Denoising autoencoders for fast real-time traffic estimation on urban road networks

Soham Ghosh[1], Muhammad Tayyab Asif[2], Laura Wynter[2]

***Abstract*— We propose a new method for traffic state estimation applicable to large urban road networks where a significant amount of the real-time and historical data is missing. Our proposed approach involves estimating the missing historical data through low-rank matrix completion, coupled with an online estimation approach for estimating the missing real-time data. In contrast to the traditional approach, the proposed method does not require re-calibration every time new streaming data becomes available. Empirical results from two metropolitan cities show that the proposed two-step approach provides comparable accuracy to a state of the art benchmark method while achieving two orders of magnitude improvement in computational speed.**

I. Introduction

Real-time traffic information is typically obtained from fixed sensors such as loop detectors, cameras and microwave signals on the roads as well as speed observations directly from the vehicles themselves. Although such probe vehicles can provide much higher spatial coverage than fixed sensors, the resulting data is in general lacking in temporal coverage. Most importantly, with probe vehicle-based traffic data, the set of links having traffic information varies over time. To address this, a number of methods for traffic state estimation have been proposed in which the models are re-calibrated as new data becomes available [1], [2]. Naturally, on-line calibration requires non-negligeable computational power for re-training and hence these methods may not be suitable for large-scale deployments.

While the set of links with probe traffic data is variable, in general, patterns tend to be dominated by a small number of highly repetitive trends [3]. In particular, traffic conditions across neighbouring roads tend to be highly correlated [4] as they do across city-scale networks [5], [6], [2], [7], [8]. In this study, we exploit such network-wide relationships to estimate missing traffic data in real-time.

Our contribution is the definition of a new method based on denoising autoencodes embedded in a two-phased approach. In the first phase, matrix completion is performed to recover missing traffic information from incomplete historical data. Then, a denoising autoencoder method is adapted to learn network-wide relationships from the imputed data for the estimation of missing real-time traffic data. Several variants of the de-noising model are proposed. Our method is compared against a state-of-the art benchmark method for online state estimation called Column based (CX) decomposition [9], [10] on real-world data from two metropolitan cities. Empirical evidence shows that the accuracy of our proposed method is comparable to that of the state-of-the art approach while offering a significant computational advantage, of roughly two orders of magnitude.

The rest of the paper is structured as follows. In section II, we provide a review of related work. Then, in section III, we present the methods, both the benchmark CX decomposition approach as well as our proposed method. In section IV, we discuss the computational complexity of various models for online estimation phase. In section V, we analyze empirically the performance of our method against the benchmark approach for a number of different degrees of missing data and across a number of other parameter variations. We conclude with suggestions for future work.

II. Related Work

Methods for road traffic estimation are typically divided into off-line imputation of historical data and online estimation. For off-line imputation, matrix/tensor completion methods have been shown to efficiently model dominant traffic patterns for recovering missing historical information [6], [11], [12], [13], [14]. The main advantage of multi-way array completion methods is that they can handle generic test networks with varying distributions of missing data [15], [16]. In this study, we consider three variants for our offline matrix imputation: iterative singular value decomposition (SVD) [16], matrix factorization [14], [17] and SoftImpute [15]. Iterative SVD is conceptually similar to the Bayesian principal component analysis formulations proposed for estimating missing values for incomplete traffic datasets [11], [12], [6].

The key challenge in performing real-time estimation is that the set of roads with missing data is likely to vary from one time interval to the next. Hence, estimation methods [18], [19] based on standard formulations of artificial neural networks (ANN) and support vector machines (SVM) are not suitable since the feature set is assumed to remain constant. For this reason, the state-of-the-art approach for such problems is based on models that can be re-calibrated in on-line manner [20], [1], [21], [22], [23], [24].

In this vein, Shan et al. [1] proposed a multiple linear regression model to exploit spatial-temporal correlations between links in the network; at each time step, a linear time heuristic algorithm was used to choose a subset of neighbouring node links, which along with available temporal data from the current link was used as a feature set to estimate

[1] Soham Ghosh is with Nanyang Technological University, Singapore (NTU). Email: soham002@e.ntu.edu.sg

[2] Muhammad Tayyab Asif and Laura Wynter are with IBM Research. Email: {mtasif, lwynter}@sg.ibm.com

Preprint submitted to 56th IEEE Conference on Decision and Control.
Received March 14, 2017.

the current speed. Only a single small network of 12 links was tested. Herring et al. [21] proposed a dynamic Bayesian network approach using traffic conditions of past time instances in the chosen link and its neighbours to estimate traffic conditions as a binary state variable (congested vs. not congested) along a small network of arterial roads. However, this approach is limited to classifying the traffic conditions into broad categories such as congested and not-congested.

