Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning

Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie

***Abstract*—Traffic flow prediction is a fundamental problem in transportation modeling and management. Many existing approaches fail to provide favorable results due to being: 1) shallow in architecture; 2) hand engineered in features; and 3) separate in learning. In this paper we propose a deep architecture that consists of two parts, i.e., a deep belief network (DBN) at the bottom and a multitask regression layer at the top. A DBN is employed here for unsupervised feature learning. It can learn effective features for traffic flow prediction in an unsupervised fashion, which has been examined and found to be effective for many areas such as image and audio classification. To the best of our knowledge, this is the first paper that applies the deep learning approach to transportation research. To incorporate multitask learning (MTL) in our deep architecture, a multitask regression layer is used above the DBN for supervised prediction. We further investigate homogeneous MTL and heterogeneous MTL for traffic flow prediction. To take full advantage of weight sharing in our deep architecture, we propose a grouping method based on the weights in the top layer to make MTL more effective. Experiments on transportation data sets show good performance of our deep architecture. Abundant experiments show that our approach achieved close to 5% improvements over the state of the art. It is also presented that MTL can improve the generalization performance of shared tasks. These positive results demonstrate that deep learning and MTL are promising in transportation research.**

***Index Terms*—Deep learning, multitask learning (MTL), task grouping, traffic flow prediction.**

I. Introduction

TRAFFIC flow prediction aims at estimating the number of vehicles given a specific region and a time interval, which is an important problem to address in transportation management [21], [34], [36]. Reliable, accurate, and consistent real-time traffic flow prediction should support: 1) real-time route guidance in advanced traveler information systems for saving time and money; 2) reliable traffic control strategies in advanced traffic management systems for reducing traffic congestion and accidents; and 3) the evaluation of these dynamic guidance and control strategies [37]. Much research has been focused on this subject in recent years. Existing traffic flow prediction approaches can be divided into three categories.

1) Time-series approaches [8], [18], [31], [32]. These approaches, such as the autoregressive integrated moving average (ARIMA) model [32] that is extended from the autoregressive–moving average (ARMA) model [18] with an extra procedure of difference, focus on finding the patterns of the temporal variation of traffic flow and then use that information for prediction
2) Probabilistic graph approaches [26], [28], [35]. The modeling and forecasting of traffic flow is done from a probabilistic graph perspective, such as a Bayesian network, a Markov chain, and Markov random fields (MRFs).
3) Nonparametric approaches [2], [4], [23], [24]. Researchers demonstrated that nonparametric approaches generally perform better due to their strong ability to capture the uncertainty and complex nonlinearity of a traffic time series. Some representative methods are artificial neural networks (NNs) [3], [24], support vector regression (SVR) [2], and local weighted learning (LWL) [23].

Most systems possess three failings. First, most are shallow in architecture. For NN approaches, most of the architectures are designed with one single hidden layer [3], [12], [37]. One of the reasons may be the unsuccessful training strategy for an NN with multiple hidden layers [11]. For other methods such as time-series approaches, a linear architecture is often preferred. Second, tedious and error-prone hand-engineered features are needed for some approaches. They require prior knowledge of specific domains for feature extraction and selection. Finally, they usually predict the traffic flow of each road separately. Take the ARIMA model as an example. It predicts the future traffic flow of a road only based on previous traffic flows observed from the road. It neglects sharing knowledge among related roads. As we know, a transportation system is a highly correlated network [29]. It is better to do traffic flow prediction from a network perspective to use the shared knowledge from related parts of the network.

In this paper, we attempt to define a deep architecture for traffic flow prediction that learns features with limited prior knowledge. This is achieved by training a deep belief network (DBN) according to the work of Hinton *et al.* [10], [11]. The key idea is to use greedy layerwise training with stacked

Manuscript received August 28, 2013; revised December 19, 2013 and March 3, 2014; accepted March 6, 2014. Date of publication April 10, 2014; date of current version September 26, 2014. This work was supported in part by the National High Technology Research and Development Program of China under Grant 2014AA015103 and in part by the National Science and Technology Support Plan under Grant 2014BAG01B02. The Associate Editor for this paper was J. Zhang. *(Corresponding author: Guojie Song.)*

W. Huang and H. Hong are with the School of Electrical Engineering and Computer Science, Peking University, Beijing 100871, China.

G. Song is with the School of Electrical Engineering and Computer Science, Peking University, Beijing 100871, China, and also with the Research Center of Intelligent Information Processing, Peking University, Beijing 100871, China (e-mail: gjsong@pku.edu.cn).

K. Xie is with the School of Electrical Engineering and Computer Science, Peking University, Beijing 100871, China, and also with the Research Center of Intelligent Traffic System, Peking University, Beijing 100871, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2014.2311123

restricted Boltzmann machines (RBMs), followed by fine-tuning. The DBN is used for **unsupervised feature learning** in traffic flow prediction. The DBN has been found to be very effective in learning the representative features from the data in an unsupervised fashion [13], [15]. Large data industries, such as Google and Microsoft, are also employing deep learning in their prediction tasks [6], [7]. For a transportation system, such as a typical complex system, the DBN can help us in learning and capturing effective features without prior knowledge. Upon them, a regression layer can be added for supervised training. The whole structure can be seen as a **pattern learning** scheme. Moreover, to bring the idea of multitask learning (MTL) [1], [5] into our study, we integrate several tasks together and jointly train the model in the top level of our deep architecture, instead of training each task separately. MTL can take full advantage of weight sharing in the DBN to provide better prediction results. Furthermore, we demonstrate that both homogeneous and heterogeneous tasks can improve the overall performance. To make multitask regression more effective, an adaptive task grouping method is proposed, instead of training all tasks jointly. The contribution of this paper could be summarized in three aspects.

1) To the best of our knowledge, this paper is the first attempt to introduce deep learning approaches into transportation research. The characteristics of transportation systems, such as the large amounts of data and the high dimensions of features, would make deep learning a promising method for transportation research.
2) Learning several related tasks together can provide better results in a transportation system. We proposed a deep architecture that integrates with MTL very well. It is demonstrated that MTL, which predicts traffic flow jointly, outperforms building predicting models separately. We investigated the effect of both homogeneous and heterogeneous tasks to validate the usefulness of MTL.
3) We proposed a grouping method based on the weights of the top layer in our deep architecture to group related tasks for better overall performance. Weight sharing of a deep network is considered in grouping.

The rest of this paper is structured as follows. Section II presents the background knowledge of traffic flow prediction and a DBN. In Section III we introduce our deep architecture for traffic flow prediction and MTL schemes. Section IV gives the experimental results of our approach and the analysis of these results. Finally, the conclusion and future works are described in Section V.

II. Background

A. Traffic Flow Prediction

Traffic flow prediction has been long regarded as a critical problem for intelligent transportation systems. It aims at estimating the number of vehicles in a road or a station in several time intervals into the future. Time intervals are usually defined as short-term intervals varying from 5 to 30 min. For operational analysis, the Highway Capacity Manual [16] suggests using a 15-min time interval.

