 65% absolute errors under 10 veh/5-min and over 85% absolute errors under 20 veh/5-min with our deep learning method. Fig. 7(b) shows that there are over 70% relative errors under 20% with our deep learning model. Most errors are gathering in a low level better than the other models in comparison. That further confirms the mean error results above. The comparison demonstrates the efficiency of our deep learning model for traffic data imputation.

3.6. Features and imputation visualizations of DSAE

In the above paragraphs, we have evaluated the realizations of the proposed DSAE model for traffic data imputation considering temporal and spatial factors. Overall, the realization algorithm training the model hierarchically using all VDSs' data

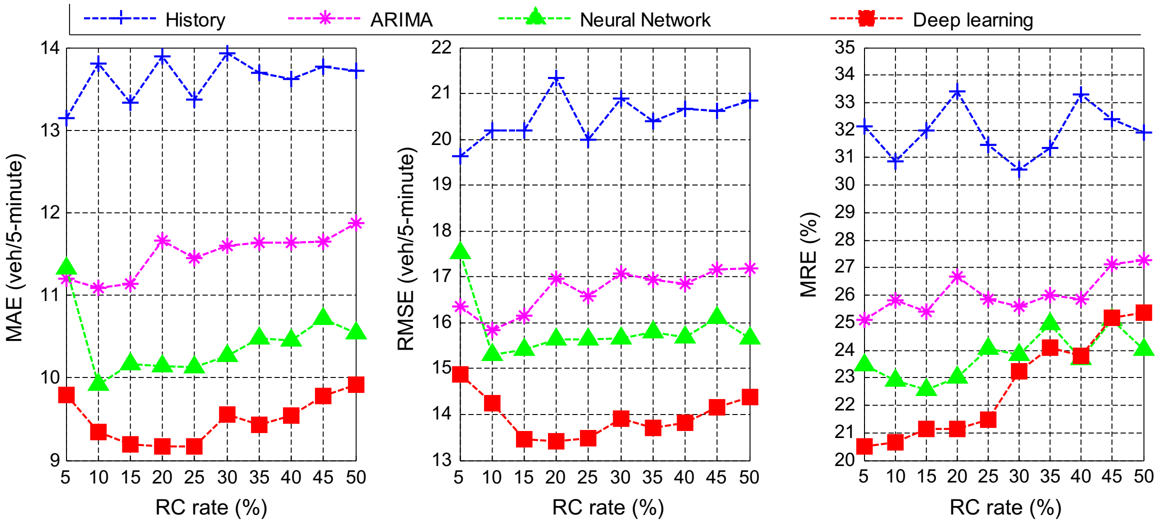

Fig. 6. The error of the imputed data.

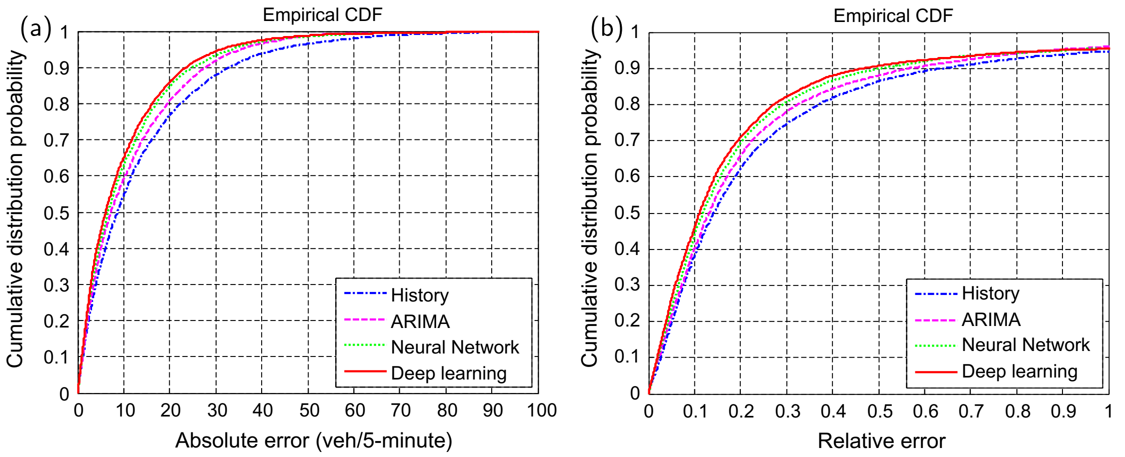

Fig. 7. Distributions of the errors: (a) cumulative distribution function of absolute error and (b) cumulative distribution function of relative error.

