On Training Traffic Predictors via Broad Learning Structures: A Benchmark Study

Di Liu, Simone Baldi, *Senior Member, IEEE*, Wenwu Yu, *Senior Member, IEEE*, Jinde Cao, *Fellow, IEEE*, and Wei Huang

***Abstract*—A fast architecture for real-time (i.e., minute-based) training of a traffic predictor is studied, based on the so-called broad learning system (BLS) paradigm. The study uses various traffic datasets by the California Department of Transportation, and employs a variety of standard algorithms (LASSO regression, shallow and deep neural networks, stacked autoencoders, convolutional, and recurrent neural networks) for comparison purposes: all algorithms are implemented in MATLAB on the same computing platform. The study demonstrates a BLS training process two-three orders of magnitude faster (tens of seconds against tens-hundreds of thousands of seconds), allowing unprecedented real-time capabilities. Additional comparisons with the extreme learning machine architecture, a learning algorithm sharing some features with BLS, confirm the fast training of least-square training as compared to gradient training.**

***Index Terms*—Broad learning system (BLS), least-square methods, real-time software, real-time training, traffic flow prediction.**

I. Introduction

TRAFFIC congestion is becoming a serious problem due to the increasing traffic volumes and to the challenges of traffic management [1], [2]. It has been shown that precise short-term traffic prediction (15–40-min ahead) can ease congestion by helping travelers in planning their path and traffic managers in implementing correct policies [3]–[6]. Deep learning architectures have emerged as a way to tackle the multipattern traffic data, such as stacked autoencoders (SAEs) with optimized structure [7], [8]; extreme learning machines (ELMs) with optimized configuration [9], [10]; neural networks with optimized structure [11]; deep neural networks (DNNs) augmented with extra features [12] or vector correction [13]; and deep belief networks combined with data-fusion [14] or regression layers [15]. The state-of-the-art essentially highlights the huge efforts needed for optimizing the deep structure or ad-hoc combining different learning structures (see also [16]–[18]). In the authors' experience [19]–[22], at least two challenges arise. First, optimizing the configuration of a deep architecture is a cumbersome task, which requires long trial and errors and high expertise [23]. Second, deep structures must be trained offline using gradient methods and/or dedicated graphical processing units (GPUs) [24]: if the generalization capabilities of the prediction are not satisfactory, the network structure must be reconfigured and the long training repeated. The combination of these two challenges prevent effective traffic predictors.

This work is motivated by the need for architectures with less time-consuming training capabilities. The recent broad learning system (BLS) framework [25]–[27] seems a promising one toward this need. Instead of improving prediction by deepening the structure, BLS extends the structure in width, allowing training via faster least-square methods [28], [29]. In this aspect, BLS shares some characteristics with ELMs [30], [31] which also use least-square methods. The distinguishing feature of BLS are incremental learning algorithms that do not require complete retraining when the network is expanded [32], [33]. To go beyond a mere application of BLS, extensive comparative experiments against state-of-the-art algorithms (LASSO regression, SAEs, shallow, deep, convolutional, and recurrent neural networks) are presented, using various traffic datasets by the California Department of Transportation (Caltrans), with flow/speed/occupancy data. The comparisons show that, when all algorithms are implemented in the same MATLAB computing platform, the training time of BLS is two-three orders of magnitude faster (tens of seconds instead of tens-hundreds of thousands of seconds), which paves the way for unprecedented real-time training capabilities. Such a fast training is achieved without sacrificing the learning performance: the prediction error of BLS

Manuscript received August 16, 2019; revised November 30, 2019 and March 22, 2020; accepted June 15, 2020. Date of publication July 20, 2020; date of current version January 17, 2022. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 3207012004A2; in part by the Special Guiding Fund for Double First-Class under Grant 3307012001A; in part by the China Scholarship Council under Grant 201906090146; in part by the Natural Science Foundation of China under Grant 61673107 and Grant 61833005; and in part by the Jiangsu Provincial Key Lab of Networked Collective Intelligence under Grant BM2017002. This article was recommended by Associate Editor H.-X. Li. *(Corresponding author: Wenwu Yu.)*

Di Liu is with the School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China, and also with the Bernoulli Institute for Mathematics, Computer Science, and Artificial Intelligence, University of Groningen, 9700 AB Groningen, The Netherlands (e-mail: liud923@126.com).

Simone Baldi is with the School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China, and also with the School of Mathematics, Southeast University, Nanjing 210096, China (e-mail: s.baldi@tudelft.nl).

Wenwu Yu is with the School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China, and also with the School of Mathematics, Southeast University, Nanjing 210096, China (e-mail: wwyu@seu.edu.cn).

Jinde Cao is with the Jiangsu Provincial Key Laboratory of Networked Collective Intelligence, Southeast University, Nanjing 210096, China, and also with the School of Mathematics, Southeast University, Nanjing 210096, China (e-mail: jdcao@seu.edu.cn).

Wei Huang is with the Intelligent Transportation System Research Center, Southeast University, Nanjing 210096, China (e-mail: hhhwei@126.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2020.3006124.

Digital Object Identifier 10.1109/TSMC.2020.3006124

is competitive against state-of-the-art algorithms.[1] Additional comparisons with the ELM architecture confirm the fast training of least-square methods as compared to gradient-based methods.

The remainder of this article is organized as follows. Section II recalls BLS. Sections III and IV present the traffic prediction test cases with extensive comparisons. Section V concludes the work.

II. Background on Broad Learning System

Let us give a quick overview of BLS, and refer to [25] for more details. BLS is constructed upon a random vector functional-link network structure [34], and comprises feature nodes and enhancement nodes arranged in a flat network. Feature nodes are nonlinear transformations of the input data, while enhancement nodes are nonlinear transformations of the feature node data. Let $X \in \mathrm{R}^{S_{num} \times M}$ be the input training data, $Y \in \mathrm{R}^{S_{num} \times N}$ be the target training data, with S_{num} the number of samples, and M, N the input and output dimensions. Let us define the ith mapped feature node as

$$Z_i = \varphi_i\big(XW_{e_i} + b_{e_i}\big), \ i = 1, \ldots, n \tag{1}$$

where φ_i is the activation function, whose weights W_{e_i} and bias b_{e_i} are randomly generated. The set of feature nodes $Z^n = [Z_1, Z_2, \ldots, Z_n]$ becomes the input to the enhancement nodes. Let us define the jth enhancement node as

$$H_j = \zeta_j\big(Z^n W_{h_j} + b_{h_j}\big), \ j = 1, 2, \ldots, m \tag{2}$$

where ζ_j is the activation function, whose weights W_{h_j} and bias b_{h_j} are also randomly generated. The set of enhancement nodes is $H^m = [H_1, H_2, \ldots, H_m]$, and the BLS structure is

$$Y = [Z_1, \ldots, Z_n | H_1, \ldots, H_m] W_n^m = \big[Z^n | H^m\big] W_n^m. \tag{3}$$

After defining $A_n^m = [Z^n | H^m]$, training is obtained via the pseudoinverse/least-square regression [25], [35]

$$W_n^m = \big(A_n^m\big)^+ Y = \Big[\big(A_n^m\big)^T A_n^m + \lambda I\Big]^{-1} \big(A_n^m\big)^T Y \tag{4}$$

where λ is a positive constant used for regularization. The resulting one-shot BLS is summarized in Algorithm 1.

Remark 1: Similarly to BLS, ELMs use nodes with randomly generated weights, organized hierarchically, with only the last layer trained via least-square regression [9], [10]. Actually, one-shot BLS and ELMs can be thought as two variants of the random vector functional-link network idea [34], where training via least-square regression was proposed. This work considers a two-layer implementation of ELM, with the feature layer and the enhancement layer organized hierarchically, with only the enhancement layer trained via least-square regression.

As compared to ELMs, the distinguishing feature of BLS is to maintain a flat structure, i.e., the feature node and the enhancement node layers are kept flat and both connected to the output layer. This helps avoiding complete retraining when the network is expanded, as explained hereafter.

[1]Even after tuning all algorithms as best as we could, it is possible that more extensive tuning leads to better performance than the one we achieved. However, such tuning would be extremely time-consuming (months), with several trial-and-error experiments for each algorithm.