Other model-based approaches try to estimate network-wide traffic conditions from origin destination data [22]. The estimation step is usually performed by re-calibrating a Kalman filter, thus limiting the scalability of such models. Mitrovic et. al. [2] proposed using CX decomposition and to recompute the relationship matrix using historical data at each time step. This method has been shown to be very accurate and is used as a benchmark for the method we propose in this work.

An autoencoder is an unsupervised feed-forward neural network that can learn lower dimensional representations $\mathbf{h}$ from interconnected variables, such as traffic conditions on neighbouring roads [25]. In this work, we consider a variation of these models termed as denoising autoencoders which can be applied to recover original input from its corrupted version. Deep architectures composed of these autoencoders (known as stacked denoising autoencoders)[26] have been successfully used in various applications such as image denoising [27], speech denoising [28], road traffic prediction [29] and in missing data imputation [30].

III. Methods

Let us denote the incomplete traffic states obtained by the sensors (GPS probes) at time t as $\mathbf{x}_d^t \in \mathbb{R}^N$, where N is the number of links in the network, and the set of roads with observed and missing data at time t as Θ^t and Φ^t, respectively. Our goal is to estimate the missing values $\{x_{d,i}^t\}_{i\in\Phi^t}$ and obtain traffic conditions $\hat{\mathbf{x}}^t$ for all links in the network. Thus we seek the traffic relations $\{\hat{x}_{d,i}^t\}_{i\in\Phi^t} = f(\{x_{d,j}^t\}_{j\in\Theta^t})$. The benchmarking method (CX decomposition) will re-learn this relationship function at every time instance t, whereas the de-noising autoencoder formulations will try to extract the relationship from historical data.

A. Benchmark method

CX decomposition can be used to obtain the relationship between different roads for different combinations of (Θ^t, Φ^t) in an online manner. The method works by re-computing the relationship function $\{\hat{x}_{d,i}^t\}_{i\in\Phi^t} = f^t(\{x_{d,j}^t\}_{j\in\Theta^t})$ at each time step as new data becomes available. We represent the relationship function as:

$$\mathbf{X} = \mathbf{C}^t\mathbf{F}^t, \tag{1}$$

where the matrix $\mathbf{X} \in \mathbb{R}^{\tau\times N}$ contains the historical traffic data, the matrix $\mathbf{C}^t \in \mathbb{R}^{\tau\times|\Theta^t|}$ contains the historical data from links $\{i\}_{i\in\Theta t}$ for which real-time data is available at time t and $\mathbf{F}^t$ represents the relationship matrix. The decomposition in (1), called column based (CX) decomposition, [9], provides a mechanism to estimate relationship function efficiently for streaming missing data as follows:

$$\mathbf{F}^t = [(\mathbf{C}^t)^T\mathbf{C}^t]^{-1}[\mathbf{C}^t]^T\mathbf{X}, \tag{2}$$

$$\hat{\mathbf{x}}^t = \mathbf{c}^t\mathbf{F}^t, \tag{3}$$

where $\mathbf{c}^t \in \mathbb{R}^{|\Theta^t|}$ contains available traffic data at time t and $[(\mathbf{C}^t)^T\mathbf{C}^t]^{-1}[\mathbf{C}^t]^T$ is the Moore Penrose pseudo-inverse of the matrix $\mathbf{C}^t$.

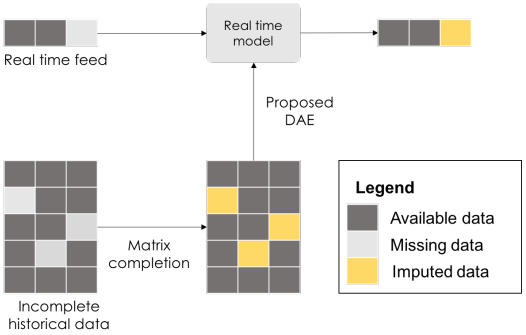


Fig. 1: Proposed method architecture.

B. Our proposed method

Figure 1 shows the components of our proposed method. The historical data is sent through an offline matrix completion process at the initial deployment of the system. Then, the real-time feed is estimated through the online model trained on the completed matrix. The real-time completed data is stored and used at subsequent time instants as the complete historical feed. In this section we describe the three offline matrix completion approaches used as well as several variants of the online autoencoder model.