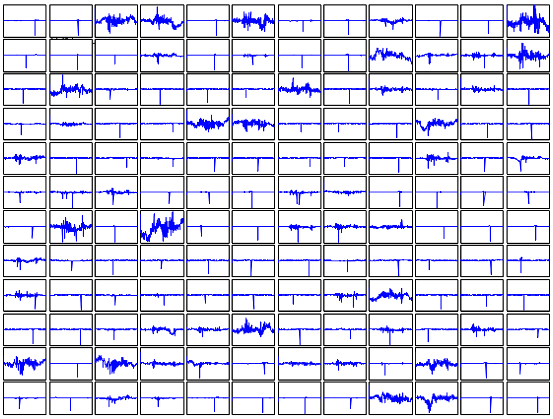

Fig. 8. Features extracted by the first hidden layer.

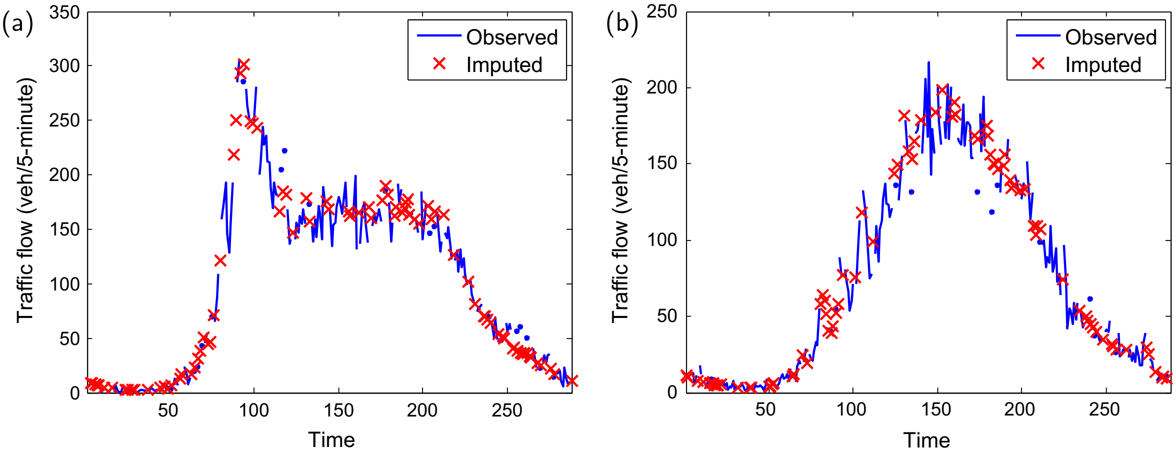

Fig. 9. Traffic flow series after imputation: (a) weekday and (b) non-weekday.

of weekdays and non-weekdays is the best one. Through comparison with other models, we have demonstrated the efficiency of our model. Here, we try to go deeply into the deep learning imputation model and explain the results it achieved. Fig. 8 visualizes the features extracted by the first hidden layer of the deep network in our model. Each mini map in the figure represents the weights linking the specific hidden node and the input nodes. In each mini map, the horizontal axis ranges from 1 to 288 since the number of input nodes is 288 and the longitudinal axis represents the value of each weight. There are 144 mini maps in the figure since the number of nodes in the first hidden layer n_1 equals 144. The figure indicates that different hidden nodes have learned to detect spikes at different positions in a traffic data vector while spikes are caused by corrupted or missing data points marked with zero. The features learned by the hidden nodes are useful for traffic data imputation. That may explain the reason why deep learning works on the task of traffic data imputation. Fig. 9 presents the traffic flow series taken from the test data after imputation using our deep learning model. We can see that the imputed data are quite consistent with the observed data whether on a weekday or a non-weekday.

4. Exploration on continuous corruption

To train the DSAE model, a critical procedure is mapping clean data $\mathbf{x}$ to its corresponding corrupted version $\widetilde{\mathbf{x}}$. Possible corrupted patterns should be integrated in the corruption mapping $q_{\mathcal{D}}(\widetilde{\mathbf{x}}|\mathbf{x})$. In the above section, we focus on random corruption. We employ the uniform distribution as the corruption mapping $q_{\mathcal{D}}(\widetilde{\mathbf{x}}|\mathbf{x})$, where each dimension of $\mathbf{x}$ has the same probability corrupted or missing and the probability equals the RC rate. Both the training set and the testing set adopt the same corruption mapping. Nevertheless, continuous corruption (CC) may occur in real traffic systems. Though the original intention of our model is to solve the problem of RC, we explore its ability on solving CC.