Algorithm 1 BLS: One-Shot Training

```
1: Given. Training data $X(k)$ and target matrix $Y(k)$, $k = 1, \ldots, S_{num}$
2: Feature group.
   for $i = 1$, $i \leq n$
       Randomly initialize $W_{e_i}$, $b_{e_i}$
       Compute $Z_i = \varphi_i(XW_{e_i} + b_{e_i})$
       Obtain the feature mapping group
           $Z^n = [Z_1, Z_2, \cdots, Z_n]$
3: Enhancement group.
   for $j = 1$, $j \leq m$
       Randomly initialize $W_{h_j}$, $b_{h_j}$
       Compute $H_j = \zeta_j(Z^n W_{h_j} + b_{h_j})$
       Obtain the enhancement nodes group
           $H^m = [H_1, H_2, \cdots, H_m]$
4: Take $A_n^m = [Z^n | H^m]$
5: One-shot training. Compute weights
           $W_n^m = [(A_n^m)^T A_n^m + \lambda I]^{-1} (A_n^m)^T Y$
6: Given. Testing data $X_t(k)$ and target matrix $Y_t(k)$, $k = 1, \ldots, S_{t,num}$
7: Feature group.
   for $i = 1$, $i \leq n$
       Compute $Z_{t,i} = \varphi_i(X_t W_{e_i} + b_{e_i})$
       Obtain the feature mapping group
           $Z_t^n = [Z_{t,1}, Z_{t,2}, \cdots, Z_{t,n}]$
8: Enhancement group.
   for $j = 1$, $j \leq m$
       Compute $H_{t,j} = \zeta_j(Z_t^n W_{h_j} + b_{h_j})$
       Obtain the enhancement nodes group
           $H_t^m = [H_{t,1}, H_{t,2}, \cdots, H_{t,m}]$
9: Take $A_{t,n}^m = [Z_t^n | H_t^m]$
10: One-shot testing. Compute estimates
           $\hat{Y}_t = A_{t,n}^m W_n^m$
```

A. Incremental Broad Learning System

If a given BLS structure cannot provide sufficient prediction accuracy, the structure is enhanced by expanding the network in width. There are two main expansion strategies: 1) increase the number of enhancement nodes and 2) increase the number of feature nodes. Let us only recall the first one for lack of space. Let H_{m+1} be the enhancement nodes to be added. Then

$$A_n^{m+1} = \big[A_n^m | \zeta_{m+1}\big(Z^n W_{h_{m+1}} + b_{h_{m+1}}\big)\big] = \big[A_n^m | H_{m+1}\big] \tag{5}$$

with randomly generated enhancement weights $W_{h_{m+1}}$, $b_{h_{m+1}}$. The incremental BLS can be described as

$$Y = \big[A_n^m | H_{m+1}\big]\Big[W_n^m | W_n^{m+1}\Big]. \tag{6}$$

Algorithm 2 BLS: Incremental Training With Additional Enhancement Nodes

```
1: One-shot training. Same steps 1-5 as Algorithm 1
2: While the training accuracy is not satisfied do
3: Additional enhancement group.
   for $j_h = 1, j_h \leq m_h$
       Randomly initialize $W_{h_{j_h}}, b_{h_{j_h}}$
       Compute $H_{j_h} = \zeta_{j_h}(Z_{j_h} W_{h_{j_h}} + b_{h_{j_h}})$
       Obtain the additional enhancement nodes group
       $$H^{m+1} = [H_1, H_2, \cdots, H_{m_h}]$$
       $$A_n^{m+1} = [A_n^m | H_{m+1}]$$
       $$(A_n^{m+1})^+ = \begin{bmatrix} (A_n^m)^+ - DB^T \\ B^T \end{bmatrix}$$
4: Incremental training. Compute additional weights
       $$W_n^{m+1} = \begin{bmatrix} W_n^m - DB^T Y \\ B^T Y \end{bmatrix}$$
5: One-shot testing. Same steps 6-10 as Algorithm 1
6: Additional enhancement group.
   for $j_h = 1, j_h \leq m_h$
       Compute $H_{t,j_h} = \zeta_{m+1}(Z_t^n W_{h_{j_h}} + b_{h_{j_h}})$
       Obtain the additional enhancement nodes group
       $$H_t^{m+1} = [H_{t,1}, H_{t,2}, \cdots, H_{t,m_h}]$$
7: Take $A_{t,n}^{m+1} = [Z_t^n | H_t^m | H_t^{m+1}]$, $W_{t,n} = [W_n^m | W_n^{m+1}]$
8: Incremental testing. Compute estimates
       $$\hat{Y}_t = A_{t,n}^{m+1} W_{t,n}$$
```

The following pseudoinverse is defined as:

$$\left(A_n^{m+1}\right)^+ = \begin{bmatrix} \left(A_n^m\right)^+ - DB^T \\ B^T \end{bmatrix} \tag{7}$$

$$D = \left(A^m\right)^+ H_{m+1} \tag{8}$$

$$B^T = \begin{cases} C^+ & \text{if } C \neq 0 \\ \left(1 + D^T D\right)^{-1} B^T (A^m)^+ & \text{if } C = 0 \end{cases} \tag{9}$$

$$C = \zeta_{m+1}\left(Z^n W_{h_{m+1}} + b_{h_{m+1}}\right) - A^m D \tag{10}$$

which allows only the additional weights W_n^{m+1} to be trained

$$W_n^{m+1} = \begin{bmatrix} W_n^m - DB^T Y \\ B^T Y \end{bmatrix}. \tag{11}$$

The resulting incremental BLS training is summarized in Algorithm 2. For completeness, Algorithm 3 provides the incremental BLS training when new feature nodes are added.

Remark 2: As compared to the BLS MATLAB code available at the creator's website `http://www.fst.umac.mo/en/staff/pchen.html`, we have rearranged the code from a classifier to a regressor, with a set of linear features to improve prediction.

Algorithm 3 BLS: Incremental Training With Additional Feature Nodes

```
1: One-shot training. Same steps 1-5 as Algorithm 1
2: While the training accuracy is not satisfied do
3: Additional feature group.
   for $i_e = 1, i_e \leq n_e$
       Randomly initialize $W_{e_{i_e}}, b_{e_{i_e}}$
       Compute $Z_{i_e} = \phi_{i_e}(X W_{e_{i_e}} + b_{e_{i_e}})$
       Obtain the additional feature nodes group
       $$Z^{m+1} = [Z_1, Z_2, \cdots, Z_{n_e}]$$
4: Additional enhancement group.
   for $j_h = 1, j_h \leq m_h$
       Randomly initialize $W_{h_{j_h}}, b_{h_{j_h}}$
       Compute $H_{j_h} = \zeta_{j_h}(Z_{j_h} W_{h_{j_h}} + b_{h_{j_h}})$
       Obtain the additional enhancement nodes group
       $$H^{m+1} = [H_1, H_2, \cdots, H_{m_h}]$$
       $$A_n^{m+1} = [A_n^m | Z_{m+1} | H_{m+1}]$$
       $$(A_n^{m+1})^+ = \begin{bmatrix} (A_n^m)^+ - DB^T \\ B^T \end{bmatrix}$$
5: Incremental training. Compute additional weights
       $$W_n^{m+1} = \begin{bmatrix} W_n^m - DB^T Y \\ B^T Y \end{bmatrix}$$
6: One-shot testing. Same steps 6-10 as Algorithm 1
7: Additional feature group.
   for $i_e = 1, i_e \leq n_e$
       Compute $Z_{t,i_e} = \phi_{i_e}(X_t W_{e_{i_e}} + b_{e_{i_e}})$
       Obtain the additional feature nodes group
       $$Z_t^{m+1} = [Z_{t,1}, Z_{t,2}, \cdots, Z_{t,n_e}]$$
8: Additional enhancement group.
   for $j_h = 1, j_h \leq m_e$
       Compute $H_{t,j_h} = \zeta_{j_h}(Z_t^n W_{h_{j_h}} + b_{h_{j_h}})$
       Obtain the additional enhancement nodes group
       $$H_t^{m+1} = [H_{t,1}, H_{t,2}, \cdots, H_{t,m_h}]$$
9: Take $A_{t,n}^{m+1} = [Z_t^n | H_t^m | Z_t^{m+1} | H_t^{m+1}]$,
   $W_{t,n} = [W_n^m | W_n^{m+1}]$,
10: Incremental testing. Compute estimates
       $$\hat{Y}_t = A_{t,n}^{m+1} W_{t,n}$$
```

III. Real-World Traffic Prediction Test Cases

The performance measurement system (PeMS) data source is one of the most popular datasets in the traffic field. The dataset is managed and updated by the Caltrans. Data are collected from nearly 40 000 detectors across all major freeways of California. The first dataset under consideration in this work is consistent with the one in [18]. It contains the flow, speed, and occupancy rate at 33 different locations in $I405$ freeway (North-bound Interstate 405). The data are aggregated every five min, resulting in 50 000 data samples used for training (cf. Fig. 1). Three different spatial/temporal predictions are considered as follows.

1) Predicting the traffic flow/speed/occupancy 15-min ahead in time, for any of the 33 locations.

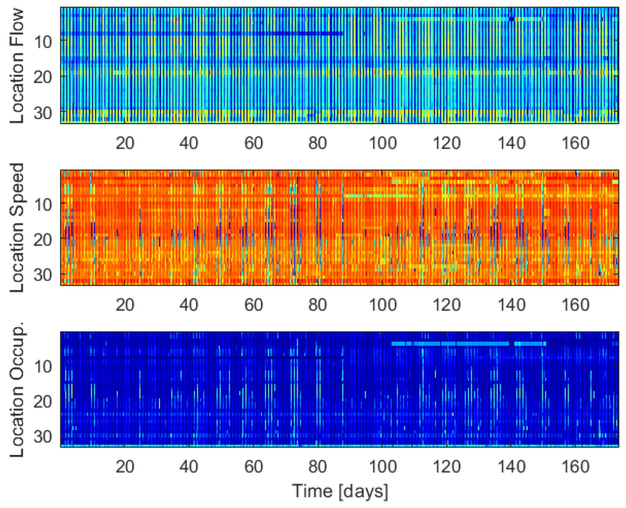

Fig. 1. Spatial/temporal traffic data used in this study.

2) Predicting the traffic flow/speed/occupancy 25-min ahead in time, for any of the 33 locations.
3) Predicting the traffic flow/speed/occupancy 40-min ahead in time, for any of the 33 locations.

As common in prediction algorithms, data have been normalized between 0 and 1, using the fact that flow has a maximum of 1300 vehicles per hour, velocity has a maximum of 90 miles per hour, and occupancy is maximum 100%.

Remark 3: Traffic predictors must be continuously updated, e.g., in the occurrence of traffic events [12]. As traffic data are typically collected on a minute base, a real-time traffic predictor processing data (including training) on a minute base would be highly desirable.