Matrix completion methods for network data implicitly assume that the underlying network can be accurately modeled as a low-rank system. It can be shown that all the missing entries m in an incomplete matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ with rank r, where $r \ll n$ can be exactly recovered if $\mathcal{O}(nrlog(n))$ entries are observed i.e. missing entries $m \leq n^2 - \mathcal{O}(nrlog(n))$ [31]. The following methods were used:

1) Offline estimation of historical data:

a) Iterative SVD: Iterative SVD (or SVDImpute) was proposed in [16] to find missing values in DNA microarrays. The method works by first filling in all missing values in the incomplete speed matrix $\mathbf{X}$ by some initial value and then solving for the singular value decomposition (SVD) via expectation maximization (EM).

b) Matrix Factorization: Matrix factorization involves determining a decomposition of the traffic data matrix as $\mathbf{X} = \mathbf{UV}$, where $\mathbf{U} \in \mathbb{R}^{n\times r}$, $\mathbf{V} \in \mathbb{R}^{r\times m}$ and r represents the rank of the suitable low-dimensional representation. Here $\mathbf{U}$ and $\mathbf{V}$ can be solved by gradient descent with a loss function given by the distance between the observed and reconstructed entries and L_1, L_2 regularization penalties on their respective norms.

Preprint submitted to 56th IEEE Conference on Decision and Control.
Received March 14, 2017.

c) SoftImpute: SoftImpute seeks to obtain a suitable reconstruction $\hat{\mathbf{X}}$ with minimum rank r^*, of the incomplete traffic data $\mathbf{X}$ subject to a non-negative regularization $(x_{ij} - \hat{x}_{ij})^2 < \epsilon$ where ϵ is a non-negative hyperparameter that sets a tolerance for the training error:

$$\begin{aligned} &\min \ \ Rank(\hat{\mathbf{X}}) \\ &\text{s.t.}(x_{ij} - \hat{x}_{ij})^2 < \epsilon, \epsilon \geq 0, \forall (i,j) \in \Phi. \end{aligned} \tag{4}$$

The optimization problem defined in (4) is NP-hard. A convex relaxation is the minimization of the nuclear norm of $\hat{\mathbf{X}}$, represented as $\left\|\hat{\mathbf{X}}\right\|_*$, expressed as follows:

$$\begin{aligned} &\min \left\|\hat{\mathbf{X}}\right\|_* \\ &\text{s.t.}(x_{ij} - \hat{x}_{ij})^2 < \epsilon, \epsilon \geq 0, \forall (i,j) \in \Phi. \end{aligned} \tag{5}$$

The stopping criterion (Δ) is a threshold on the relative change in the Frobenius norm of $\hat{\mathbf{X}}$.

2) Online estimation for streaming data:

a) Denoising autoencoders: We consider network-wide estimation as a de-noising problem where missing field data is treated as observations corrupted by noise. We would like to obtain a generalized relationship function $\{\hat{x}^t_{d,i}\}_{i \in \Phi^t} = f(\{x^t_{d,j}\}_{j \in \Theta^t})$, which can work for most of the combinations of (Θ^t, Φ^t). The key computational advantage of using denoising autoencoders (DAE) is that a set of models (weights and biases) can be learnt offline by simulating different proportions of incomplete input $\mathbf{x}^t_d = \mathbf{x}^t \odot \boldsymbol{\delta}^t$ through applying a corruption mask $\boldsymbol{\delta}^t \in \{0,1\}^N$ to train the system and then comparing with ground truth data $\mathbf{x}^t \in \mathbb{R}^N$. The online estimation can thus be done quite fast as it only requires one forward pass through the trained network.

Consider first a simple single layer autoencoder. Suppose that $\mathbf{x} \in \mathbb{R}^N$ represents the traffic speeds along N links. The autoencoder maps the traffic data $\mathbf{x}$ to a low-dimensional representation as:

$$\mathbf{h} = \sigma(\mathbf{W}_1 \mathbf{x}_d + \mathbf{b}_1), \tag{6}$$

$$\hat{\mathbf{x}} = \sigma(\mathbf{W}_2 \mathbf{h} + \mathbf{b}_2), \tag{7}$$

where σ is a non-linear activation function (typically the logistic function) and $\mathbf{b}_1 \in \mathbb{R}^k$, $\mathbf{b}_2 \in \mathbb{R}^N$ are the bias vectors. The weights $\mathbf{W}_1, \mathbf{W}_2$ and bias terms can be learnt by minimizing the following loss function:

$$\underset{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2}{\operatorname{argmin}} \ \ \frac{1}{|\tau|} \sum_{t \in \tau} \left\| \sigma(\mathbf{W}_2[\sigma(\mathbf{W_1 x^t_d} + \mathbf{b_1})] + \mathbf{b}_2) - \mathbf{x}^t \right\|_2^2, \tag{8}$$

where $|\tau|$ is the size of training set. The weights are coupled by setting $\mathbf{W}_2 = \mathbf{W}_1^T$ [32].