Here, we consider the situation of continuous corruption and random corruption (CCRC), where both CC and RC may occur in corrupted vectors. Fig. 10 is an example of CCRC. In a way, RC and CC are special cases of CCRC. The RC rate in

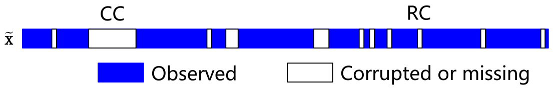

Fig. 10. A corrupted vector with CCRC.

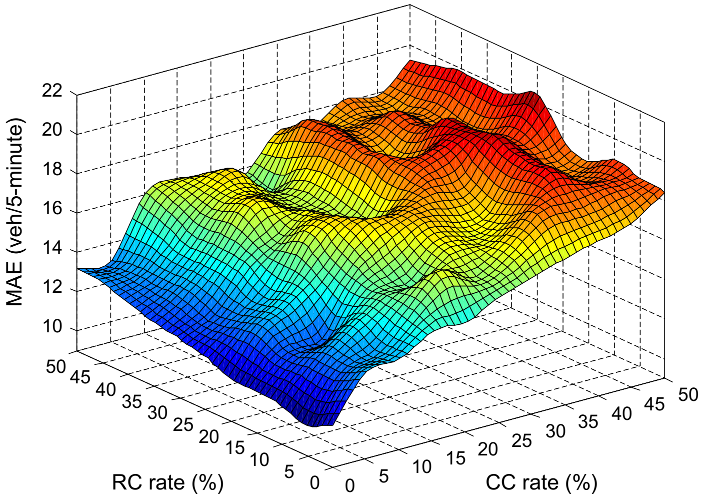

Fig. 11. The error of the imputed data in the situation of CCRC.

the above section is up to 50%, which is so high that CC may exist. Thus here we define CC to the situations where at least 5% continuous dimensions corrupted or missing. For a traffic data vector, that means a corruption as long as 1.2 h. For CCRC, we train and test the DSAE model adopting the corruption mapping, where the positions of CCRC are stochastic and both CC rate and RC rate range from 5% to 50%. After conducting a series of experiments using the simplest realization scheme, we find that the results of CCRC are reasonable. Here, we show the resulted MAE as an example in Fig. 11. Apart from CCRC, Fig. 11 also shows the results of CC when RC rate equals 0% and the results of RC when CC rate equals 0%. The figure indicates that the errors of CCRC are within an acceptable range, though the MAE of CC is larger than that of RC under the same corruption rate and CCRC tends to have bigger errors than only RC or CC. The results demonstrate that our model can be applied to CCRC and CC recovering.

5. Conclusion

In this paper, we view traffic data imputation as the problem of corrupted data recovering, and propose a deep learning model named DSAE to solve this problem. The proposed DSAE model is constructed using a DAE filled with SAE in the middle layers. We investigate 18 different realizations of the model considering temporal and spatial factors. In the aspect of temporal factors, the difference between weekdays and non-weekdays is considered. While in the aspect of spatial factors, the influences of a VDS's upstream, downstream and all the other VDSs are considered. Through a series of experiments using the data from Caltrans PeMS (PeMS, 2014), we evaluate the performances of the realizations considering these factors and find that the realization algorithm training the model hierarchically using all VDSs' data of weekdays and non-weekdays performs best. Additionally, this realization is convenient since it can be used for weekdays' and non-weekdays' traffic data imputation simultaneously. Through comparisons with the history model, ARIMA, and the BP neural network model, under the RC rate ranging from 5% to 50%, our model shows its advantages on the imputation accuracy. By visualizing the features extracted by the first hidden layer, we try to explain why the deep learning model works on traffic data imputation. And by visualizing the traffic flow series after imputation, we show the consistency of the imputed data output by our model with the observed data. Besides, this paper explores the ability of our model on CC recovering. The experimental results demonstrate that our model can be applied to it as well. Deep learning is promising in traffic data imputation, and deserves further investigation on developing new structures of deep learning models for imputation and applying these models in practice.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 61233001, 61203166, 71232006 and 61533019, and the Early Career Development Award of SKLMCCS.

References

Al-Deek, H., Venkata, C., Ravi Chandra, S., 2004. New algorithms for filtering and imputation of real-time and archived dual-loop detector data in I-4 data warehouse. Transp. Res. Rec.: J. Transp. Res. Board, 116–126.