IV. Comparison With State-of-the-Art Algorithms

We compare BLS against standard methods, all implemented in the same platform (Dell Precision workstation, Intel Xeon 3.2 GHz, 8-GB RAM, and MATLAB R2017b). All algorithms have been trained according to the MATLAB documentation[2]:

1) *Shallow BP Neural Network With One Hidden Layer:* We have used two architectures: hidden layer with ten neurons, and with 25 neurons, abbreviated as BP-10 and BP-25, respectively.
2) *DNN (With More Hidden Layers):* We have used two architectures: three hidden layers with 10, 5, and 2 neurons, and four hidden layers with 10, 10, 10, and 4 neurons, abbreviated as DNN-3 and DNN-4.
3) *L1-Norm Regularized Least-Squares Regression (LASSO):* LASSO identifies the most representative features via the L1-Norm regularization, and fits them using a linear model. We have used the elastic net method with $\alpha = 0.75$, with the option to remove redundant predictors by using tenfold cross-validated fits.
4) *SAE:* Its architecture is an NN with multiple layers of sparse autoencoders in which the outputs of each layer is wired to the inputs of the successive layer. Training an SAE requires:
   a) training the first autoencoder, i.e., a sparse autoencoder on the training data without using the labels;
   b) training the second autoencoder using the features that were generated from the first autoencoder as the training data for the second autoencoder;
   c) training the final softmax layer in a supervised fashion using labels for the training data;
   d) to further improve the performance, BP is performed on the stacked neural network.

   We have used two implementations of SAE, one with first/second autoencoders of hidden size 10/5, and one with first/second autoencoders of hidden size 15/7, abbreviated as SAE-1 and SAE-2, respectively.
5) *Convolutional Neural Network (CNN):* A very standard deep learning method implementing four main operations.
   a) *Convolution (Conv):* Sliding a matrix over the data so as to extract features from the input.
   b) *Rectified Linear Unit (ReLU):* Introducing nonlinearity in the structure to capture nonlinear relationships.
   c) *Pooling (Pool):* Reducing the dimensionality of each feature map to retain the most important information.
   d) *Fully Connected Layer (FCL):* Traditional multilayer perceptron with softmax activation function, so as to use the output from convolutional and pooling layers as features for regression.

   Two implementations of CNN are adopted: one with Conv-ReLU-Pool-Conv-ReLU-FCL, and one with [Conv-ReLU-Pool](2 times)-[Conv-ReLU](2 times)-FCL. The networks are abbreviated as CNN-1 and CNN-2.
6) *Long Short-Term Memory (LSTM) CNN:* A type of recurrent neural network that can learn long-term dependencies between time steps of sequence data. The network starts with a sequence input layer followed by an LSTM layer. The network ends with an FCL. Two implementations of LSTM are adopted: one with a LSTM layer of hidden size 25, and a deeper network with an extra LSTM layer of hidden size 25. The networks are abbreviated as LSTM-1 and LSTM-2.

To further confirm the benefits of least-square training, we provide an ELM machine implementation where feature node layer and the enhancement node layer organized hierarchically, with only the enhancement layer trained via least-square

[2] backpropagation (BP), DDN: `www.mathworks.com/help/deeplearning/ref/train.html`,
LASSO: `www.mathworks.com/help/stats/lasso.html`,
SAE: `www.mathworks.com/help/deeplearning/examples/train-stacked-autoencoders-for-image-classification.html`,
CNN: `www.mathworks.com/help/deeplearning/examples/train- a-convolutional-neural-network-for-regression.html`, and
LSTM: `www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html`.

TABLE I
COMPARISON RESULTS FOR 15-min AHEAD PREDICTION. IN BOLD ARE THE SMALLEST ERRORS. DISPERSION AROUND THE MEAN ERROR IS REPORTED IN PERCENTAGE. THE GRAY BOX COMPARES THE TRAINING TIME VERSUS THE SLOWEST AND THE FASTEST BLS AS POWER OF 10

	Training time (s)	*Testing error*	*Flow*	*Speed*	*Occupancy*
BP-10	12088	MAE:	0.0277 $(1 \pm 0.72\%)$	0.0261 $(1 \pm 0.18\%)$	0.0150 $(1 \pm 4.80\%)$
	versus BLS:	MSE:	0.0410 $(1 \pm 1.36\%)$	0.0428 $(1 \pm 2.34\%)$	0.0305 $(1 \pm 10.36\%)$
	From $10^{1.9}$ to $10^{2.4}$ times slower	MAPE:	0.1559 $(1 \pm 0.77\%)$	0.0509 $(1 \pm 6.91\%)$	0.2426 $(1 \pm 15.49\%)$
BP-25	33072	MAE:	0.0277 $(1 \pm 0.92\%)$	0.0261 $(1 \pm 1.98\%)$	0.0150 $(1 \pm 0.16\%)$
	versus BLS:	MSE:	0.0410 $(1 \pm 1.42\%)$	0.0457 $(1 \pm 1.80\%)$	0.0311 $(1 \pm 1.36\%)$
	From $10^{2.4}$ to $10^{3.0}$ times slower	MAPE:	0.1661 $(1 \pm 5.20\%)$	0.0558 $(1 \pm 2.32\%)$	0.2505 $(1 \pm 2.35\%)$
DNN-3	9387	MAE:	0.0267 $(1 \pm 0.21\%)$	0.0244 $(1 \pm 0.54\%)$	0.0144 $(1 \pm 8.01\%)$
	versus BLS:	MSE:	0.0406 $(1 \pm 1.61\%)$	0.0439 $(1 \pm 0.53\%)$	0.0322 $(1 \pm 17.91\%)$
	From $10^{1.8}$ to $10^{2.4}$ times slower	MAPE:	0.1549 $(1 \pm 3.31\%)$	0.0528 $(1 \pm 2.25\%)$	0.2395 $(1 \pm 18.22\%)$
DNN-4	13879	MAE:	0.0266 $(1 \pm 5.17\%)$	0.0246 $(1 \pm 1.74\%)$	0.0142 $(1 \pm 0.93\%)$
	versus BLS:	MSE:	0.0408 $(1 \pm 8.35\%)$	0.0449 $(1 \pm 2.55\%)$	0.0309 $(1 \pm 3.38\%)$
	From $10^{2.0}$ to $10^{2.6}$ times slower	MAPE:	0.1688 $(1 \pm 17.19\%)$	0.0543 $(1 \pm 8.65\%)$	0.2312 $(1 \pm 1.26\%)$
LASSO	14328	MAE:	0.0257 $(1 \pm 0.11\%)$	0.0243 $(1 \pm 0.12\%)$	**0.0131** $(1 \pm 0.07\%)$
	versus BLS:	MSE:	0.0366 $(1 \pm 0.07\%)$	0.0424 $(1 \pm 0.18\%)$	0.0279 $(1 \pm 0.03\%)$
	From $10^{2.0}$ to $10^{2.6}$ times slower	MAPE:	0.1548 $(1 \pm 0.17\%)$	0.0513 $(1 \pm 0.09\%)$	0.1884 $(1 \pm 0.07\%)$
SAE-1	20357	MAE:	0.0267 $(1 \pm 1.09\%)$	0.0242 $(1 \pm 0.32\%)$	0.0133 $(1 \pm 1.70\%)$
	versus BLS:	MSE:	0.0382 $(1 \pm 1.42\%)$	0.0411 $(1 \pm 0.38\%)$	0.0282 $(1 \pm 1.58\%)$
	From $10^{2.2}$ to $10^{2.8}$ times slower	MAPE:	0.1588 $(1 \pm 1.01\%)$	0.0497 $(1 \pm 0.09\%)$	0.1886 $(1 \pm 2.23\%)$
SAE-2	34353	MAE:	0.0267 $(1 \pm 0.19\%)$	0.0243 $(1 \pm 0.21\%)$	0.0132 $(1 \pm 0.19\%)$
	versus BLS:	MSE:	0.0381 $(1 \pm 0.05\%)$	0.0413 $(1 \pm 0.29\%)$	0.0282 $(1 \pm 2.25\%)$
	From $10^{2.4}$ to $10^{3.0}$ times slower	MAPE:	0.1604 $(1 \pm 0.05\%)$	0.0500 $(1 \pm 0.29\%)$	**0.1865** $(1 \pm 1.05\%)$
CNN-1	65013	MAE:	0.0367 $(1 \pm 3.61\%)$	0.0363 $(1 \pm 8.01\%)$	0.0188 $(1 \pm 3.15\%)$
	versus BLS:	MSE:	0.0473 $(1 \pm 2.42\%)$	0.0525 $(1 \pm 2.87\%)$	0.0384 $(1 \pm 4.01\%)$
	From $10^{2.7}$ to $10^{3.2}$ times slower	MAPE:	0.3143 $(1 \pm 0.77\%)$	0.0743 $(1 \pm 2.61\%)$	0.3417 $(1 \pm 7.46\%)$
CNN-2	120882	MAE:	0.0334 $(1 \pm 3.65\%)$	0.0307 $(1 \pm 9.02\%)$	0.0183 $(1 \pm 4.06\%)$
	versus BLS:	MSE:	0.0435 $(1 \pm 3.50\%)$	0.0487 $(1 \pm 3.39\%)$	0.0374 $(1 \pm 3.62\%)$
	From $10^{2.9}$ to $10^{3.5}$ times slower	MAPE:	0.3081 $(1 \pm 0.94\%)$	0.0685 $(1 \pm 3.19\%)$	0.3523 $(1 \pm 7.05\%)$
LSTM-1	13234	MAE:	0.0872 $(1 \pm 1.82\%)$	0.0863 $(1 \pm 2.70\%)$	0.0275 $(1 \pm 0.27\%)$
	versus BLS:	MSE:	0.1062 $(1 \pm 2.29\%)$	0.1377 $(1 \pm 1.09\%)$	0.0500 $(1 \pm 0.85\%)$
	From $10^{2.0}$ to $10^{2.6}$ times slower	MAPE:	0.8740 $(1 \pm 5.85\%)$	0.2307 $(1 \pm 0.88\%)$	0.4231 $(1 \pm 0.17\%)$
LSTM-2	53193	MAE:	0.0410 $(1 \pm 6.41\%)$	0.0728 $(1 \pm 12.55\%)$	0.0201 $(1 \pm 9.81\%)$
	versus BLS:	MSE:	0.0605 $(1 \pm 5.83\%)$	0.1134 $(1 \pm 7.18\%)$	0.0411 $(1 \pm 6.24\%)$
	From $10^{2.6}$ to $10^{3.2}$ times slower	MAPE:	0.2816 $(1 \pm 4.00\%)$	0.1939 $(1 \pm 8.19\%)$	0.3534 $(1 \pm 13.76\%)$
ELM	27	MAE:	0.0247 $(1 \pm 0.10\%)$	**0.0239** $(1 \pm 0.14\%)$	0.0132 $(1 \pm 0.14\%)$
	versus BLS:	MSE:	0.0356 $(1 \pm 0.16\%)$	**0.0408** $(1 \pm 0.07\%)$	**0.0275** $(1 \pm 0.07\%)$
	Around $10^{0.1}$ times faster	MAPE:	0.1410 $(1 \pm 0.99\%)$	**0.0494** $(1 \pm 0.11\%)$	0.1878 $(1 \pm 0.46\%)$
BLS-one shot	36	MAE:	**0.0246** $(1 \pm 0.09\%)$	**0.0239** $(1 \pm 0.15\%)$	**0.0131** $(1 \pm 0.14\%)$
		MSE:	**0.0355** $(1 \pm 0.16\%)$	**0.0408** $(1 \pm 0.07\%)$	**0.0275** $(1 \pm 0.06\%)$
		MAPE:	**0.1409** $(1 \pm 0.99\%)$	**0.0494** $(1 \pm 0.10\%)$	0.1877 $(1 \pm 0.46\%)$
BLS-enhan.	143	MAE:	**0.0246** $(1 \pm 0.13\%)$	**0.0239** $(1 \pm 0.25\%)$	**0.0131** $(1 \pm 0.05\%)$
		MSE:	**0.0355** $(1 \pm 0.03\%)$	**0.0408** $(1 \pm 0.15\%)$	**0.0275** $(1 \pm 0.09\%)$
		MAPE:	**0.1409** $(1 \pm 0.36\%)$	0.0495 $(1 \pm 0.39\%)$	0.1875 $(1 \pm 0.31\%)$
BLS-enhan. feat.	126	MAE:	0.0247 $(1 \pm 0.32\%)$	0.0241 $(1 \pm 0.78\%)$	0.0133 $(1 \pm 0.44\%)$
		MSE:	0.0357 $(1 \pm 0.24\%)$	0.0410 $(1 \pm 0.39\%)$	**0.0275** $(1 \pm 0.14\%)$
		MAPE:	0.1426 $(1 \pm 0.80\%)$	0.0499 $(1 \pm 0.88\%)$	0.1928 $(1 \pm 1.10\%)$