To extract features in a more efficient manner, the model described above can be extended by adding hidden layers $\{\mathbf{h}_i\}_{i=1}^m$. However, such models tend to plagued by issues such as vanishing gradient and poor local solutions. To avoid these problems, we follow the procedure of training each layer separately [33]. For instance, hidden layer $\mathbf{h}_i$ will be trained after the unit below it has finished training, taking the hidden layer of $\mathbf{h}_{i-1}$ as an input. The hidden layers $\mathbf{h}_i$ concatenate to form $\mathbf{h}_c$ (12), which is later fed as an input

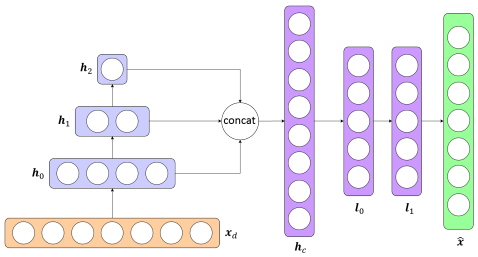


Fig. 2: Stacked Denoising Autoencoder

to an artificial neural network (ANN). Hence the stacked de-noising operation can be represented as follows (the bias terms have been omitted for brevity):

$$\mathbf{x}_d = \mathbf{x} \odot \boldsymbol{\delta} \tag{9}$$

$$\mathbf{h}_0 = \sigma(\mathbf{W}_0 \mathbf{x}_d) \tag{10}$$

$$\mathbf{h}_k = \sigma(\mathbf{W}_n \mathbf{h}_{k-1}) \text{ where } 1 \leq k < n \tag{11}$$

$$\mathbf{h}_c = \mathbf{h}_0 \| \mathbf{h}_1 \| \ldots \| \mathbf{h}_k \tag{12}$$

The ANN is trained to minimise reconstruction loss as described in (8). The hidden layers in the ANN are labelled as $\mathbf{l}_i$. The architecture used for the stacked denoising autoencoder (SDAE) (see Fig. 2) is different from the architecture in [26] which unrolled the encoder part of the network to form the decoder. The neural network estimation operation works as follows (omitting the bias terms):

$$\mathbf{l}_0 = \sigma(\mathbf{W}_0^{\text{ANN}} \mathbf{h}_c) \tag{13}$$

$$\mathbf{l}_p = \sigma(\mathbf{W}_p^{\text{ANN}} \mathbf{l}_{p-1}) \text{ where } 1 \leq p < m \tag{14}$$

$$\hat{\mathbf{x}} = \sigma(\mathbf{W}_m^{\text{ANN}} \mathbf{l}_{m-1}). \tag{15}$$

This formulation offers the advantage of constructing gradually higher levels of representation, thereby extracting features that are not locally constrained.

b) Temporal Denoising Autoencoder: In the previous model (SDAE), we considered current available traffic data $\mathbf{x}^t_d$ to estimate missing traffic information across the network at time t, thereby modeling the spatial relationships between different roads. Here we extend the model to include temporal lags. This temporal denoising autoencoder (TDAE) captures relationships between the current $\mathbf{x}^{(t)}$ ($h^{(t)}$) and past network conditions $\mathbf{x}^{(t-k)}$ ($\mathbf{h}^{(t-k)}$) to form h^t_c. The final reconstruction is done in the last layer from h_f. This model is expressed as follows:

$$\mathbf{x}^{t-k}_d = \mathbf{x}^{t-k} \circ \delta^{t-k} \text{ where } k \geq 0 \tag{16}$$

$$\mathbf{h}^{t-k} = \sigma(\mathbf{W} \mathbf{x}^{t-k}_d) \tag{17}$$

$$\mathbf{h_c} = \mathbf{h}^t \| \mathbf{h}^{t-1} \| \ldots \| \mathbf{h}^{t-k} \tag{18}$$

$$\mathbf{h_f} = \sigma(\mathbf{W}_{hh} \mathbf{h}_c) \tag{19}$$

$$\hat{\mathbf{x}} = \sigma(\mathbf{W}_{hf} \mathbf{h}_f). \tag{20}$$

In our experiments $k \leq 2$ is considered. First a DAE is trained on the $\mathbf{X}_{\text{train}}$. The learned weights parameter $\mathbf{W}$ is taken from the pre-trained DAE (17) and $\mathbf{W}_{hf}$ is initially set to be $\mathbf{W}^T$, which is the decoder part of the original DAE.