Allison, P.D., 2001. Missing Data, vol. 13. Sage Publications.

Ben-Akiva, M., Bierlaire, M., Koutsopoulos, H., Mishalani, R., DynaMIT: a simulation-based system for traffic prediction. In: DACCORS Short Term Forecasting Workshop, The Netherlands, Citeseer.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., 2007. Greedy layer-wise training of deep networks. Adv. Neural Inform. Process. Syst. 19, 153.

Cameron, G.D., Duncan, G.I., 1996. PARAMICS – parallel microscopic simulation of road traffic. J. Supercomput. 10, 25–53.

Castrillon, F., Guin, A., Guensler, R., Laval, J., 2012. Comparison of modeling approaches for imputation of video detection data in intelligent transportation systems. Transp. Res. Rec.: J. Transp. Res. Board, 138–147.

Chang, G., Zhang, Y., Yao, D., 2012. Missing data imputation for traffic flow based on improved local least squares. Tsinghua Sci. Technol. 17, 304–309.

Chen, C., Kwon, J., Rice, J., Skabardonis, A., Varaiya, P., 2003. Detecting errors and imputing missing data for single-loop surveillance systems. Transp. Res. Rec.: J. Transp. Res. Board, 160–167.

Chen, H.B., Bell, M., 2002. Instrumented city database analysts using multi-agents. Transp. Res. Part C: Emerg. Technol. 10, 419–432.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T., 2014. DeCAF: a deep convolutional activation feature for generic visual recognition. In: Proceedings of The 31st International Conference on Machine Learning, pp. 647–655.

Duan, Y., Lv, Y., Kang, W., Zhao, Y., 2014. A deep learning based approach for traffic data imputation. In: Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on. IEEE, pp. 912–917.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S., 2010. Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11, 625–660.

Farhan, J., Fwa, T., 2013. Airport pavement missing data management and imputation with stochastic multiple imputation model. Transp. Res. Rec.: J. Transp. Res. Board, 43–54.

Fellendorf, M., 1994. VISSIM: a microscopic simulation tool to evaluate actuated signal control including bus priority. In: 64th Institute of Transportation Engineers Annual Meeting. Springer, pp. 1–9.

Fernndez-Moctezuma, R.J., Tufte, K., Maier, D., Bertini, R.L., 2007. Toward management and imputation of unavailable data in online advanced traveler information systems. In: Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE, pp. 24–29.

Gan, M., Chen, C.P., Li, H.X., Chen, L., 2015. Gradient radial basis function based varying-coefficient autoregressive model for nonlinear and nonstationary time series. IEEE Signal Process. Lett. 22, 809–812.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587.

Hinton, G.E., 2012. A practical guide to training restricted Boltzmann machines. In: Neural Networks: Tricks of the Trade, LNCS 7700. second ed. Springer-Verlag.

Kim, H., Lovell, D.J., 2006. Traffic information imputation using a linear model in vehicular ad hoc networks. In: Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE, pp. 1406–1411.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444.

Li, L., Li, Y., Li, Z., 2013. Efficient missing data imputing for traffic flow by considering temporal and spatial dependence. Transp. Res. Part C: Emerg. Technol. 34, 108–120.

Li, L., Su, X., Zhang, Y., Lin, Y., Li, Z., 2015. Trend modeling for traffic time series analysis: an integrated study. IEEE Trans. Intell. Transp. Syst. 16, 3430–3439.

Li, Y., Li, Z., Li, L., 2014. Missing traffic data: comparison of imputation methods. IET Intel. Transp. Syst. 8, 51–57.

Liu, Z., Sharma, S., Datla, S., 2008. Imputation of missing traffic data during holiday periods. Transp. Plann. Technol. 31, 525–544.

Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F.Y., 2015. Traffic flow prediction with big data: a deep learning approach. IEEE Trans. Intell. Transp. Syst. 16, 865–873.

Mahmassani, H., Hu, T., Jayakrishnan, R., 1992. Dynamic traffic assignment and simulation for advanced network informatics (DYNASMART). In: Proceedings of the 2nd International CAPRI Seminar on Urban Traffic Networks, Capri, Italy.

Ming, Z., Lingras, P., Sharma, S., 2004. Estimation of missing traffic counts using factor, genetic, neural, and regression techniques. Transp. Res. Part C: Emerg. Technol. 12, 139–166.

Muralidharan, A., Dervisoglu, G., Horowitz, R., 2009. Freeway traffic flow simulation using the link node cell transmission model. In: American Control Conference, 2009. ACC'09. IEEE, pp. 2916–2921.