regression (see Remark 1). Tuning is challenging due to the long training time: we spent several months tuning the six algorithms, but a few hours to tune BLS (and ELM), thanks to their fast training time. We have found that the following parameter search spaces work well on the PeMS dataset.

1) *BLS One-Shot (and Also ELM):* 1600 feature nodes, 3200 enhancement nodes.
2) *BLS With Incremental Enhancement Nodes (Abbreviated as BLS Enhanc.):* Initially, 1500 feature nodes and 500 enhancement nodes, with three incremental steps where 500 new enhancement nodes are added each step.
3) *BLS With Incremental Feature and Enhancement Nodes (Abbreviated as BLS Enhan.-Feat.):* Initially 550 feature nodes and 1000 enhancement nodes, with three

TABLE II
COMPARISON RESULTS FOR 25-min AHEAD PREDICTION. IN BOLD ARE THE SMALLEST ERRORS. DISPERSION AROUND THE MEAN ERROR IS REPORTED IN PERCENTAGE. THE GRAY BOX COMPARES THE TRAINING TIME VERSUS THE SLOWEST AND THE FASTEST BLS AS POWER OF 10

	Training time (s)	*Testing error*	*Flow*	*Speed*	*Occupancy*
BP-10	9369	MAE:	0.0287 $(1 \pm 1.35\%)$	0.0306 $(1 \pm 0.96\%)$	0.0170 $(1 \pm 0.65\%)$
	versus BLS:	MSE:	0.0423 $(1 \pm 2.14\%)$	0.0549 $(1 \pm 0.75\%)$	0.0369 $(1 \pm 0.59\%)$
	From $10^{1.8}$ to $10^{2.5}$ times slower	MAPE:	0.1780 $(1 \pm 3.89\%)$	0.0687 $(1 \pm 1.00\%)$	0.2782 $(1 \pm 2.48\%)$
BP-25	27316	MAE:	0.0296 $(1 \pm 0.59\%)$	0.0327 $(1 \pm 0.21\%)$	0.0175 $(1 \pm 3.31\%)$
	versus BLS:	MSE:	0.0428 $(1 \pm 0.18\%)$	0.0577 $(1 \pm 0.06\%)$	0.0362 $(1 \pm 4.24\%)$
	From $10^{2.3}$ to $10^{3.0}$ times slower	MAPE:	0.1841 $(1 \pm 3.74\%)$	0.0739 $(1 \pm 2.08\%)$	0.2725 $(1 \pm 7.10\%)$
DNN-3	5921	MAE:	0.0281 $(1 \pm 0.43\%)$	0.0308 $(1 \pm 1.02\%)$	0.0166 $(1 \pm 0.08\%)$
	versus BLS:	MSE:	0.0411 $(1 \pm 0.03\%)$	0.0568 $(1 \pm 0.26\%)$	0.0359 $(1 \pm 0.70\%)$
	From $10^{1.6}$ to $10^{2.3}$ times slower	MAPE:	0.1713 $(1 \pm 2.82\%)$	0.0700 $(1 \pm 0.16\%)$	0.2559 $(1 \pm 2.04\%)$
DNN-4	13356	MAE:	0.0281 $(1 \pm 0.41\%)$	0.0302 $(1 \pm 0.38\%)$	0.0169 $(1 \pm 7.23\%)$
	versus BLS:	MSE:	0.0416 $(1 \pm 0.05\%)$	0.0567 $(1 \pm 0.89\%)$	0.0371 $(1 \pm 8.87\%)$
	From $10^{2.0}$ to $10^{2.7}$ times slower	MAPE:	0.1731 $(1 \pm 4.41\%)$	0.0704 $(1 \pm 0.06\%)$	0.2761 $(1 \pm 20.61\%)$
LASSO	12420	MAE:	0.0296 $(1 \pm 0.01\%)$	0.0315 $(1 \pm 0.01\%)$	0.0166 $(1 \pm 0.11\%)$
	versus BLS:	MSE:	0.0413 $(1 \pm 0.01\%)$	0.0546 $(1 \pm 0.04\%)$	0.0374 $(1 \pm 0.57\%)$
	From $10^{1.9}$ to $10^{2.7}$ times slower	MAPE:	0.1908 $(1 \pm 0.10\%)$	0.0688 $(1 \pm 0.02\%)$	0.2603 $(1 \pm 0.25\%)$
SAE-1	14382	MAE:	0.0304 $(1 \pm 0.48\%)$	0.0301 $(1 \pm 0.05\%)$	**0.0163** $(1 \pm 0.09\%)$
	versus BLS:	MSE:	0.0432 $(1 \pm 1.65\%)$	0.0520 $(1 \pm 0.50\%)$	**0.0371** $(1 \pm 1.46\%)$
	From $10^{2.0}$ to $10^{2.7}$ times slower	MAPE:	0.1854 $(1 \pm 0.30\%)$	0.0639 $(1 \pm 0.10\%)$	**0.2487** $(1 \pm 0.15\%)$
SAE-2	33785	MAE:	0.0303 $(1 \pm 0.65\%)$	0.0302 $(1 \pm 0.51\%)$	0.0164 $(1 \pm 0.43\%)$
	versus BLS:	MSE:	0.0429 $(1 \pm 1.68\%)$	0.0522 $(1 \pm 0.46\%)$	0.0372 $(1 \pm 0.37\%)$
	From $10^{2.4}$ to $10^{3.1}$ times slower	MAPE:	0.1841 $(1 \pm 0.04\%)$	0.0643 $(1 \pm 0.85\%)$	0.2491 $(1 \pm 0.75\%)$
CNN-1	65013	MAE:	0.0401 $(1 \pm 5.47\%)$	0.0389 $(1 \pm 4.25\%)$	0.0202 $(1 \pm 7.95\%)$
	versus BLS:	MSE:	0.0514 $(1 \pm 4.19\%)$	0.0589 $(1 \pm 0.84\%)$	0.0422 $(1 \pm 2.04\%)$
	From $10^{2.6}$ to $10^{3.4}$ times slower	MAPE:	0.3418 $(1 \pm 4.11\%)$	0.0828 $(1 \pm 1.20\%)$	0.3834 $(1 \pm 10.68\%)$
CNN-2	107913	MAE:	0.0338 $(1 \pm 6.10\%)$	0.0343 $(1 \pm 4.56\%)$	0.0188 $(1 \pm 7.59\%)$
	versus BLS:	MSE:	0.0445 $(1 \pm 4.37\%)$	0.0574 $(1 \pm 1.36\%)$	0.0411 $(1 \pm 2.64\%)$
	From $10^{2.9}$ to $10^{3.6}$ times slower	MAPE:	0.3001 $(1 \pm 4.98\%)$	0.0800 $(1 \pm 1.50\%)$	0.3371 $(1 \pm 12.85\%)$
LSTM-1	23447	MAE:	0.0379 $(1 \pm 0.49\%)$	0.0556 $(1 \pm 1.37\%)$	0.0194 $(1 \pm 1.00\%)$
	versus BLS:	MSE:	0.0561 $(1 \pm 0.04\%)$	0.0880 $(1 \pm 0.86\%)$	0.0403 $(1 \pm 0.80\%)$
	From $10^{2.2}$ to $10^{2.9}$ times slower	MAPE:	0.2601 $(1 \pm 0.94\%)$	0.1471 $(1 \pm 1.04\%)$	0.3049 $(1 \pm 1.87\%)$
LSTM-2	66326	MAE:	0.0456 $(1 \pm 1.54\%)$	0.0730 $(1 \pm 3.41\%)$	0.0208 $(1 \pm 1.84\%)$
	versus BLS:	MSE:	0.0651 $(1 \pm 0.10\%)$	0.1135 $(1 \pm 1.23\%)$	0.0422 $(1 \pm 1.34\%)$
	From $10^{2.6}$ to $10^{3.4}$ times slower	MAPE:	0.3393 $(1 \pm 1.72\%)$	0.1935 $(1 \pm 2.64\%)$	0.3554 $(1 \pm 2.03\%)$
ELM	21	MAE:	0.0275 $(1 \pm 0.04\%)$	0.0301 $(1 \pm 0.19\%)$	**0.0163** $(1 \pm 0.16\%)$
	versus BLS:	MSE:	0.0392 $(1 \pm 0.19\%)$	**0.0519** $(1 \pm 0.04\%)$	0.0387 $(1 \pm 0.16\%)$
	Around $10^{0.1}$ times faster	MAPE:	**0.1646** $(1 \pm 1.14\%)$	0.0649 $(1 \pm 0.05\%)$	0.2646 $(1 \pm 0.67\%)$
BLS-one shot	27	MAE:	**0.0274** $(1 \pm 0.03\%)$	**0.0300** $(1 \pm 0.19\%)$	**0.0163** $(1 \pm 0.16\%)$
		MSE:	**0.0391** $(1 \pm 0.19\%)$	**0.0519** $(1 \pm 0.03\%)$	0.0387 $(1 \pm 0.17\%)$
		MAPE:	**0.1646** $(1 \pm 1.14\%)$	**0.0648** $(1 \pm 0.04\%)$	0.2646 $(1 \pm 0.68\%)$
BLS-enhan.	147	MAE:	**0.0274** $(1 \pm 0.03\%)$	**0.0300** $(1 \pm 0.23\%)$	**0.0163** $(1 \pm 0.21\%)$
		MSE:	**0.0391** $(1 \pm 0.18\%)$	**0.0519** $(1 \pm 0.05\%)$	0.0387 $(1 \pm 0.20\%)$
		MAPE:	**0.1646** $(1 \pm 1.40\%)$	**0.0648** $(1 \pm 0.13\%)$	0.2641 $(1 \pm 0.81\%)$
BLS-enhan. feat.	125	MAE:	0.0277 $(1 \pm 0.59\%)$	0.0302 $(1 \pm 0.38\%)$	**0.0163** $(1 \pm 0.11\%)$
		MSE:	0.0395 $(1 \pm 0.39\%)$	0.0521 $(1 \pm 0.23\%)$	0.0383 $(1 \pm 0.26\%)$
		MAPE:	0.1667 $(1 \pm 0.98\%)$	0.0657 $(1 \pm 0.48\%)$	0.2645 $(1 \pm 0.63\%)$