Preprint submitted to 56th IEEE Conference on Decision and Control.
Received March 14, 2017.

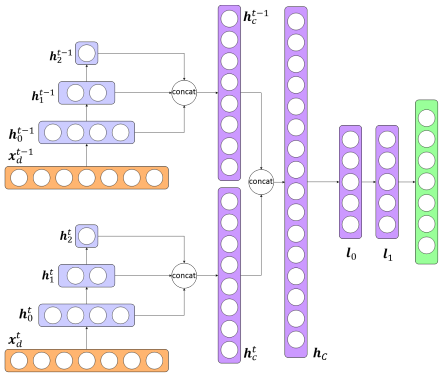


Fig. 3: Temporal Stacked Denoising Autoencoder.

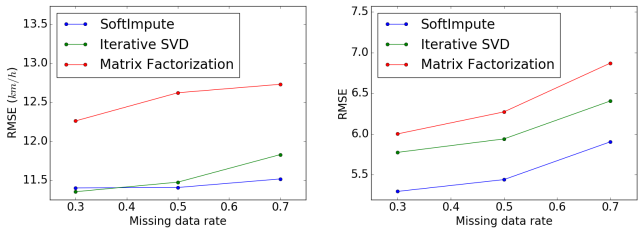
c) Temporal Stacked Denoising Autoencoder: A natural extension of the previous models is to define a Temporal Stacked Denoising Autoencoder (TSDAE); in this model, three SDAE architecures are used for each of the time instances t, $t-1$ and $t-2$. The combined hidden representations from each time instance $\mathbf{h}_c^t$, $\mathbf{h}_c^{t-1}$ and $\mathbf{h}_c^{t-2}$ are concatenated to give $\mathbf{h}_c$. Similar to the SDAE, an ANN is then trained to reconstruct $\mathbf{x}$ from $\mathbf{h}_c$. An example structure with one temporal lag is shown in Figure 3.

IV. Computational Complexity

In this section we compare the computational complexity of the benchmark algorithm with our proposed method.

The basic DAE unit consists of a forward pass which involves two matrix/vector multiplication: $\mathbf{W}$ and $\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{W} \in \mathbb{R}^{n \times h}$, Here, n is the number of links, and h is the number of hidden units. Hence the time complexity is $\mathcal{O}(nh)$ for a basic DAE unit. The TDAE involves the concatenation operation which we assume to be linear in the sizes of the vectors being concatenated. Assume that the TDAE (and the TSDAE) use t time steps. Hence the time complexity for TDAE is $\mathcal{O}(tnh + th)$. In our experiments, we use $t = 2$. The SDAE requires a forward pass through multiple levels which is $\mathcal{O}(nh_0 + \sum_{i=1}^{p-1} h_i h_{i+1})$, a final concatenation operation which is linear in the sum of the hidden layers $\mathcal{O}(\sum_{i=1}^{p} h_i)$, and another forward pass of the concatenated vector which takes $\mathcal{O}((\sum_{i=1}^{p} h_i)l_0 + \sum_{j=1}^{q-1} l_j l_{j+1} + l_q n)$ (where l_j is the size of the j^{th} layer in the ANN). Assume that there are p hidden layers in the SDAE, and q hidden layers in the ANN. Therefore, the overall complexity of the SDAE is $\mathcal{O}(nh_0 + \sum_{i=1}^{p-1} h_i h_{i+1} + \sum_{i=1}^{p} h_i + (\sum_{i=1}^{p} h_i)l_0 + \sum_{j=1}^{q-1} l_j l_{j+1} + l_q n)$. Similarly, the TSDAE has a complexity of $\mathcal{O}(tnh_0 + t\sum_{i=1}^{p-1} h_i h_{i+1} + t\sum_{i=1}^{p} h_i + t(\sum_{i=1}^{p} h_i)l_0 + \sum_{j=1}^{q-1} l_j l_{j+1} + l_q n)$.

Note that the complexity of our method scales linearly with the size of sub-network, while CX-Decomposition requires the computation of pseudo-inverse at every time instance to learn the relationship matrix between observed traffic conditions at $c < n$ links and the rest of the network (see (3)). This requires $\mathcal{O}(n^3)$ and $\mathcal{O}(nc)$ (or $\mathcal{O}(n^2/\rho)$, where ρ is the compression factor) computations, respectively.