Muralidharan, A., Horowitz, R., 2009. Imputation of ramp flow data for freeway traffic simulation. Transp. Res. Rec.: J. Transp. Res. Board, 58–64.

Nan, Z., Fei-Yue, W., Fenghua, Z., Dongbin, Z., Shuming, T., 2008. DynaCAS: computational experiments and decision support for ITS. IEEE Intell. Syst. 23, 19–23.

Ni, D., Leonard, J.D., Guin, A., Feng, C., 2005. Multiple imputation scheme for overcoming the missing values and variability issues in ITS data. J. Transp. Eng. 131, 931–938.

Ni, D., Leonard II, J., 2005. Markov chain monte carlo multiple imputation using Bayesian networks for incomplete intelligent transportation systems data. Transp. Res. Rec.: J. Transp. Res. Board, 57–67.

Nihan, N.L., 1997. Aid to determining freeway metering rates and detecting loop errors. J. Transp. Eng. 123, 454–458.

PeMS, 2014. Performance Measurement System. Available at: <http://pems.dot.ca.gov/>. (accessed June 2014).

Qu, L., Li, L., Zhang, Y., Hu, J., 2009. PPCA-based missing data imputation for traffic flow volume: a systematical approach. IEEE Trans. Intell. Transp. Syst. 10, 512–522.

Sharma, S., Lingras*, P., Zhong, M., 2004. Effect of missing values estimations on traffic parameters. Transp. Plan. Technol. 27, 119–144.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., 2016. Mastering the game of go with deep neural networks and tree search. Nature 529, 484–489.

Smith, B., Babiceanu, S., 2004. Investigation of extraction, transformation, and loading techniques for traffic data warehouses. Transp. Res. Rec.: J. Transp. Res. Board, 9–16.

Smith, B., Scherer, W., Conklin, J., 2003. Exploring imputation techniques for missing data in transportation management systems. Transp. Res. Rec.: J. Transp. Res. Board, 132–142.

Steimetz, S.S., Brownstone, D., 2005. Estimating commuters ''value of time" with noisy data: a multiple imputation approach. Transp. Res. Part B: Methodol. 39, 865–889.

Tan, H., Feng, G., Feng, J., Wang, W., Zhang, Y.J., Li, F., 2013. A tensor-based method for missing traffic data completion. Transp. Res. Part C: Emerg. Technol. 28, 15–27.

Tang, J., Zhang, G., Wang, Y., Wang, H., Liu, F., 2015. A hybrid approach to integrate fuzzy C-means based imputation method with genetic algorithm for missing traffic volume data estimation. Transp. Res. Part C: Emerg. Technol. 51, 29–40.

Van Lint, J., Hoogendoorn, S., van Zuylen, H.J., 2005. Accurate freeway travel time prediction with state-space neural networks under missing data. Transp. Res. Part C: Emerg. Technol. 13, 347–369.

Varaiya, P., 2001. Freeway Performance Measurement System, PeMS v3, Phase 1: Final Report. California Partners for Advanced Transit and Highways (PATH).

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. ACM, 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. J. Mach. Learn. Res. 11, 3371–3408.

Wang, F.Y., 2010. Parallel control and management for intelligent transportation systems: concepts, architectures, and applications. IEEE Trans. Intell. Transp. Syst. 11, 630–638.

Wang, F.Y., Zhang, J.J., et al, 2016. Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond. IEEE/CAA J. Automat. Sinica 3, 113–120.

Wang, J., Zou, N., Chang, G.L., 2008. Travel time prediction: empirical analysis of missing data issues for advanced traveler information system applications. Transp. Res. Rec.: J. Transp. Res. Board, 81–91.

Zhang, Y., Liu, Y., 2009. Data imputation using least squares support vector machines in urban arterial streets. IEEE Signal Process. Lett. 16, 414–417.

Zhong, M., Sharma, S., 2006. Matching hourly, daily, and monthly traffic patterns to estimate missing volume data. Transp. Res. Rec.: J. Transp. Res. Board, 32–42.

Zhong, M., Sharma, S., Lingras, P., 2004. Genetically designed models for accurate imputation of missing traffic counts. Transp. Res. Rec.: J. Transp. Res. Board, 71–79.

Zhong, M., Sharma, S., Liu, Z., 2005. Assessing robustness of imputation models based on data from different jurisdictions: examples of alberta and saskatchewan, canada. Transp. Res. Rec.: J. Transp. Res. Board, 116–126.