incremental steps where 50 new feature nodes and 500 new enhancement nodes are added each step.

A. Result of Comparisons on First Dataset

The result of the comparisons are in Tables I–III (15, 25, 40-min ahead prediction). The prediction performance is measured in terms of the following.

1) *Mean Absolute Error (MAE):* $1/N \sum_{i=1}^{N} |y_i - \hat{y}_i|$.
2) *Mean Square Error (MSE):* $\sqrt{1/N \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}$.
3) *Mean Absolute Percentage Error (MAPE):* $1/N \sum_{i=1}^{N} |y_i - \hat{y}_i| / |\max\{y_i, 0.01\}|$.

Where y_i denotes the value of a label, and $\hat{y}_i$ is its predicted value resulting from the algorithm. The max operator in MAPE is used to avoid division by zero. Another measure

TABLE III
COMPARISON RESULTS FOR 40-min AHEAD PREDICTION. IN BOLD ARE THE SMALLEST ERRORS. DISPERSION AROUND THE MEAN ERROR IS REPORTED IN PERCENTAGE. THE GRAY BOX COMPARES THE TRAINING TIME VERSUS THE SLOWEST AND THE FASTEST BLS AS POWER OF 10

	Training time (s)	*Testing error*	*Flow*	*Speed*	*Occupancy*
BP-10	6896	MAE:	0.0330 $(1 \pm 1.04\%)$	0.0367 $(1 \pm 2.14\%)$	0.0191 $(1 \pm 4.53\%)$
	versus BLS:	MSE:	0.0476 $(1 \pm 0.73\%)$	0.0663 $(1 \pm 1.23\%)$	0.0394 $(1 \pm 7.32\%)$
	From $10^{1.7}$ to $10^{2.3}$ times slower	MAPE:	0.2316 $(1 \pm 1.30\%)$	0.0863 $(1 \pm 2.52\%)$	0.2988 $(1 \pm 0.57\%)$
BP-25	28415	MAE:	0.0341 $(1 \pm 0.05\%)$	0.0389 $(1 \pm 1.08\%)$	0.0204 $(1 \pm 1.92\%)$
	versus BLS:	MSE:	0.0486 $(1 \pm 0.45\%)$	0.0689 $(1 \pm 1.80\%)$	0.0409 $(1 \pm 7.06\%)$
	From $10^{2.3}$ to $10^{2.9}$ times slower	MAPE:	0.2397 $(1 \pm 0.44\%)$	0.0885 $(1 \pm 1.42\%)$	0.3292 $(1 \pm 5.86\%)$
DNN-3	5905	MAE:	0.0324 $(1 \pm 1.01\%)$	0.0360 $(1 \pm 0.69\%)$	0.0192 $(1 \pm 6.13\%)$
	versus BLS:	MSE:	0.0472 $(1 \pm 1.35\%)$	0.0669 $(1 \pm 1.30\%)$	0.0402 $(1 \pm 7.30\%)$
	From $10^{1.6}$ to $10^{2.2}$ times slower	MAPE:	0.2146 $(1 \pm 5.53\%)$	0.0830 $(1 \pm 2.19\%)$	0.3340 $(1 \pm 19.44\%)$
DNN-4	6142	MAE:	0.0320 $(1 \pm 1.41\%)$	0.0357 $(1 \pm 1.33\%)$	0.0193 $(1 \pm 2.19\%)$
	versus BLS:	MSE:	0.0469 $(1 \pm 1.60\%)$	0.0675 $(1 \pm 0.83\%)$	0.0414 $(1 \pm 1.06\%)$
	From $10^{1.6}$ to $10^{2.2}$ times slower	MAPE:	0.2172 $(1 \pm 3.13\%)$	0.0803 $(1 \pm 4.73\%)$	0.3206 $(1 \pm 1.73\%)$
LASSO	12855	MAE:	0.0360 $(1 \pm 0.05\%)$	0.0385 $(1 \pm 0.13\%)$	0.0197 $(1 \pm 0.44\%)$
	versus BLS:	MSE:	0.0492 $(1 \pm 0.05\%)$	0.0651 $(1 \pm 0.08\%)$	0.0402 $(1 \pm 0.11\%)$
	From $10^{2.0}$ to $10^{2.6}$ times slower	MAPE:	0.2537 $(1 \pm 0.04\%)$	0.0872 $(1 \pm 0.05\%)$	0.3192 $(1 \pm 0.71\%)$
SAE-1	14261	MAE:	0.0364 $(1 \pm 1.76\%)$	**0.0347** $(1 \pm 0.05\%)$	0.0186 $(1 \pm 0.57\%)$
	versus BLS:	MSE:	0.0512 $(1 \pm 2.17\%)$	0.0605 $(1 \pm 0.05\%)$	**0.0394** $(1 \pm 0.76\%)$
	From $10^{2.0}$ to $10^{2.6}$ times slower	MAPE:	0.2289 $(1 \pm 0.94\%)$	**0.0777** $(1 \pm 0.69\%)$	**0.2907** $(1 \pm 1.12\%)$
SAE-2	32107	MAE:	0.0364 $(1 \pm 0.51\%)$	**0.0347** $(1 \pm 0.13\%)$	0.0185 $(1 \pm 0.76\%)$
	versus BLS:	MSE:	0.0511 $(1 \pm 0.47\%)$	0.0605 $(1 \pm 0.55\%)$	0.0395 $(1 \pm 1.59\%)$
	From $10^{2.4}$ to $10^{3.0}$ times slower	MAPE:	0.2297 $(1 \pm 1.40\%)$	0.0778 $(1 \pm 0.58\%)$	0.2912 $(1 \pm 0.72\%)$
CNN-1	62874	MAE:	0.0428 $(1 \pm 5.37\%)$	0.0435 $(1 \pm 4.20\%)$	0.0210 $(1 \pm 5.64\%)$
	versus BLS:	MSE:	0.0543 $(1 \pm 3.84\%)$	0.0666 $(1 \pm 1.67\%)$	0.0434 $(1 \pm 3.07\%)$
	From $10^{2.6}$ to $10^{3.3}$ times slower	MAPE:	0.4126 $(1 \pm 9.24\%)$	0.0954 $(1 \pm 0.94\%)$	0.3873 $(1 \pm 1.20\%)$
CNN-2	108440	MAE:	0.0368 $(1 \pm 5.28\%)$	0.0374 $(1 \pm 4.95\%)$	0.0202 $(1 \pm 4.27\%)$
	versus BLS:	MSE:	0.0488 $(1 \pm 3.62\%)$	0.0650 $(1 \pm 1.39\%)$	0.0424 $(1 \pm 2.29\%)$
	From $10^{2.9}$ to $10^{3.5}$ times slower	MAPE:	0.3186 $(1 \pm 9.71\%)$	0.0929 $(1 \pm 1.61\%)$	0.3812 $(1 \pm 1.58\%)$
LSTM-1	21940	MAE:	0.0431 $(1 \pm 1.73\%)$	0.0565 $(1 \pm 1.12\%)$	0.0217 $(1 \pm 3.62\%)$
	versus BLS:	MSE:	0.0610 $(1 \pm 1.41\%)$	0.0895 $(1 \pm 0.69\%)$	0.0427 $(1 \pm 3.15\%)$
	From $10^{2.2}$ to $10^{2.8}$ times slower	MAPE:	0.3275 $(1 \pm 1.16\%)$	0.1479 $(1 \pm 0.61\%)$	0.3559 $(1 \pm 7.38\%)$
LSTM-2	60374	MAE:	0.0512 $(1 \pm 1.91\%)$	0.0735 $(1 \pm 1.06\%)$	0.0235 $(1 \pm 4.52\%)$
	versus BLS:	MSE:	0.0706 $(1 \pm 1.74\%)$	0.1150 $(1 \pm 0.76\%)$	0.0445 $(1 \pm 3.03\%)$
	From $10^{2.6}$ to $10^{3.2}$ times slower	MAPE:	0.3985 $(1 \pm 1.42\%)$	0.1966 $(1 \pm 0.83\%)$	0.4271 $(1 \pm 6.67\%)$
ELM	29	MAE:	0.0320 $(1 \pm 0.45\%)$	0.0352 $(1 \pm 0.16\%)$	0.0183 $(1 \pm 0.23\%)$
	versus BLS:	MSE:	0.0449 $(1 \pm 0.35\%)$	**0.0604** $(1 \pm 0.08\%)$	0.0396 $(1 \pm 0.11\%)$
	Around $10^{0.1}$ times faster	MAPE:	0.2013 $(1 \pm 1.62\%)$	0.0792 $(1 \pm 0.09\%)$	0.2994 $(1 \pm 0.69\%)$
BLS-	35	MAE:	**0.0319** $(1 \pm 0.09\%)$	0.0351 $(1 \pm 0.29\%)$	**0.0182** $(1 \pm 0.11\%)$
one shot		MSE:	**0.0447** $(1 \pm 0.33\%)$	**0.0604** $(1 \pm 0.06\%)$	0.0396 $(1 \pm 0.11\%)$
		MAPE:	**0.2005** $(1 \pm 1.59\%)$	0.0789 $(1 \pm 0.06\%)$	0.2955 $(1 \pm 0.65\%)$
BLS-	143	MAE:	**0.0319** $(1 \pm 0.44\%)$	0.0351 $(1 \pm 0.17\%)$	**0.0182** $(1 \pm 0.23\%)$
enhan.		MSE:	**0.0447** $(1 \pm 0.50\%)$	**0.0604** $(1 \pm 0.05\%)$	0.0397 $(1 \pm 0.25\%)$
		MAPE:	**0.2005** $(1 \pm 0.42\%)$	0.0789 $(1 \pm 0.26\%)$	0.2955 $(1 \pm 0.50\%)$
BLS-	127	MAE:	0.0322 $(1 \pm 1.03\%)$	0.0325 $(1 \pm 0.62\%)$	0.0184 $(1 \pm 0.45\%)$
enhan.		MSE:	0.0452 $(1 \pm 0.77\%)$	0.0606 $(1 \pm 0.33\%)$	0.0396 $(1 \pm 0.13\%)$
feat.		MAPE:	0.2010 $(1 \pm 2.11\%)$	0.0795 $(1 \pm 0.72\%)$	0.2975 $(1 \pm 1.20\%)$

of performance in Tables I–III is the training time, reported in order of magnitude (power of 10) versus BLS. All results are averaged over five trials to obtain an average performance.

The training time is related to the depth of the structure: CNN-2 and CNN-1 are the slowest algorithms, LSTM-2 is slow due to the double LSTM layer, SAEs require several steps

TABLE IV
COMPARISON RESULTS FOR ON BAY-PEMS, FOR 15, 25, AND 40-min AHEAD. IN BOLD ARE THE SMALLEST ERRORS. DISPERSION AROUND THE MEAN ERROR IS REPORTED IN PERCENTAGE. THE GRAY BOX COMPARES THE TRAINING TIME VERSUS THE SLOWEST AND THE FASTEST BLS AS POWER OF 10