Two types of data are usually used in traffic flow prediction. The first type of data is collected by sensors on each road, such as inductive loops. The task is to predict the traffic flow on each road or segment. The other type of data is collected at the entrance and exit of a road segment. For example, we would get a card from the toll station entering a highway and have to turn it back when we leave the highway. This kind of data is referred to as the entrance–exit station data. Despite predicting the traffic flow on each road, another task is to forecast the traffic flow in each station, particularly the exit station. In addition to these two common types of data, a variety of data sources, such as lidar, radar, and video from surveillance cameras [19], have emerged in traffic flow prediction research. In this paper, the data from the California Freeway Performance Measurement System (PeMS)[1] are used as the first type of data, and the data from a highway system in China are used as the second type of data.

The traffic flow of the ith observation point (spatial identification, road, or station) at the tth time interval is denoted by $f_{i,t}$. At time T, the task is to predict traffic flow $f_{i,T+1}$ at time $T+1$ based on the traffic flow sequence $F = \{f_{i,t} | i \in O,\ t = 1, 2, \ldots, T\}$ in the past, where O is the full set of observation points (roads and stations). Some works may focus on predicting the traffic flow of the next several time intervals from $T+1$ to $T+n$ as well.

Traffic flow prediction consists of two steps, i.e., feature learning and model learning. Feature learning learns a feature representation model g, which extracts and selects the most representative features from the traffic flow sequence F of all stations in the past. After feature learning, the traffic flow sequence is transformed into feature space $g(F) \rightarrow X$. Prediction task $f_{i,T+1}$ can be represented as Y. In some approaches, feature learning is usually hand-engineered. Some important factors for transportation, such as speed, the volume of vehicles, and density, are calculated from raw data and used as features for prediction. Moreover, in most approaches, only time-series features are employed for prediction. For example, the ARIMA model only selects the previous traffic flow of a specific point j as

$$\forall y_j = f_{j,T+1}, \quad x_j = \{f_{j,t} | t = T, T-1, \ldots, T-m+1\} \quad (1)$$

where m is the time step in the ARIMA model. It does not use any extra data, except the history traffic flows for task j itself. Feature learning generates appropriate features as the input for the prediction model. Model learning is a supervised learning problem. Given the feature X and task Y pairs obtained from history traffic flow $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$, it learns the best parameters for predicting model $\widehat{Y} = h(X)$ that minimizes the loss function, i.e.,

$$L(Y, \widehat{Y}) = \frac{1}{2}(Y - \widehat{Y})^2. \quad (2)$$

Although the specific form of predicting models may be different, the objective function is always similar.

[1] http://pems.dot.ca.gov/

Another preliminary is the measurements for traffic flow prediction. Two most commonly used measurements are the mean absolute percentage error (MAPE) and the root-mean-square error [37]. MAPE is used as the basic measurement in our study, as reported in Section IV. It is computed as

$$\text{MAPE}(y,\, y') = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - y_i'|}{y_i}. \tag{3}$$

B. DBN

Recent work on deep learning has demonstrated that deep sigmoidal networks can be trained layerwise to produce good results for many tasks such as image and audio classification [10], [13], [14]. The idea of deep learning is, first, using large amounts of unlabeled data to learn a feature presentation by pretraining a multilayer NN in an unsupervised way and, then, using labeled data for supervised fine-tuning to slightly adjust the learned features for better prediction.

The DBN is the most common and effective approach among all deep learning models. It is a stack of RBMs, each having only one hidden layer. The learned units' activations of one RBM are used as the "data" for the next RBM in the stack. Hinton *et al.* proposed a way to perform fast greedy learning of a DBN, which learns one layer at a time [10].

An RBM is a particular type of MRFs. It is an undirected graphical model in which visible variables $\boldsymbol{v}$ are connected to stochastic hidden units $\boldsymbol{h}$ using undirected weighted connections [30]. They are restricted that there are no connections within hidden variables or visible variables. The model defines a probability distribution over $\boldsymbol{v}$, $\boldsymbol{h}(P(\boldsymbol{v},\, \boldsymbol{h}))$ via an energy function $(E(\boldsymbol{v},\, \boldsymbol{h};\theta))$ [30]. Supposing it is a binary RBM, it can be written as

$$\begin{aligned}-\log P(\boldsymbol{v},\, \boldsymbol{h}) &\propto E(\boldsymbol{v},\, \boldsymbol{h};\theta)\\ &= -\sum_{i=1}^{|V|}\sum_{j=1}^{|H|} w_{ij}v_i h_j - \sum_{i=1}^{|V|} b_i v_i - \sum_{j=1}^{|H|} a_j h_j \end{aligned} \tag{4}$$

where $\theta = (\boldsymbol{w},\, \boldsymbol{b},\, \boldsymbol{a})$ is the parameter set, w_{ij} is the symmetric weight between visible unit i and hidden unit j, and b_i and a_j are their bias. The number of visible and hidden units is represented as $|V|$ and $|H|$, respectively. This configuration makes it easy to compute the conditional probability distributions, when $\boldsymbol{v}$ or $\boldsymbol{h}$ is fixed, as

$$\begin{aligned} p\left(h_j|\boldsymbol{v};\theta\right) &= \text{sigm}\left(\sum_{i=1}^{|V|} w_{ij}v_i + a_j\right)\\ p\left(v_i|\boldsymbol{h};\theta\right) &= \text{sigm}\left(\sum_{j=1}^{|H|} w_{ij}h_j + b_i\right)\end{aligned} \tag{5}$$

where $\text{sigm}(x) = (1/(1+e^{-x}))$ is a sigmoid function. The parameters of the model $\theta = (\boldsymbol{w},\, \boldsymbol{b},\, \boldsymbol{a})$ can be learned using contrastive divergence [9] effectively.

Then, we can stack several RBMs together into a DBN. The key idea behind training a DBN by training a series of RBMs is that the parameters θ learned by an RBM define both $p(\boldsymbol{v}|\boldsymbol{h},\theta)$ and prior distribution $p(\boldsymbol{h}|\theta)$ [17]. Therefore, the probability of generating visible variables can be written as

$$p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p\left(\boldsymbol{h}|\theta\right)p(v|h,\, \theta)\,. \tag{6}$$

After θ is learned from an RBM, $p(\boldsymbol{v}|\boldsymbol{h},\, \theta)$ is kept. In addition, $p(\boldsymbol{h}|\theta)$ can be replaced by a consecutive RBM, which treats the hidden layer of the previous RBM as visible data. This way, it can improve a variational lower bound on the probability of the training data, as introduced in [10]. The DBN can be used as an unsupervised feature learning method if no labels are provided.

III. Learning Architecture

In this section, we first propose a deep learning architecture (DLA) for traffic flow prediction. Then, we describe how to incorporate MTL into our learning architecture.

A. DLA

Many previous approaches of traffic flow prediction are shallow in architecture. Instead, we advocate a deep architecture in this paper. For transportation systems such as a complex system, one single hidden layer usually would not be enough in describing the complicated relations between inputs and outputs. A deep architecture can show its advantage in dealing with these complicated relations. In addition, the deep architecture could learn features with as less prior knowledge as possible.

Here, we employ a DBN for unsupervised feature learning and add a regression layer above the DBN for traffic flow prediction. Training a deep multilayered NN is generally difficult because the error gradient would explode or vanish when the number of layers increases. Recent work on deep learning has made training deep NNs more effective since Hinton's breakthrough in 2006 [10]. Then, we use a sigmoid regression at the top layer in our approach so that we can perform supervised fine-tuning on the whole architecture easily. The sigmoid regression layer can be also replaced with other regression models such as SVR.