(a) City 1. (b) City 2.

Fig. 4: RMSE values for offline imputation using matrix completion models.

TABLE I: RMSE of real-time estimation when models were trained on complete historical data (No imputation was performed). CL refers to the ratio of missing traffic information in real-time data only.

CL	0.3		0.5		0.7	
	City 1	City 2	City 1	City 2	City 1	City 2
CX	11.11	7.22	11.05	7.01	11.08	6.79
SDAE	11.29	5.99	11.30	6.01	11.44	6.04
TDAE	11.30	5.76	11.39	5.80	11.57	5.93
TSDAE	11.20	5.72	11.22	5.78	11.31	5.90

V. Numerical Experiments

For analysis, we considered data from two metropolitan cities. City 1 data consists of three weeks of traffic speed information from the downtown area of the city in 2015. The test network consisted of expressway sections as well as major arterial roads carrying traffic towards and from city centre and included $p = 1152$ road segments. We used 2 weeks of speed data for training and 1 week for testing. While more training data could be used, 2 weeks was found to be sufficient to achieve good accuracy. The aggregation interval for this data set is 5 minutes.

The second dataset consists of occupancy data from major arterial roads in the city center of City 2. The occupancy data has an aggregation interval of 6 minutes, and the test network was composed of 600 links. We used again 2 weeks of data from 2015 for training and 1 week of data for testing.

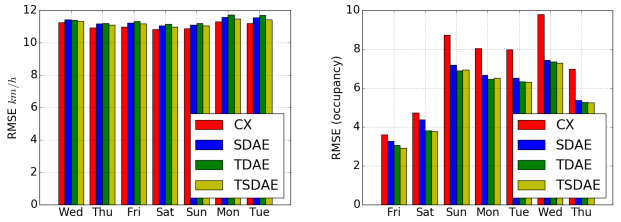
Initially we examine the accuracy of the proposed model when the training corruption level (missing data) is the same as that of the test set. The performance is compared in terms of root mean squared error (RMSE) given by $[\sum_{i=1}^{p}\sum_{t=1}^{n} c_i^t (x_i^t - \hat{x}_i^t)^2 / \sum_{i=1}^{p}\sum_{t=1}^{n} c_i^t]^{1/2}$, where $c_i^t = 1$ if the traffic information is missing for i^{th} link at time t and $c_i^t = 0$ otherwise. The missing data ratio/corruption level (CL) is defined as $\Pr(c_i^t = 1)$.

For each city, imputation of historical data was performed by applying the method which achieved the highest accuracy. The autoencoder architectures were implemented in `theano`[34]. Figure 4 illustrates the error of the offline matrix completion methods. In general, the three offline methods provide similar performance. Traffic on arterial roads with traffic signals and traffic occupancy data tends to be more volatile than speed data and expressway traffic.

Preprint submitted to 56th IEEE Conference on Decision and Control.
Received March 14, 2017.

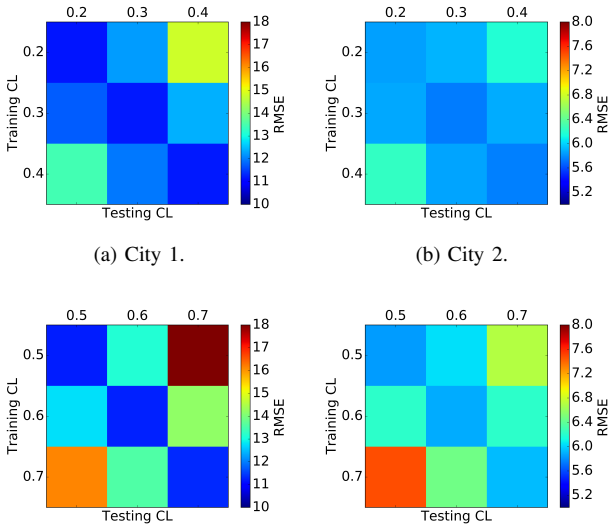
TABLE II: RMSE of real-time estimation when models were trained on imputed historical data. CL refers here to the ratio of missing traffic information in both the real-time and the historical data.

CL	0.3		0.5		0.7	
	City 1	City 2	City 1	City 2	City 1	City 2
CX	11.25	7.30	11.35	7.41	11.59	7.82
SDAE	11.24	5.96	11.35	5.99	11.53	6.06
TDAE	11.26	5.81	11.42	5.88	11.61	6.02
TSDAE	11.17	5.77	11.27	5.84	11.38	5.96



(a) City 1. (b) City 2.