	Training time (s)	*Testing error*	*15 min. ahead*	*25 min. ahead*	*40 min. ahead*
BP-10	4797	MAE:	$0.0530\ (1 \pm 1.16\%)$	$0.0527\ (1 \pm 2.34\%)$	$0.0524\ (1 \pm 4.04\%)$
	versus BLS:	MSE:	$0.0821\ (1 \pm 0.81\%)$	$0.0891\ (1 \pm 1.25\%)$	$0.0854\ (1 \pm 6.84\%)$
	From $10^{2.6}$ to $10^{3.2}$ times slower	MAPE:	$0.0817\ (1 \pm 1.26\%)$	$0.0829\ (1 \pm 2.94\%)$	$0.0867\ (1 \pm 0.99\%)$
BP-25	19839	MAE:	$0.0600\ (1 \pm 0.14\%)$	$0.0604\ (1 \pm 1.02\%)$	$0.0630\ (1 \pm 1.73\%)$
	versus BLS:	MSE:	$0.0912\ (1 \pm 0.61\%)$	$0.0921\ (1 \pm 1.15\%)$	$0.0942\ (1 \pm 5.92\%)$
	From $10^{3.2}$ to $10^{3.8}$ times slower	MAPE:	$0.0931\ (1 \pm 0.49\%)$	$0.0926\ (1 \pm 1.52\%)$	$0.0990\ (1 \pm 5.48\%)$
DNN-3	3180	MAE:	$0.0546\ (1 \pm 0.86\%)$	$0.0440\ (1 \pm 0.71\%)$	$0.0419\ (1 \pm 6.00\%)$
	versus BLS:	MSE:	$0.0853\ (1 \pm 1.25\%)$	$0.0857\ (1 \pm 1.46\%)$	$0.0805\ (1 \pm 8.11\%)$
	From $10^{2.4}$ to $10^{3.0}$ times slower	MAPE:	$0.0911\ (1 \pm 4.74\%)$	$0.0713\ (1 \pm 2.46\%)$	**0.0784** $(1 \pm 15.63\%)$
DNN-4	7596	MAE:	$0.0450\ (1 \pm 1.52\%)$	$0.0497\ (1 \pm 1.61\%)$	$0.0471\ (1 \pm 2.06\%)$
	versus BLS:	MSE:	$0.0792\ (1 \pm 2.05\%)$	$0.0889\ (1 \pm 0.96\%)$	$0.0848\ (1 \pm 1.21\%)$
	From $10^{2.8}$ to $10^{3.4}$ times slower	MAPE:	$0.0739\ (1 \pm 2.89\%)$	$0.0771\ (1 \pm 3.63\%)$	$0.0792\ (1 \pm 1.88\%)$
LASSO	7931	MAE:	$0.0487\ (1 \pm 0.06\%)$	$0.0426\ (1 \pm 0.12\%)$	$0.0539\ (1 \pm 0.32\%)$
	versus BLS:	MSE:	$0.0780\ (1 \pm 0.06\%)$	$0.0734\ (1 \pm 0.11\%)$	$0.0864\ (1 \pm 0.20\%)$
	From $10^{2.8}$ to $10^{3.4}$ times slower	MAPE:	$0.0812\ (1 \pm 0.05\%)$	$0.0730\ (1 \pm 0.08\%)$	$0.0924\ (1 \pm 0.55\%)$
SAE-1	8410	MAE:	**0.0429** $(1 \pm 1.63\%)$	$0.0406\ (1 \pm 0.13\%)$	$0.0471\ (1 \pm 0.49\%)$
	versus BLS:	MSE:	$0.0718\ (1 \pm 1.90\%)$	$0.0715\ (1 \pm 0.09\%)$	$0.0798\ (1 \pm 0.67\%)$
	From $10^{2.8}$ to $10^{3.4}$ times slower	MAPE:	$0.0752\ (1 \pm 0.61\%)$	**0.0700** $(1 \pm 0.68\%)$	$0.0822\ (1 \pm 1.03\%)$
SAE-2	11884	MAE:	$0.0438\ (1 \pm 0.55\%)$	**0.0403** $(1 \pm 0.17\%)$	$0.0476\ (1 \pm 0.70\%)$
	versus BLS:	MSE:	$0.0731\ (1 \pm 0.51\%)$	$0.0708\ (1 \pm 0.48\%)$	$0.0798\ (1 \pm 1.74\%)$
	From $10^{3.0}$ to $10^{3.6}$ times slower	MAPE:	$0.0746\ (1 \pm 1.05\%)$	**0.0700** $(1 \pm 0.86\%)$	$0.0834\ (1 \pm 0.73\%)$
CNN-1	15680	MAE:	$0.1030\ (1 \pm 4.84\%)$	$0.1015\ (1 \pm 4.46\%)$	$0.1044\ (1 \pm 5.62\%)$
	versus BLS:	MSE:	$0.1340\ (1 \pm 4.04\%)$	$0.1324\ (1 \pm 2.13\%)$	$0.1362\ (1 \pm 3.55\%)$
	From $10^{3.1}$ to $10^{3.7}$ times slower	MAPE:	$0.1508\ (1 \pm 7.72\%)$	$0.1486\ (1 \pm 1.26\%)$	$0.1566\ (1 \pm 2.84\%)$
CNN-2	27481	MAE:	$0.1072\ (1 \pm 4.56\%)$	$0.1005\ (1 \pm 4.38\%)$	$0.1094\ (1 \pm 5.03\%)$
	versus BLS:	MSE:	$0.1426\ (1 \pm 3.94\%)$	$0.1310\ (1 \pm 2.93\%)$	$0.1459\ (1 \pm 3.06\%)$
	From $10^{3.3}$ to $10^{4.0}$ times slower	MAPE:	$0.1589\ (1 \pm 7.93\%)$	$0.1472\ (1 \pm 1.99\%)$	$0.1643\ (1 \pm 2.34\%)$
LSTM-1	8616	MAE:	$0.0630\ (1 \pm 1.77\%)$	$0.0546\ (1 \pm 1.35\%)$	$0.0638\ (1 \pm 3.17\%)$
	versus BLS:	MSE:	$0.0967\ (1 \pm 1.52\%)$	$0.0853\ (1 \pm 1.14\%)$	$0.0993\ (1 \pm 3.05\%)$
	From $10^{2.8}$ to $10^{3.5}$ times slower	MAPE:	$0.1076\ (1 \pm 1.67\%)$	$0.0911\ (1 \pm 0.88\%)$	$0.1114\ (1 \pm 6.53\%)$
LSTM-2	17623	MAE:	$0.0587\ (1 \pm 1.80\%)$	$0.0563\ (1 \pm 1.36\%)$	$0.1012\ (1 \pm 4.52\%)$
	versus BLS:	MSE:	$0.0984\ (1 \pm 1.62\%)$	$0.0937\ (1 \pm 1.04\%)$	$0.0802\ (1 \pm 2.95\%)$
	From $10^{3.1}$ to $10^{3.8}$ times slower	MAPE:	$0.1019\ (1 \pm 1.38\%)$	$0.0965\ (1 \pm 1.13\%)$	$0.1019\ (1 \pm 6.02\%)$
ELM	3	MAE:	$0.0445\ (1 \pm 0.09\%)$	$0.0418\ (1 \pm 0.30\%)$	$0.0471\ (1 \pm 0.11\%)$
		MSE:	$0.0688\ (1 \pm 0.32\%)$	**0.0687** $(1 \pm 0.06\%)$	**0.0751** $(1 \pm 0.10\%)$
		MAPE:	**0.0751** $(1 \pm 1.58\%)$	$0.0701\ (1 \pm 0.07\%)$	$0.0813\ (1 \pm 0.65\%)$
BLS- one shot	3	MAE:	$0.0443\ (1 \pm 0.09\%)$	$0.0417\ (1 \pm 0.29\%)$	**0.0470** $(1 \pm 0.11\%)$
		MSE:	**0.0687** $(1 \pm 0.33\%)$	**0.0687** $(1 \pm 0.06\%)$	**0.0751** $(1 \pm 0.11\%)$
		MAPE:	**0.0751** $(1 \pm 1.59\%)$	**0.0700** $(1 \pm 0.06\%)$	$0.0811\ (1 \pm 0.65\%)$
BLS- enhan.	13	MAE:	$0.0443\ (1 \pm 0.44\%)$	$0.0417\ (1 \pm 0.17\%)$	**0.0470** $(1 \pm 0.23\%)$
		MSE:	**0.0687** $(1 \pm 0.50\%)$	**0.0687** $(1 \pm 0.05\%)$	**0.0751** $(1 \pm 0.25\%)$
		MAPE:	**0.0751** $(1 \pm 0.42\%)$	**0.0700** $(1 \pm 0.26\%)$	$0.0811\ (1 \pm 0.50\%)$
BLS- enhan. feat.	12	MAE:	$0.0448\ (1 \pm 1.03\%)$	$0.0419\ (1 \pm 0.62\%)$	$0.0472\ (1 \pm 0.45\%)$
		MSE:	$0.0690\ (1 \pm 0.77\%)$	$0.0688\ (1 \pm 0.33\%)$	**0.0751** $(1 \pm 0.13\%)$
		MAPE:	$0.0753\ (1 \pm 2.11\%)$	**0.0700** $(1 \pm 0.72\%)$	$0.0814\ (1 \pm 1.20\%)$