Our deep architecture for traffic flow prediction on a single road or station is summarized in Fig. 1(a). Input space X is, generally, the raw data we collected. To perform prediction from a network perspective, we let all the observation points be in the input space. Moreover, we use the traffic flow of several previous time intervals. Therefore, the input space is large ($|O| \times k$, where $|O|$ is the number of observation points, and k is the number of time intervals). The number of previous time intervals in the inputs k is the only prior knowledge we need to build the predicting model. We tested several values of k and chose the best value through cross validation ($k = 4$ here). Unlike other NN-based approaches, we did not apply any artificial feature extraction and selection from the raw data. The only preprocessing work is normalizing the traffic flow into [0, 1]. Input vector X can be represented as $X = \{x_i^t | t \in T,\, i \in O\}$, where x_i^t is the normalized number of vehicles in observation point i in time interval t.

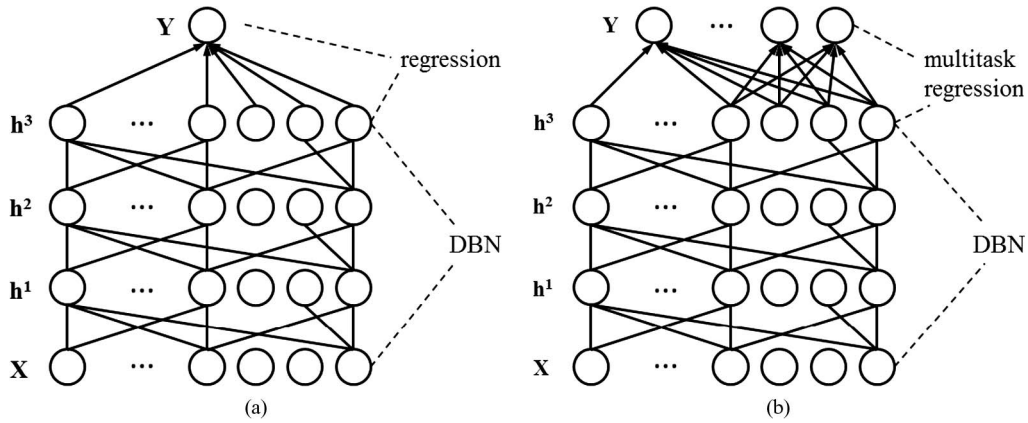


Fig. 1. Deep architecture for traffic flow prediction. (a) DBN at the bottom for unsupervised feature learning with a sigmoid regression layer at the top for supervised prediction. It is an architecture for traffic flow prediction of a single road. (b) All tasks are jointly trained via multitask regression.

Unlike the binary RBM, as introduced in Section II-B, we replace it with real-valued units [22] that have Gaussian noise to model the traffic flow data. Energy function and conditional probability distributions are given as

$$-\log P(\boldsymbol{v},\boldsymbol{h}) \propto E(\boldsymbol{v},\boldsymbol{h};\theta) = \sum_{i=1}^{|V|}\frac{(v_i-b_i)^2}{2\sigma_i^2} - \sum_{j=1}^{|H|} a_j h_j - \sum_{i=1}^{|V|}\sum_{j=1}^{|H|}\frac{v_i}{\sigma_i} h_j w_{ij} \tag{7}$$

$$p\left(h_j|\boldsymbol{v};\theta\right) = \text{sigm}\left(\sum_{i=1}^{|V|} w_{ij}v_i + a_j\right)$$
$$p\left(v_i|\boldsymbol{h};\theta\right) = N\left(b_i + \sigma_i\sum_{j=1}^{|H|} h_j w_{ij},\, \sigma_i^2\right) \tag{8}$$

where σ is the standard deviation vector of Gaussian visible units, and $N(\mu,\, \sigma^2)$ is the Gaussian distribution with mean μ and variance σ.

Since the traffic flows of all the observation points are used as input data, we have to regularize the model for sparsity [15]. We encourage each hidden unit to have a predetermined expected activation by a regularization penalty of the following form:

$$\lambda\sum_{j=1}^{|H|}\left(\rho - \frac{1}{m}\left(\sum_{k=1}^{m} E\left[h_j|\boldsymbol{v}^k\right]\right)\right)^2 \tag{9}$$

where ρ determines the sparsity, and $\boldsymbol{v}^k$ is a sample in the training set with m total samples.

We can treat the DBN on the bottom as a feature learning model. Each layer in the DBN is a process of nonlinear feature transformation. Features learned in the top layer of the DBN are the most representative features for modeling the data. It can be denoted by $H^p = \{h_1^p,\, h_2^p, \ldots, h_m^p\}$, where p represents the top layer, and m is the number of features in the top layer. The representative features are learned in an unsupervised way and can be used for various tasks, such as classification and regression. In our architecture, the most representative features H^p are used as the input vector for prediction (the top regression layer). Moreover, since we employ sigmoid regression in the regression layer, the whole structure can be seen as a complete structure of an NN. Features learned from the DBN can be fine-tuned via error backpropagation on the whole structure by using labeled data for better prediction. It is also feasible if we do not perform fine-tuning on the whole structure. Then, the DBN is the feature learning model, and the sigmoid regression layer is the prediction model.

In conclusion, we use an example to go through the whole process of the prediction model and demonstrate the advantage of our deep architecture. First, we use the raw data of all the observation points as the input so that only limited prior knowledge on the transportation system is required. Then, a DBN is applied in learning the representative and robust features from the inputs via several layers of nonlinear feature transformation to describe the complex mapping of inputs and features in the transportation system. Finally, a regression layer is used to produce the predicted results from the features learned by the DBN. The result can be more accurate since the features learned by the DBN are more representative.

B. MTL

MTL, which learns several tasks at the same time with the aim of a mutual benefit, is a paradigm in machine learning [1]. It can improve learning for one task by using the information contained in related tasks via learning tasks in parallel while using a shared representation. The information from one task can help the related tasks to be learned more effectively.

In a transportation system, all roads and entrance–exit stations are connected to each other. There is a lot of shared information among these roads and stations. Therefore, it is promising to incorporate MTL in our architecture. We consider the prediction of one road or station as a task. In a single-task learning approach, we can train the regression layer for each task separately [as shown in Fig. 1(a)]. We may have to build a deep architecture for each task if we want to employ fine-tuning on the whole structure. In our MTL architecture, we put a group of related tasks together in the top regression layer. These tasks are trained and later fine-tuned via backpropagation jointly. It is thus reasonable to expect better results for each task when learning other related tasks at the same time. Sharing

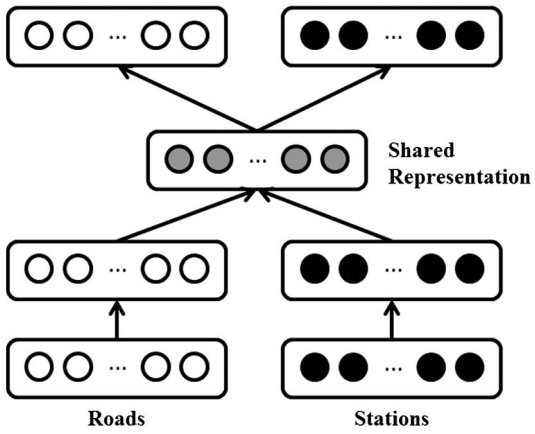


Fig. 2. Architecture of heterogeneous MTL, which learns a shared representation for heterogeneous tasks.

deep layers in the DBN would improve the features produced by these deep layers via joint fine-tuning and improve the generalization performance. Thus, improvements are gained from weight sharing.

