Optimized Structure of the Traffic Flow Forecasting Model With a Deep Learning Approach

Hao-Fan Yang, Tharam S. Dillon, *Life Fellow, IEEE*, and Yi-Ping Phoebe Chen, *Senior Member, IEEE*

***Abstract*— Forecasting accuracy is an important issue for successful intelligent traffic management, especially in the domain of traffic efficiency and congestion reduction. The dawning of the big data era brings opportunities to greatly improve prediction accuracy. In this paper, we propose a novel model, stacked autoencoder Levenberg–Marquardt model, which is a type of deep architecture of neural network approach aiming to improve forecasting accuracy. The proposed model is designed using the Taguchi method to develop an optimized structure and to learn traffic flow features through layer-by-layer feature granulation with a greedy layerwise unsupervised learning algorithm. It is applied to real-world data collected from the M6 freeway in the U.K. and is compared with three existing traffic predictors. To the best of our knowledge, this is the first time that an optimized structure of the traffic flow forecasting model with a deep learning approach is presented. The evaluation results demonstrate that the proposed model with an optimized structure has superior performance in traffic flow forecasting.**

***Index Terms*— Deep learning, forecasting, neural network (NN) applications, stacked denoising autoencoders.**

I. Introduction

IMPROVING the accuracy of traffic flow forecasting can bring benefit to the intelligent traffic management on traffic efficiency and congestion reduction [1], [2]. Traffic flow forecasting is quite a complex process that is affected by many factors, such as traffic patterns, applied areas, data collection, and so on. In recent computational traffic flow forecasting approaches, researchers have to select the traffic features and modeling parameters from the obtained data according to some fundamental assumptions applied in the past literature. That is, the correctness of these assumptions may crucially affect forecasting accuracy. However, if the volume of data is big enough, the hidden statistical information and potential correlations will be revealed by the data set itself [3]. Therefore, if a large volume of traffic data is adopted, we may be able to avoid many failures caused by assumptions and the accuracy of traffic prediction can be improved by learning the information and correlations hidden in the data [3], [4].

Manuscript received January 18, 2016; revised April 5, 2016; accepted May 28, 2016. Date of publication January 18, 2016; date of current version September 15, 2017. This work was supported by the Australia Research Linkage Grants Scheme. *(Corresponding author: Yi-Ping Phoebe Chen.)*

The authors are with the Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia (e-mail: h16yang@students.latrobe.edu.au; tharam.dillon7@gmail.com; phoebe.chen@latrobe.edu.au).

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/TNNLS.2016.2574840

Computational forecasting approaches, such as neural networks (NNs) [5], Bayesian modeling [6], statistical modeling [7], fuzzy logic [8], and hybrid modeling [9], [10], have been widely used in traffic flow forecasting. These approaches, particularly NNs, are proven to be useful in discovering the statistical rules hidden in the traffic data and predicting the future traffic flow. However, most of them are shallow traffic models and the computed forecasting results need to be more accurate. Moreover, with widespread traffic sensing facilities and advancements in sensing technologies [11], [12], the amount of collected traffic flow data can be greatly increased. The dawning of the big data era brings opportunities to greatly promote the improvement of prediction accuracy [13]. These inspire us to extend and apply the concept of deep learning for traffic flow forecasting. Deep learning is a type of machine learning method based on learning representations of data, and it has been successfully applied to assist in many fields, for instance, classification tasks, information processing, pattern recognition, and object detection [14], [15]. The concept of deep learning is to use deep architectures (multiple layers of nonlinear processing units) to extract and transform the inherent features in the data from the lowest level to the highest level, and every continuous layer uses the output from the previous layer as input [4], [16]. As traffic flow prediction is a complicated process, deep learning related algorithms can represent traffic features from the obtained data without prior knowledge, which may greatly improve forecasting accuracy.