for training. The training time for all these algorithms span from tens of thousands to hundreds of thousands of seconds. It is prohibitive to deploy and train/retrain such algorithms in real-time, whereas BLS one-shot takes tens of seconds, more than three orders of magnitude faster. Remarkably, even when extra training time is required for the incremental enhancement and feature nodes, still BLS is around two orders of magnitude faster than standard algorithms. The experiments show that, even with incremental learning, BLS allows minute-based training features, and paving the way to real-time training.

In practice, the prediction performance of all algorithms is stochastic due to the random initialization of the weights and/or nonconvexity of the optimization. To evaluate stochastic behavior, Tables I–III report the dispersion around the mean value of the error, based on an approximate Gaussian distribution with dispersion $\pm 3\sigma$. The dispersion is reported in percentage for better assessment. The percentages show that, despite BLS adopts nodes with random weights is performance is consistent and comparable or better than the other algorithms: this consistency was reported in literature also for ELMs [36], which also uses nodes with random weights.

Because they belong to the same family of algorithms stemming from random vector functional-link networks [34], let us compare BLS one-shot with ELM: ELM is a bit faster, while BLS is a bit more accurate. Both aspects can be explained by the fact that ELM trains only the last layer. Because less links have to be trained, the least-square problem of ELM is smaller than the least-square problem of BLS, and thus training is a bit faster; at the same time, the missing links lead to slightly worse performance. ELM and BLS one-shot would give exactly the same result if ELM is arranged in a flat structure (instead of hierarchical).

B. Validation on Different Dataset

To confirm the results on a different dataset, we further tested the algorithms on the California Bay dataset (abbreviated as BAY-PEMS). Similar to other traffic datasets available online, such as the Los-Angeles metropolitan dataset and the Shenzhen taxi dataset [37], [38], the dataset contains only speed data. However, it is still interesting as data are collected from 325 sections: therefore, the regression problem is even larger than the one of the first PeMS dataset. The corresponding results are reported in Table IV, respectively. Similar comments as the first dataset apply to this dataset in terms of fast training time and competitive prediction accuracy.

Remark 4: In this work, we have used the same parameter settings for all prediction cases (15, 25, 40 min ahead), resulting in a unique network structure for all predictions. Although it is not mandatory to have the same network structure, this can have some benefits in terms of computational efficiency, e.g., implementing the same network while simply switching the network gains according to the desired prediction, or distributing computations. Because distributed computing algorithms strongly depend on the network structure [39], a common network structure would ease their implementation.

Remark 5: For any learning method (except LASSO) we have used two different implementations: this is done in order to show that we have tuned all the algorithms as best as we could. Increasing sizes or adding extra layers does not in general improve performance: for example, BP-25 is not unquestionably better than BP-10, DNN-4 is not unquestionably better than DNN-3 and so on.

V. Conclusion

We have investigated a BLS as a fast architecture for traffic prediction. Extensive comparisons have been made with popular learning algorithms (LASSO, SAEs, shallow, deep, convolutional, and recurrent neural networks), with all algorithms implemented in the same computing platform. The training time of BLS is shown to be two-three orders of magnitude faster, i.e., tens of seconds against tens-hundreds of thousands of seconds.

Acknowledgment

The authors would like to thank Prof. C. L. Philip Chen for helpful discussions around the BLS paradigm.

References

[1] A. Abadi, T. Rajabioun, and P. A. Ioannou, "Traffic flow prediction for road transportation networks with limited traffic data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 653–662, Apr. 2015.

[2] F. Ahmad, S. A. Mahmud, and F. Z. Yousaf, "Shortest processing time scheduling to reduce traffic congestion in dense urban areas," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 5, pp. 838–855, May 2017.

[3] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2011, pp. 316–324.

[4] M. Clemente, M. P. Fanti, G. Iacobellis, M. Nolich, and W. Ukovich, "A decision support system for user-based vehicle relocation in car sharing systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 8, pp. 1283–1296, Aug. 2018.

[5] D. Li, L. Deng, Z. Cai, B. Franks, and X. Yao, "Intelligent transportation system in Macao based on deep self-coding learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3253–3260, Jul. 2018.