In MTL, tasks can be divided into **homogeneous** and **heterogeneous** tasks. The difference between them is that heterogeneous tasks can also consider sharing information between tasks from heterogeneous data sources. It is also referred to as multimodal learning in some studies [20]. In our case, homogeneous tasks refer to the same kinds of observation points (all roads or all stations). The traffic flow in related roads and sections would share similar patterns, even similar values. Therefore, it is valuable to consider MTL in the deep architecture for traffic flow prediction. As shown in Fig. 1(b), a multitask regression layer, instead of a simple regression layer, is above the DBN. Not only the features are shared in multitask regression but also the process of supervised fine-tuning. In a single-task deep architecture, fine-tuning is done for each task separately. While in the multitask architecture, several related tasks would perform fine-tuning of the whole architecture jointly. It is beneficial for learning better features and improving the overall performance. In the process of fine-tuning, the model tends to find a group of weights that can produce better overall performance.

In heterogeneous MTL, the tasks of roads and stations can integrate with each other. Although the traffic flows of roads and stations are different in quantity and variation trends, they are highly correlated, and they all imply patterns of the transportation system. Vehicles enter a highway from an entrance station and leave a highway from an exit station several time intervals later. Traffic flow on the road is highly related with the number of vehicles entering and leaving the highway. Therefore, there is a lot of sharing information between roads and stations. Simultaneously training heterogeneous tasks can improve the performance of the model as well. We give an illustration of the heterogeneous multitask architecture for traffic flow prediction in Fig. 2. The input for the model is also heterogeneous, and the traffic flows of roads and stations are integrated together. The model tends to learn a shared feature representation for all tasks. Then, the shared feature representation can be used for heterogeneous tasks.

C. Task Grouping

An important problem in MTL is **grouping**. Only grouping related tasks together can improve the overall performance. It is demonstrated in [1] that improvements are made when the tasks that were jointly trained are related to each other. That is, weight sharing is effective in related tasks, whereas it is useless in unrelated tasks. Grouping aims to find the most related tasks for the best overall performance.

Since the transportation network is highly correlated, the most simple grouping method for traffic flow prediction is putting all tasks together. It may improve the performance to some extent, but it is not the best approach. Many clustering approaches can be used to find related tasks in traffic flow prediction. We can cluster spatially nearby roads or stations together as related tasks. Grouping can be also done by clustering according to the number of vehicles or the variations of the traffic flow [12], [25].

Most aforementioned clustering methods use the data for grouping. Since MTL is achieved by weight sharing, using the weight in the deep architecture may be better for grouping. Here, we propose a **weight clustering** method, which finds related tasks according to the weight of the top layer of the deep architecture. Notice that the deepest layer of the DBN is the most representative and most useful features learned from the data. It is used as the input data for the regression layer above and shared for all the related tasks. Therefore, weights in the regression layer can be seen as the connectivity of each task to these representative features. With the top layer features H^p, we can derive a weight distribution $W_i^p = \{w_1^p, w_2^p, \ldots, w_m^p\}$ for each task i by training the prediction model for each task separately or all the tasks together. Clustering according to the top layer weights can group tasks with similar weight distributions together. These tasks would have similar relationships to representative features H^p. As a result, there would be more sharing of information between these tasks. Different techniques can be employed in the detailed clustering process. Simple K-means is adopted in our experiments.

IV. Experiments and Results

A. Experimental Settings

Data Sets: Two data sets are used in this study. One benchmark data set is the PeMS data set. PeMS is the most widely used data set in traffic flow prediction. The data are collected from inductive loops, and the task is to predict the traffic flow on the road near the loop detector. This system continuously collects loop detector data in real time for more than 8100 freeway locations throughout the State of California. Then, the data are aggregated as counts of cars into 5-min periods and are accessible in the Internet for research. We further aggregate the data into 15-min periods, as suggested by the Highway Capacity Manual. The traffic flow of a road is obtained from averaging all the loop detectors in the road. Then, we choose roads of top 50 traffic flows for study since roads with large traffic flows attract more attention in transportation research. We average the data of the loop detectors in the same road to compute the traffic flow of a link road. Another data set we employed is from

the highway system of China [the entrance–exit station of a highway (EESH)]. In each EESH, there is a station for charging and recording related information. Data are collected in each station and aggregated into 15-min periods. Two prediction tasks are concerned in the EESH, i.e., predicting the traffic flow in each station and predicting the traffic flow on each road from the station flows. The traffic flow on the road is collected by loop detectors, which are similar to PeMS. However, only eight loop detectors are employed in the highway we studied. The task for PeMS is road traffic flow prediction, whereas the task for the EESH is station flow prediction in default. The EESH is mainly used in MTL experiments.

Input and Output: The input (the feature vector) in the model is the traffic flow of all observation points in previous k time intervals. The traffic flow is represented as the number of vehicles, and it would be normalized by dividing the maximum traffic flow. The output of the model is the prediction value of traffic flow in the next time interval of an observation point. In Section III-B, the output would be the traffic flow of the related observation points instead of a single point.

Training and Testing Data: Both data sets contain data during 12 months in 2011, and we use the data of the first 10 months as the training set and the later 2 months as the testing set.

Evaluation Metrics: MAPE is used for error measurement. We can get the mean accuracy (MA) by MA $= 1 -$ MAPE. For the overall performance evaluation, we use the weighted MA (WMA), which takes the traffic flow as weight, and it implies the aim of predicting high-flow areas more accurately.

Architectures: There are several parameters that we have to define for the deep architecture for traffic flow prediction, such as the nodes in each layer, the layer size, epochs, and the time intervals k of the input data, as introduced. These parameters are determined through cross validation only on the training set to ensure fairness when comparing with other approaches. In the next section, we will further analyze the effect of each parameter on the final results.

Although deep learning costs a lot in storage and computation, most of the experiments can be finished in less than 1 h in a single machine with Core i3 central processing unit, 4-GB memory, and 512-MB graphics processing unit (GPU) memory. During the process of computing, we used the GPU for acceleration.

B. Structure of Deep Architecture

Our model contains several parameters to be defined for building the architecture. Time intervals k, which determine the structure of the input data, range from 1 to 16 (15 min–4 h). We choose the layer size from one to seven layers. For simplicity, the number of nodes in each layer is set to be the same. It is chosen from $\{16, 32, 64, 128, 256, 512, 1024\}$. The epochs of training are also important in the learning phase. The model would overfit in training data when the number of epochs is too large. We let the epochs range from 10 to 100 with 10 as a gap. We first randomly choose each parameter from the possible set, and then, we choose the best configuration from 1000 random runs. The best structure we recorded is as follows: layer size $= 3$, nodes in layers $= 128$, epochs $= 40$, and time intervals $k = 4$.