Fig. 5: Estimation RMSE by day of the week with missing data 50%.



(a) City 1. (b) City 2.

(c) City 1. (d) City 2.

Fig. 6: RMSE values over multiple missing data ratios training vs. testing.

As such, SoftImpute which uses nuclear norm regularization performs somewhat better on the City 2 dataset.

For each city, we conducted two sets of experiments: In the first case, we trained the estimation models (CX, de-noising autoencoders) with complete historical data (see Table I). In the second case, historical data is also missing so we imputed the data using the best offline method for each and then trained the real-time models on the imputed data (see Table II). Figure 5 shows the estimation error of the real-time methods for different days of the week. In the City 1 dataset all methods provide roughly equivalent accuracy, whereas on the occupancy data from the urban network of City 2 the proposed methods achieve lower error.

Next, we allow the corruption level of the online data to vary with respect to the offline calibrated models, as would be the case in an online implementation of the proposed method. We calibrate a set of 6 models trained on corruption levels 0.2, 0.3, 0.4, 0.5, 0.6 and 0.7. For testing, we test datasets in the range [0.2,0.4] on all three models calibrated in that range, and similarly for the range [0.5,0.7]. Recall that the benchmark CX method is recalibrated online and is hence optimal with respect to the online corruption level. Figure 6 provides the RMSE, in this case using the TSDAE formulation, when there is missing data rate mismatch between the training and test set. A perfect match between training and test missing data rate gives the boxes on the diagonal of the matrix, having the lowest RMSE.

To assess the computational advantage of the proposed method, a set of 10 synthetic networks was generated by replicating up to 104 copies of the network and data of City 1 from the original 1152-link network for network sizes of up to 120,000 links. The experiments were run on a Macbook Pro 2014 (2.6 GHz Intel Core i5, 8GB RAM). Figure 7

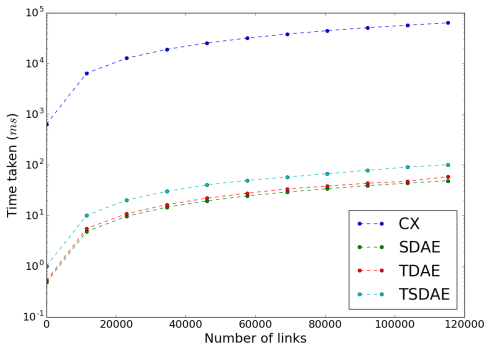


Fig. 7: Computation time of real-time traffic estimation (ms) for synthetic graphs of increasing size.

shows how computation time scales for each method with the size of the network. The main observation is that while all methods scale similarly, the proposed methods based on the de-noising autoencoders achieve around 2 orders of magnitude improvement over the benchmark CX decomposition method. Hence the accuracy of the proposed method is comparable to that of the state-of-the-art benchmark CX decomposition while affording a significant computation advantage.

VI. Conclusion

We proposed a two-step offline-online procedure based on de-noising autoencoders to estimate network-wide traffic conditions from limited real-time data and incomplete historical data. The first step is required only at initialization of the method, whereby a matrix completion is performed to fill in missing historical data. A set of trained models generated with varying levels of missing data in the offline

Preprint submitted to 56th IEEE Conference on Decision and Control.
Received March 14, 2017.

phase is subsequently used online, where the appropriate model is selected based on the missing data rate in the real-time sample.

We compare the proposed approach against the state-of-the-art online-calibrated CX decomposition method. Our results show that the proposed approach achieves comparable accuracy to CX decomposition while offering a two-order-of-magnitude improvement in computational time, due to the fact that the calibration is performed offline.

A main source of difficulty in traffic state estimation and prediction is handling non-stationary or incident conditions on the road. Hence a valuable extension of this work would involve incorporating information about unplanned events such as accidents or extreme weather conditions.

REFERENCES

[1] Z. Shan, Y. Xia, P. Hou, and J. He, "Fusing incomplete multisensor heterogeneous data to estimate urban traffic," *IEEE MultiMedia*, vol. 23, no. 3, pp. 56–63, July 2016.

[2] N. Mitrovic, A. Narayanan, M. T. Asif, A. Rauf, J. Dauwels, and P. Jaillet, "On centralized and decentralized architectures for traffic applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1988–1997, July 2016.

[3] S. Jiang, J. Ferreira, and M. C. González, "Clustering daily patterns of human activities in the city," *Data Mining and Knowledge Discovery*, vol. 25, no. 3, pp. 478–510, 2012.