[6] L. Guo, H. Chen, Q. Liu, and B. Gao, "A computationally efficient and hierarchical control strategy for velocity optimization of on-road vehicles," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 1, pp. 31–41, Jan. 2019.

[7] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[8] H. F. Yang, T. S. Dillon, and Y.-P. P. Chen, "Optimized structure of the traffic flow forecasting model with a deep learning approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2371–2381, Oct. 2017.

[9] H. Yang, T. S. Dillon, E. Chang, and Y.-P. P. Chen, "Optimized configuration of exponential smoothing and extreme learning machine for traffic flow forecasting," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 23–34, Jan. 2019.

[10] Y. Xing, X. Ban, and R. Liu, "A short-term traffic flow prediction method based on kernel extreme learning machine," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, 2018, pp. 533–536.

[11] D. Chen, "Research on traffic flow prediction in the big data environment based on the improved RBF neural network," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2000–2008, Aug. 2017.

[12] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transp. Res. C Emerg. Technol.*, vol. 79, pp. 1–17, Jun. 2017.

[13] T. Ma, Z. Zhou, and B. Abdulhai, "Nonlinear multivariate time–space threshold vector error correction model for short term traffic state prediction," *Transp. Res. B Methodol.*, vol. 76, pp. 27–47, Jan. 2015.

[14] A. Koesdwiady, R. Soua, and F. Karray, "Improving traffic flow prediction with weather information in connected cars: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9508–9517, Dec. 2016.

[15] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct. 2014.

[16] L.-W. Chen and C.-C. Chang, "Cooperative traffic control with green wave coordination for multiple intersections based on the Internet of Vehicles," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1321–1335, Jul. 2017.

[17] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, "A multiobjective sparse feature learning model for deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3263–3277, Dec. 2015.

[18] Y. Wu, H. Tan, L. Qin, B. Ran, and Z. Jiang, "A hybrid deep learning based traffic flow prediction method and its understanding," *Transp. Res. C Emerg. Technol.*, vol. 90, no. 1, pp. 166–180, 2018.

[19] J. Guo, Z. Liu, W. Huang, Y. Wei, and J. Cao, "Short-term traffic flow prediction using fuzzy information granulation approach under different time intervals," *IET Intell. Transp. Syst.*, vol. 12, no. 2, pp. 143–150, 2018.

[20] W. Huang *et al.*, "Real-time prediction of seasonal heteroscedasticity in vehicular traffic flow series," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 10, pp. 3170–3180, Oct. 2018.

[21] D. Liu, W. Yu, S. Baldi, J. Cao, and W. Huang, "A switching-based adaptive dynamic programming method to optimal traffic signaling," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Aug. 6, 2019, doi: 10.1109/TSMC.2019.2930138.

[22] Y. Liu *et al.*, "Dynamic traffic demand uncertainty prediction using radio-frequency identification data and link volume data," *IET Intell. Transp. Syst.*, vol. 13, no. 8, pp. 1309–1317, 2019.

[23] H. Cheng *et al.* (2016). *Wide & Deep Learning for Recommender Systems*. [Online]. Available: http://arxiv.org/abs/1606.07792

[24] G. Pandey and A. Dukkipati. (2014). *To Go Deep or Wide in Learning?* [Online]. Available: http://arxiv.org/abs/1402.5634

[25] C. L. P. Chen and Z. L. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.

[26] T. Zhang, X. Wang, X. Xu, and C. L. P. Chen, "GCB-Net: Graph convolutional broad network and its application in emotion recognition," *IEEE Trans. Affect. Comput.*, early access, Aug. 27, 2016, doi: 10.1109/TAFFC.2019.2937768.

[27] T. Zhang, G. Su, C. Qing, X. Xu, B. Cai, and X. Xing, "Hierarchical lifelong learning by sharing representations and integrating hypothesis," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Feb. 27, 2019, doi: 10.1109/TSMC.2018.2884996.

[28] S. B. Jiang, P. K. Wong, R. Guan, Y. Liang, and J. Li, "An efficient fault diagnostic method for three-phase induction motors based on incremental broad learning and non-negative matrix factorization," *IEEE Access*, vol. 7, pp. 17780–17790, 2019.

[29] D. Ma, J. Wang, Q. Sun, and X. Hu, "A novel broad learning system based leakage detection and universal localization method for pipeline networks," *IEEE Access*, vol. 7, pp. 42343–42353, 2019.

[30] H. Zhou, G.-B. Huang, Z. Lin, H. Wang, and Y. C. Soh, "Stacked extreme learning machines," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 2013–2025, Sep. 2015.

[31] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.

[32] Q. Zhou and X. He, "Broad learning model based on enhanced features learning," *IEEE Access*, vol. 7, pp. 42536–42550, 2019.

[33] C. L. P. Chen, Z. Liu, and S. Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1191–1204, Apr. 2019.

[34] S. Dehuri and S.-B. Cho, "A comprehensive survey on functional link neural networks and an adaptive PSO–BP learning for CFLNN," *Neural Comput. Appl.*, vol. 19, no. 2, pp. 187–205, 2010.

[35] M. Xu, M. Han, C. P. Chen, and T. Qiu, "Recurrent broad learning systems for time series prediction," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1405–1417, Apr. 2020.

[36] R. Zhang, Y. Lan, G.-B. Huang, and Z.-B. Xu, "Universal approximation of extreme learning machine with adaptive growth of hidden nodes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 365–371, Feb. 2012.

[37] L. Zhao *et al.*, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, early access, Aug. 22, 2019, doi: 10.1109/TITS.2019.2935152.

[38] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–6.

[39] J. Chen, K. Li, K. Bilal, X. Zhou, K. Li, and P. S. Yu, "A bi-layered parallel training architecture for large-scale convolutional neural networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 965–976, May 2019.

Di Liu received the B.Sc. degree in electronic information science and technology from Hubei University of Science and Technology, Xianning, China, and the M.Sc. degree in control science and engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2014 and 2017, respectively. She is currently pursuing the double Ph.D. degree with the School of Cyber Science and Engineering, Southeast University, Nanjing, China, and with the Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, Groningen, The Netherlands.

Her research interests include adaptive learning systems and control, with application in intelligent transportation and automatic vehicles.

Simone Baldi (Senior Member, IEEE) received the B.Sc. degree in electrical engineering, and the M.Sc. and Ph.D. degrees in automatic control systems engineering from the University of Florence, Florence, Italy, in 2005, 2007, and 2011, respectively.

He is a Professor with the School of Mathematics, Southeast University, Nanjing, China, with a guest position with Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, where he was an Assistant Professor. His research interests include adaptive and learning systems with applications in smart energy and intelligent vehicle systems.

Prof. Baldi was awarded Outstanding Reviewer of *Applied Energy* in 2016, *Automatica* in 2017, and *IET Control Theory and Applications* in 2018. Since March 2019, he has been the Subject Editor of the *International Journal of Adaptive Control and Signal Processing*.

Wenwu Yu (Senior Member, IEEE) received the B.Sc. degree in information and computing science and the M.Sc. degree in applied mathematics from the Department of Mathematics, Southeast University, Nanjing, China, in 2004 and 2007, respectively, and the Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2010.

He is currently the Founding Director of the Laboratory of Cooperative Control of Complex Systems, the Deputy Associate Director of the Jiangsu Provincial Key Laboratory of Networked Collective Intelligence, an Associate Dean of the School of Mathematics, and a Full Professor with the Young Endowed Chair Honor, Southeast University. He publishes about 100 SCI journal papers with more than 10 000 citations. His research interests include multiagent systems, complex networks and systems, distributed optimization, smart grids, and intelligent transportation systems.

Dr. Yu was listed by the Clarivate Analytics/Thomson Reuters Highly Cited Researchers in Engineering in 2014 and 2019. He serves as an Editorial Board Member of several flag journals, including the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, *Science China Information Sciences*, and *Science China Technological Sciences*.

Jinde Cao (Fellow, IEEE) received the B.S. degree in mathematics from Anhui Normal University, Wuhun, China, in 1986, the M.S. degree in applied mathematics from Yunnan University, Kunming, China, in 1989, and the Ph.D. degree in applied mathematics from Sichuan University, Chengdu, China, in 1998.

He is an Endowed Chair Professor, the Dean of the School of Mathematics, the Director of the Jiangsu Provincial Key Laboratory of Networked Collective Intelligence of China, and the Director of the Research Center for Complex Systems and Network Sciences, Southeast University, Nanjing, China.

Prof. Cao was a recipient of the National Innovation Award of China, Obada Prize, and the Highly Cited Researcher Award in Engineering, Computer Science, and Mathematics by Thomson Reuters/Clarivate Analytics. He is elected as a member of the Academy of Europe and the European Academy of Sciences and Arts, a fellow of the Pakistan Academy of Sciences, an IASCYS Academician, and a full member of Sigma Xi.

Wei Huang received the B.S., M.S., and Ph.D. degrees in road engineering from Southeast University, Nanjing, China, in 1982, 1986, and 1995, respectively.

He is currently a Distinguished Professor of Civil Engineering with the Intelligent Transportation System Research Center, Southeast University. He is a Member of the Chinese Academy of Engineering. He enjoys the State Council Special Allowance and receives support from the New Century Talent Program, the National Outstanding Mid-Aged Experts Program, the National Talents Engineering Program, and the Yangtze Scholar Program from various agencies and organizations. He has authored or coauthored 13 books. He is one of the forerunners in the research fields of long-span steel bridge pavement and intelligent transportation systems of China.

Dr. Huang as the leading awardee, he received 26 awards from both the national and provincial level.