TABLE I
EFFECT OF THE NUMBER OF LAYERS

Layers	WMA	Weights	Time
1	0.846	12800	83s
2	0.875	29184	219s
3	0.897	45568	337s
4	0.889	61952	466s
5	0.881	78336	574s
6	0.863	94720	688s
7	0.837	111104	820s

TABLE II
EFFECT OF THE NODES IN A LAYER

Nodes	WMA	Weights	Time
16	0.812	2112	51s
32	0.843	5248	79s
64	0.881	14592	152s
128	0.897	45568	337s
256	0.891	156672	894s
512	0.886	575488	2544s
1024	0.884	2199552	8549s

Then, we test the effect of each parameter on our deep architecture while keeping the other parameters fixed. In this step, the testing set is used to evaluate the generalization error. It is possible to find a better parameter configuration using grid search or other heuristic searching methods. However, due to the large search spaces, it would be very tedious and computationally unacceptable. Random search in a fixed set is preferred in our experiments. The default task for structure parameter selection is traffic flow prediction on PeMS.

First, we examine the influences of different network structures. The issue of network size is one of the most typical problems for NN design. The learning time and the generalization capability of the particular NN model are highly affected by the network size parameters. The result is reported in Tables I and II. Table I shows the WMA, the number of weights, and the training time with variations of the number of layers. In this experiment, the number of nodes in each layer is similarly fixed (128 here). The performance can be improved with the increase in layers from one to three. More complex structures do not have advantages over a three-layer structure. Since we only employ the training data for 10 months, models with very complex structures would be underfitted. The number of weights and the training time show the spatiotemporal complexity of each model. They almost linearly increase with the increase in layers.

Table II gives the result of the variation of the number of nodes in each layer. Similarly, 128 nodes in each layer is the best choice. Unlike the result of the number of layers, the spatiotemporal complexity increases exponentially. More nodes in each layer would cause unnecessary burdens for model training and compromise the performance. However, fewer nodes in each layer may lead the model to not be able to learn representative features. For the consideration of both simplicity and accuracy, the structure of three layers with 128 nodes in each layer is used in the later experiments.

Then, we investigate the effect of epochs and input time intervals k. Fig. 3 shows the curve of accuracy on the training set and the testing set as a function of the number of epochs. With the increase in epochs, the error on the training set can be improved, whereas the generalization capability does not

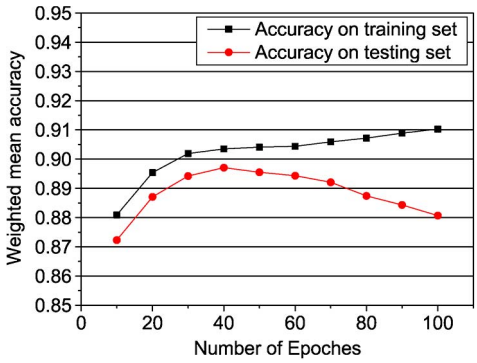


Fig. 3. Effect of epochs.

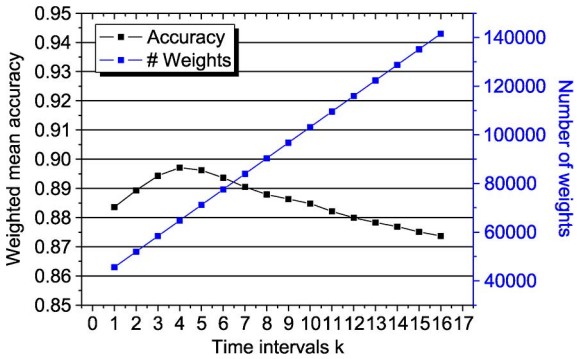


Fig. 4. Effect of input time intervals.

improve if the number of epochs is larger than 40. The model seems to be overfit on data when the number of epochs is too large. Apparently, large epochs, which would lead to a large temporal cost, is not appropriate in our model, although they can improve the accuracy on the training set.

For time intervals k (see Fig. 4), a large k would increase the size of the first layer, which can be seen from the number of weights. It is almost a linear increasing trend, but it fails at improving the performance after $k = 4$. The average travel time of cars on the road is about 1 h (four time intervals). Therefore, the traffic flow of each road in four time intervals is most related to each other. If k is over 4, more unrelated inputs would make the complex architecture have difficulty learning a good representation. In fact, time intervals k for each task (road or station) should be different. For roads with more long distance cars, a big k may be better. We should separately choose an appropriate k for each task instead of choosing the same k for each task for the best overall performance. However, for the simplicity of computation and later MTL, we used a fixed k for each task in this paper.

C. Results of DLA

In this section, we investigate the learning and generalization capabilities of our DLA, and we compare it with other existing approaches. Several widespread methods are employed for comparing approaches in this study. They are the ARIMA model [32], the Bayesian model [28], the SVR model [2], the LWL model [23], the multivariate nonparametric regression model [4], the NN model [24], and the NN-S model [3]. These models are trained and tested using the same training and testing sets as used for the deep architecture, whereas the input data may be a little bit different. The input data for the NN and NN-S methods are the same with the input for our deep architecture. Only single-task learning is considered for the sake of fair comparison.

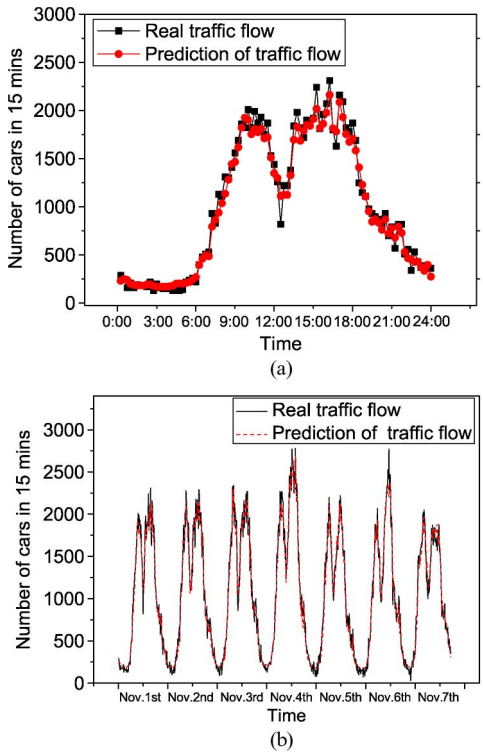


Fig. 5. Predicting the results of the road with the largest flow. (a) Prediction of the traffic flow and that of the real traffic flow in one day (November 1). (b) Prediction of the traffic flow and that of the real traffic flow in one week (November 1–November 7).

We first give an illustration of the performance of our approach in Fig. 5. The prediction of the number of vehicles and that of the real number of vehicles for the first day and the first week in the testing set are presented in Fig. 5. Our approach cannot follow the fluctuation of the number of cars when the traffic flow is low. The fluctuation is random in nonpeak periods of the day. The DLA tends to predict an average flow at that time. Our deep architecture is effective in peak times with a large traffic flow. The prediction of the traffic flow and that of the real traffic flow can match very well. It implies that the deep architecture is useful in learning the patterns of a transportation system. Since the prediction for peak times attracts more attention in a transportation system, our approach would be promising in traffic flow prediction.