[4] W. Min and L. Wynter, "Real-time road traffic prediction with spatio-temporal correlations," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 4, pp. 606–616, 2011.

[5] Y. Kamarianakis, W. Shen, and L. Wynter, "Real-time road traffic forecasting using regime-switching space-time models and adaptive lasso," *Applied stochastic models in business and industry*, vol. 28, no. 4, pp. 297–315, 2012.

[6] M. T. Asif, N. Mitrovic, J. Dauwels, and P. Jaillet, "Matrix and tensor based methods for missing data estimation in large traffic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1816–1825, July 2016.

[7] M. T. Asif, K. Srinivasan, N. Mitrovic, J. Dauwels, and P. Jaillet, "Near-lossless compression for large traffic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1817–1826, Aug 2015.

[8] A. Hofleitner, R. Herring, A. Bayen, Y. Han, F. Moutarde, and A. De La Fortelle, "Large scale estimation of arterial traffic and structural analysis of traffic patterns using probe vehicles," in *Transportation Research Board 91st Annual Meeting (TRB'2012)*, 2012.

[9] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Relative-error cur matrix decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 844–881, 2008.

[10] N. Mitrovic, M. T. Asif, J. Dauwels, and P. Jaillet, "Low-dimensional models for compressed sensing and prediction of large-scale traffic data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2949–2954, 2015.

[11] L. Qu, L. Li, Y. Zhang, and J. Hu, "Ppca-based missing data imputation for traffic flow volume: a systematical approach," *IEEE Transactions on intelligent transportation systems*, vol. 10, no. 3, pp. 512–522, 2009.

[12] M. T. Asif, N. Mitrovic, L. Garg, J. Dauwels, and P. Jaillet, "Low-dimensional models for missing data imputation in road networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3527–3531.

[13] L. Li, Y. Li, and Z. Li, "Efficient missing data imputing for traffic flow by considering temporal and spatial dependence," *Transportation research part C: emerging technologies*, vol. 34, pp. 108–120, 2013.

[14] Z. Li, Y. Zhu, H. Zhu, and M. Li, "Compressive sensing approach to urban traffic sensing," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. IEEE, 2011, pp. 889–898.

[15] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *Journal of machine learning research*, vol. 11, no. Aug, pp. 2287–2322, 2010.

[16] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, Jun 2001.

[17] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang, "A compressive sensing approach to urban traffic estimation with probe vehicles," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2289–2302, 2013.

[18] Y. Zhang and Y. Liu, "Missing traffic flow data prediction using least squares support vector machines in urban arterial streets," in *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*. IEEE, 2009, pp. 76–83.

[19] C. Chen, J. Kwon, J. Rice, A. Skabardonis, and P. Varaiya, "Detecting errors and imputing missing data for single-loop surveillance systems," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1855, pp. 160–167, 2003.

[20] L. Li, X. Chen, and L. Zhang, "Multimodel ensemble for freeway traffic state estimations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1323–1336, 2014.

[21] R. Herring, A. Hofleitner, P. Abbeel, and A. Bayen, "Estimating arterial traffic conditions using sparse probe data," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, Sept 2010, pp. 929–936.

[22] S. Gupta, R. Seshadri, B. Atasoy, F. C. Pereira, S. Wang, V.-A. Vu, G. Tan, W. Dong, Y. Lu, C. Antoniou *et al.*, "Real time optimization of network control strategies in dynamit2. 0," in *Transportation Research Board 95th Annual Meeting*, no. 16-5560, 2016.

[23] C. Antoniou, M. Ben-Akiva, and H. N. Koutsopoulos, "Nonlinear kalman filtering algorithms for on-line calibration of dynamic traffic assignment models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 4, pp. 661–670, 2007.

[24] D. B. Work, O.-P. Tossavainen, S. Blandin, A. M. Bayen, T. Iwuchukwu, and K. Tracton, "An ensemble kalman filtering approach to highway traffic estimation using gps enabled mobile devices," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 5062–5068.

[25] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, jul 2006. [Online]. Available: http://dx.doi.org/10.1126/science.1127647

[26] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. antoine Manzagol, "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," 2010.

[27] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 341–349.

[28] X. Feng, Y. Zhang, and J. Glass, "Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1759–1763.

[29] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[30] F. Strub and J. Mary, "Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs," in *NIPS Workshop on Machine Learning for eCommerce*, Montreal, Canada, Dec. 2015. [Online]. Available: https://hal.inria.fr/hal-01256422

[31] E. J. Candès and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[32] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[33] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 153–160. [Online]. Available: http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf

[34] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688