Then, we compare our approach with existing methods. Four tasks are used here to evaluate each method: 1) predicting the road with the largest traffic flow; 2) predicting roads with traffic flows in the top 50 (the reported result is the weighted average of 50 roads); 3) predicting the traffic flow of the top 50 roads in peak times; and 4) predicting the traffic flow of the top 50 roads in several time intervals in advance. The results are demonstrated in Fig. 6. In Fig. 6(a), our DLA without hand-engineered features can outperform all existing approaches. For the road with the largest flow, all the approaches work well. The accuracy is over 90%, as reported in many existing

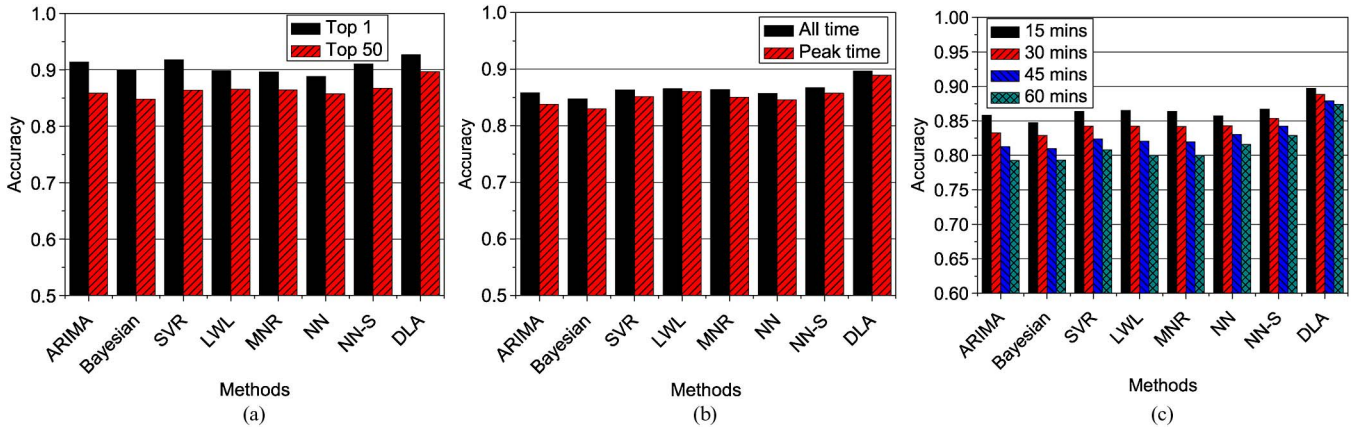


Fig. 6. Performance comparison of our deep architecture with existing approaches for traffic flow prediction with existing approaches. (a) Comparison of the prediction accuracy of the top 1 road and the top 50 roads. (b) Comparison of prediction in peak times. (c) Comparison of predicting multitime intervals.

studies. The DLA can improve the accuracy slightly (about 0.6%). The advantage of the DLA is more obvious when we take the top 50 roads into account. Existing approaches are not very effective for roads with middle and low levels of traffic flow. Due to the ability to deal with a nonlinear structure and unsupervised feature learning, the DLA can provide favorable results in almost all the roads, with an improvement on the weighted overall accuracy by over 3%. If we take the average MA into account without the weight, improvements are more obvious (over 5%). This is because improvements are bigger when only examining the middle flow or small flow roads. Roads with middle or small levels of traffic flows are not only dependent on themselves but are also related with other roads. Predicting from a network perspective would be more appropriate in this situation. In addition, middle or small flow roads are influenced by more factors than large flow roads. Unsupervised feature learning can demonstrate its advantage. The deep learning method can improve the prediction performance by discovering better feature representations. Prediction in peak times (8:00–11:00 and 16:00–19:00 in the experiments, which are determined by the traffic flow) performs nearly the same as prediction in all time periods, as presented in Fig. 6(b). The accuracy of the DLA does not drop a lot. The advantage of the DLA is still near 3%. In some roads, the prediction in peak times is even more accurate than the prediction in nonpeak times [Fig. 5(a) may be an example].

Fig. 6(c) demonstrates another advantage of our deep architecture. It is required in existing approaches that the input data should be strongly related with the output. Thus, they are usually limited in short-term traffic flow prediction. Accuracy would decrease a lot when predicting the traffic flow of several time intervals in advance. However, the DLA can be still robust for long-term traffic flow prediction. We take the top 50 roads into account in the experiments. The improvements of the prediction accuracy would increase from 3% to near 7% when the prediction time interval is increased from 15 to 60 min. We also examined the case of 12 h later. The accuracy of the DLA is still over 75%, whereas for the ARIMA model, it is only near 60%. Long-term traffic flow prediction is as important as short-term traffic flow prediction in a transportation system [33]. Transportation administrators have to estimate the possible traffic congestion in advance and make better transportation induction and guidance, and drivers also want to know the possible transportation status before they start a trip (usually several hours in advance).

In conclusion, the deep learning method is effective in traffic flow prediction. It can imply the complex relationship of a transportation system. The advantage of unsupervised feature learning would make this approach easier for application. It is promising to apply deep learning into transportation research. Many related transportation problems, such as transportation induction and transportation management, can employ the deep learning method for better results.

D. Results of MTL

The results reported earlier were all obtained in a single-task learning fashion. Here, we validate the effectiveness of MTL. The comparisons of single-task learning and MTL are all made under our deep architecture. The EESH is used here because it supports different kinds of MTL tasks.

We first examine the performance of homogenous MTL. We train a deep architecture with all tasks in the top layer and compare it with training each task separately. The results are presented in Fig. 7. Task refers to exit station flow prediction in Fig. 7(a), whereas it refers to road traffic flow prediction in Fig. 7(b). In Fig. 7(a), we can see that jointly training all tasks can improve the accuracy for most of the stations (about 90%). The accuracy improvements for some tasks can reach 4%. Similarly, for road flow prediction on the EESH, as shown in Fig. 7(b), training an eight-task learning model can improve the accuracy a lot (2.2% on average). Jointly training all tasks is the most convenient method because it only needs one run of fine-tuning rather than fine-tuning for each task. However, it is not the most effective method. As demonstrated in Fig. 7(a), not all the tasks benefited from MTL. Most of the improvements are not very significant (less than 2%) in fact. It is not the most optimal approach because some of the tasks are not related. Unrelated tasks may have a negative effect on model training. As in Fig. 7(b), only eight tasks are trained and learned together. These tasks are not only similar in traffic flow but also very near from a spatial perspective. They can be seen as related tasks.

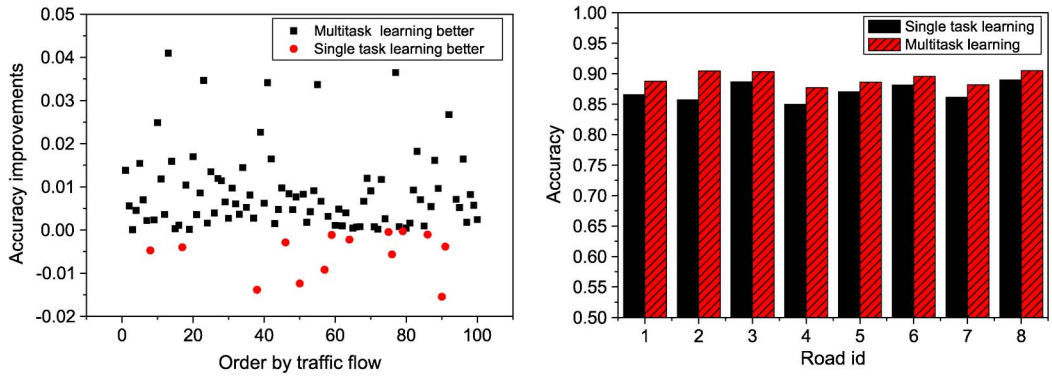


Fig. 7. Comparison of single-task learning and MTL. (a) Accuracy improvements of MTL of station flow prediction on the EESH. (b) Accuracy comparison of road flow prediction on the EESH.

TABLE III
PERFORMANCE OF HETEROGENOUS MTL

Input data	Output task	WMA of stations	WMA of roads
stations	Single task stations	79.97%	
roads	Single task roads		87.02%
stations + roads	Single task stations	80.26%	
stations + roads	Single task roads		87.15%
stations	Multi-task stations	80.54%	
roads	Multi-task roads		89.27%
stations+roads	Multi-task stations	80.68%	
stations+roads	Multi-task roads		89.39%
stations	Multi-task stations + roads	81.39%	89.98%
stations + roads	Multi-task stations + roads	**81.53%**	**90.18%**

Therefore, all tasks benefited from MTL. We can improve the overall performance via better grouping, which only integrates the most related tasks into a group. Experiments on grouping would be introduced later.

For heterogenous MTL, we tested multiple settings, as shown in Table III. Single-task learning is used as the baseline here. The results of MTL for stations and roads are the same as those for homogenous MTL. Then, we integrate the input of station flow prediction and road flow prediction together. For the output, we also put two heterogenous tasks together. We can see in Table III that such kind of heterogenous MTL is effective. The weighted overall accuracy can be improved compared with single-task learning and homogenous MTL. The setting uses both the traffic flow of stations and that of roads as the input, and predicting the traffic flow of the exit station and that of the road at the same time (bold in table) is superior to other settings. It can improve the accuracy of prediction for stations and that of prediction for roads. Although predictions for roads and stations are heterogeneous tasks, they are related to each other. Stations connect all the roads and form a transportation network. Vehicles arriving in an exit station all come from nearby roads. Therefore, two different kinds of tasks can mutually benefit from each other via information and weight sharing. It also validates the claim of MTL that only integrating related tasks together can improve the final performance. From the result, we can also see that the data fusion in the input also plays a positive role in prediction. However, the effect of MTL in the output is more obvious than that of the data fusion in the input. MTL in the output can learn shared representations via weight learning and sharing. It works in the whole process of structure learning, whereas the data fusion only takes effect on the input.

Finally, we investigate how to make MTL more effective. In previous experiments, we simply put all tasks together. Here, we use the idea of grouping to find tasks with strong relations instead of training all tasks together. As introduced in previous sections, we proposed a grouping method based on the weights of the top layer on the deep architecture. We compare it with other grouping methods, i.e., random grouping, clustering by flow, and clustering by flow variation. Flow variation is calculated as

$$v_{i,j} = \frac{f_{i,j} - f_{i,j-1}}{f_{i,j-1}}. \tag{10}$$

The input vector for grouping by flow is the average traffic flow and the average traffic flow variation in 96 time intervals, whereas the input vector for grouping by weights is the weights in the top layer, as introduced in Section IV. We use K-means as the clustering method for the aforementioned three grouping methods in the experiments. The only difference is the input vector. The number of clusters is determined by the performance on the testing set, and we use the best performance for comparison. It is different for each approach.

The results are reported in Table IV. The WMA on the EESH is for station flow prediction. The accuracy of random grouping is similar to single-task learning. It implies that training unrelated tasks cannot improve the performance. Therefore, grouping is necessary in MTL. Among all grouping methods, grouping by weights performs the best. It can outperform single-task learning by nearly 3%, whereas simultaneously training all tasks can only improve by about 0.6%. Notice that heterogenous MTL with many extra inputs can only improve

TABLE IV
PERFORMANCE OF DIFFERENT GROUPING METHODS

Method	WMA on EESH	WMA on PeMS
Single task learning	79.97%	89.71%
Grouping all	80.54%	90.56%
Random grouping	79.99%	89.75%
Grouping by flow	80.99%	91.42%
Grouping by flow variation	81.33%	91.41%
Grouping by weights	**82.37%**	**91.74%**

1.5% when training all tasks jointly. It can be seen as a relatively large improvement in the traffic flow prediction problem. Grouping by traffic flow and its variation is also better than grouping all tasks. This verified the claim that we should find related tasks for MTL since the weights in the top layer of our deep architecture are related with top-level feature representation. It also validates that unsupervised feature learning is effective in traffic flow prediction. Since we only predict roads with traffic flow in the top 50 in the experiments on PeMS, these roads are not spatially related to each other. It is interesting that grouping by weights would cluster nearby roads together when examining the results of grouping. The traffic flow and the flow variation are similar in nearby roads so that nearby roads would be also grouped together in grouping by flow and grouping by flow variation. The final grouping results between the three clustering approaches are slight. However, the traffic flow in nearby stations may vary a lot. The advantage of grouping by weights is more obvious in the results.

To summarize, MTL can be incorporated with our deep architecture easily and effectively. Both homogeneous MTL and heterogeneous MTL are useful in traffic flow prediction. The most optimal approach is grouping related tasks together. Our deep architecture for traffic flow prediction can provide helpful information for task grouping.

V. CONCLUSION

In this paper, we have presented a deep architecture for traffic flow prediction, which has been implemented as a stack of RBMs at the bottom with a regression layer at the top. The stack architecture at the bottom is a DBN, and it is effective for unsupervised feature learning. This is the first work employing deep learning in the transportation area. Without hand-engineered feature extraction and selection, our architecture can learn a good representation of features. The top regression layer is used for supervised training. We employed the idea of MTL in our deep architecture and examined that the overall performance can be improved via the MTL scheme. Furthermore, we proposed a clustering method based on the weights at the top layer of our architecture for task grouping.

From experiments on two real traffic flow data sets, we demonstrated that our deep architecture can improve the accuracy of traffic flow prediction. With limited prior knowledge, it can learn effective feature representations. Our experiments also show the validated effectiveness of MTL. Both homogenous MTL and heterogenous MTL can improve the generalization performance. Next, we presented that the most effective way of MTL is grouping related tasks. The result of our approach can outperform the state-of-the-art approach with near 5% improvements. It is a promising start of applying the deep learning method to transportation research. It implies that deep learning is not only effective in neural-related areas such as image and audio recognition. For other complex systems, it is also useful.

There are still many potential studies to do with regard to deep learning in transportation research. One study is using temporal deep NNs instead of static networks. Deep learning is traditionally used for static tasks such as image classification. The problem of how to use temporal information in traffic flow prediction would be interesting and valuable to explore. Another possible direction is building a robust prediction system based on a deep architecture. In a real application, many problems, such as missing data and data noise, would make the theoretically sound approach not practical [27]. A deep architecture is more robust than other methods due to its complex structure. It is important to use this advantage for building a practical prediction system. Finally, it would be interesting to investigate the effectiveness of deep learning in urban transportation systems. A highway is a simple transportation network, whereas urban transportation may be much more complex. A deep architecture may show its advantage in such complex systems.

REFERENCES

[1] R. Caruana, *Multitask Learning*. New York, NY, USA: Springer-Verlag, 1998.

[2] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, "Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 6164–6173, Apr. 2009.

[3] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, "Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 644–654, Jun. 2012.

[4] S. Clark, "Traffic prediction using multivariate nonparametric regression," *J. Transp. Eng.*, vol. 129, no. 2, pp. 161–168, Mar. 2003.

[5] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[6] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proc. NIPS*, 2012, pp. 1232–1240.

[7] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *Proc. IEEE ICASSP*, 2012, pp. 2133–2136.

[8] B. Ghosh, B. Basu, and M. O'Mahony, "Bayesian time-series model for short-term traffic flow forecasting," *J. Transp. Eng.*, vol. 133, no. 3, pp. 180–189, Mar. 2007.

[9] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, May 2006.

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[12] F. Jin and S. Sun, "Neural network multitask learning for traffic flow forecasting," in *Proc. IEEE IJCNN (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1897–1901.

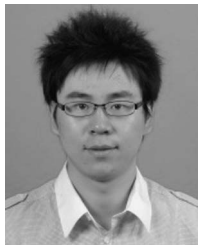
[13] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, pp. 1106–1114.

[14] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *J. Mach. Learn. Res.*, vol. 10, pp. 1–40, Jan. 2009.

[15] H. Lee, C. Ekanadham, and A. Ng, "Sparse deep belief net model for visual area v2," in *Adv. Neural Inf. Process. Syst.*, 2008, vol. 20, pp. 873–880.

[16] *Highway Capacity Manual*, Transportation Research Board, Washington, DC, USA, 2000.

[17] A.-R. Mohamed, G. Dahl, and G. Hinton, "Deep belief networks for phone recognition," in *Proc. NIPS Workshop Deep Learn. Speech Recog. Related Appl.*, 2009, pp. 1–9.
[18] C. K. Moorthy and B. G. Ratcliffe, "Short term traffic forecasting using time series methods," *Transp. Planning Technol.*, vol. 12, no. 1, pp. 45–56, Jul. 1988.
[19] B. T. Morris, C. Tran, G. Scora, M. M. Trivedi, and M. J. Barth, "Real-time video-based traffic measurement and visualization system for energy/emissions," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1667–1678, Dec. 2012.
[20] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Ng, "Multimodal deep learning," in *Proc. 28th ICML*, 2011, pp. 689–696.
[21] T. L. Pan, A. Sumalee, R. X. Zhong, and N. Indra-Payoong, "Short-term traffic state prediction based on temporal–spatial correlation," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1242–1254, Sep. 2013.
[22] R. Salakhutdinov and G. Hinton, "Using deep belief nets to learn covariance kernels for Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, vol. 20, pp. 1249–1256.
[23] M. Shuai, K. Xie, W. Pu, G. Song, and X. Ma, "An online approach based on locally weighted learning for short-term traffic flow prediction," in *Proc. 16th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2008, p. 45.
[24] B. L. Smith and M. J. Demetsky, "Short-term traffic flow prediction: Neural network approach," *Transp. Res. Rec.*, vol. 1453, pp. 98–104, 1994.
[25] S. Sun, "Traffic flow forecasting based on multitask ensemble learning," in *Proc. 1st ACM/SIGEVO Summit Genetic Evol. Comput.*, 2009, pp. 961–964.
[26] S. Sun and X. Xu, "Variational inference for infinite mixtures of Gaussian processes with applications to traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 466–475, Jun. 2011.
[27] S. Sun and C. Zhang, "The selective random subspace predictor for traffic flow forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 367–373, Jun. 2007.
[28] S. Sun, C. Zhang, and G. Yu, "A Bayesian network approach to traffic flow forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 124–132, Mar. 2006.
[29] M.-C. Tan, S. C. Wong, J.-M. Xu, Z.-R. Guan, and P. Zhang, "An aggregation approach to short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 1, pp. 60–69, Mar. 2009.
[30] Y. W. Teh and G. E. Hinton, "Rate-coded restricted Boltzmann machines for face recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 908–914.
[31] T. Thomas, W. Weijermars, and E. Van Berkum, "Predictions of urban volumes in single time series," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 71–80, Mar. 2010.
[32] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining Kohonen maps with ARIMA time series models to forecast traffic flow," *Transp. Res. C., Emerging Technol.*, vol. 4, no. 5, pp. 307–318, Oct. 1996.
[33] F.-Y. Wang, "Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 630–638, Sep. 2010.
[34] Q. Ye, W. Y. Szeto, and S. C. Wong, "Short-term traffic speed forecasting based on data recorded at irregular intervals," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1727–1737, Dec. 2012.
[35] G. Yu, J. Hu, C. Zhang, L. Zhuang, and J. Song, "Short-term traffic flow forecasting based on Markov chain model," in *Proc. IEEE Intell. Veh. Symp.*, 2003, pp. 208–212.
[36] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.
[37] W. Zheng, D.-H. Lee, and Q. Shi, "Short-term freeway traffic flow prediction: Bayesian combined neural network approach," *J. Transp. Eng.*, vol. 132, no. 2, pp. 114–121, 2006.

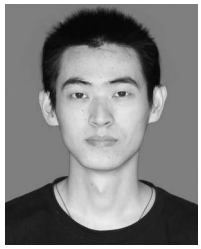


Wenhao Huang received the B.E. degree in software engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2010. Since 2010 he has been working toward the Ph.D. degree in the School of Electrical Engineering and Computer Science, Peking University, Beijing, China.

His research interests include spatiotemporal data mining, deep learning, multitask learning, intelligent transportation systems, and pervasive computing.

Guojie Song received the B.S. and M.S. degrees from Zhengzhou University, Zhengzhou, China, in 1998 and 2001, respectively, and the Ph.D. degree from Peking University, Beijing, China, in 2004.

From 2004 to 2005 he was a Research Fellow with the Singapore Management University, Singapore. He is currently an Associate Professor with the School of Electronic Engineering and Computing Science and the Vice Director of the Research Center of Intelligent Information Processing, Peking University. His research interests include various techniques of data mining, machine learning and their applications in intelligent transportation systems, and social networks.



Haikun Hong received the B.E. degree in software engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2011. Since 2011 he has been working toward the Ph.D. degree in the School of Electrical Engineering and Computer Science, Peking University, Beijing, China.

His research interests include deep neural networks, intelligent transportation systems, and spatiotemporal data mining.

Kunqing Xie received the B.S. degree from Shanxi Normal University in 1982 and the M.S. degree from Peking University in 1987. He received his Ph.D degree from Beijing Normal University in 1998.

He is a Professor with the School of Electronic Engineering and Computer Science, where he is the Dean of the Department of Intelligent Science and the Director of the Research Center of Intelligent Traffic System, Peking University, Beijing, China. He has presided over several research programs in the provincial, ministerial, and national levels, and in international cooperation. He is the author or coauthor of more than 80 papers. His research interests include spatial databases and data warehouses, spatiotemporal information analysis and data mining, remote sensing and geographic information systems, and city planning and intelligent traffic systems.

Mr. Xie received several provincial-level and ministerial-level awards in research and teaching.