In this paper, we propose a novel model, stacked autoencoder Levenberg–Marquardt (SAE-LM) model, which is a type of deep architecture of the NN approach aiming to improve the accuracy of short-term traffic flow forecasting. The SAE-LM model is designed using the Taguchi method to develop an optimized structure and to learn traffic flow features through layer-by-layer feature granulation with a greedy layerwise unsupervised learning algorithm and is applied to real-world data collected from the M6 freeway in the U.K. To the best of our knowledge, this is the first time that an optimized structure of the traffic flow forecasting model with a deep learning approach is presented. Using a solid experimental setup and execution, we obtained useful results that can be used for further research.

The rest of this paper is organized as follows. Section II reviews the current studies on NNs for traffic flow forecasting and expresses the motivation. Section III explains the methodologies and designs used in this paper. Section IV details the experimental results and evaluates the performance of the

SAE-LM model with an optimized configuration. Finally, the conclusion is given in Section V.

II. Current Applications of Neural Networks for Traffic Flow Forecasting and Motivation

Traffic flow forecasting is based on collected traffic data to make predictions about the likely traffic flow changes in the short term, and it can provide the predictive functionality for traffic management and control strategies [9], [17]. A number of traffic flow forecasting models have been developed to improve transportation efficiency and a wide variety of techniques have been adopted to optimize these models. Using NNs is one of the most popular approaches in traffic flow forecasting because of its learning capabilities. NNs are a kind of information processing technique, which can be trained to learn a relationship in a data set. Even if the underlying relationships in a data set are not apparent, the NN-based model is still capable of generalizing accurate predictions due to its nonlinear and nonparametric nature [18]. Many NN-based forecasting models have been applied to improve transportation efficiency and have been proven to be effective in solving problems within the complex relationships involved, for instance, traffic pattern recognition, traffic data classification, and traffic flow prediction [9], [19], [20].

Hybrid exponential smoothing and the Levenberg–Marquardt (LM) algorithm were used to train NNs for short-term traffic flow forecasting in [21]. The forecasting model was developed based on the traffic flow data (traffic speeds and headways) collected from detector stations located on the Mitchell freeway in Western Australia. The exponential smoothing approach was applied to remove the lumpiness from the captured traffic flow data, as this lumpiness may reduce the generalization capability [22]. The LM algorithm was then adopted to train NNs since it provides a numerical solution to the nonlinear problem, which can minimize a function over a space of the function parameters, and also, it is stable and can generate good convergence [23]. A three-layered NN was implemented to predict the future traffic flow condition. The experimental results showed that the proposed NN-based forecasting model can deliver forecasting results with the mean of the forecasted error less than 13%.

An integration of the particle swarm optimization algorithm and NNs was adopted to develop short-term traffic flow predictors in [9]. Simple multilayer (three layers) NN-based models were used to address the strongly nonlinear characteristics of short-term traffic flow data. The particle swarm optimization algorithm was adopted to represent both structures and parameters of short-term traffic flow predictors. That is, optimal structures and parameters can be determined automatically without involving some of the fundamental assumptions applied in the past literature or trial-and-error methods based on human expertise [24]. The proposed model was applied to predict traffic speeds and densities on a section of freeway in Western Australia. The forecasting accuracy was evaluated using mean absolute error (MAE) and root-mean-square error (RMSE). The experimental results indicated that the proposed model is able to predict the traffic flow conditions with an average MAE of less than 16% and an average RMSE of less than 17% on all forecasted business days, which is more accurate than the Kalman filter [25] and genetic algorithms [26].

A radial basis function NN (RBFNN) model was proposed in [27] to exploit the spatiotemporal features from the volume of adjacent intersection to further optimize forecasting performance. The RBFNN is a three-layer feedforward NN with one radial basis layer (hidden layer), which can uniformly approximate any continuous function with a prospected accuracy. It also has good local generalization abilities and faster convergence speed compared with other NN-based methods [28]. The traffic volume data (the amount of vehicles, a maximum of 15-min traffic flow rate, and a minimum of 15-min traffic flow rate) were collected by the loop detectors in Baotou City, China. The forecasting accuracy was evaluated by mean absolute deviation (MAD), mean absolute percentage error (MAPE), and RMSE. The results of the experiments showed that using more related input can improve the performance, with the MAD, MAPE, and RMSE being 13.6%, 12.2%, and 16.3%, respectively.

In summary, a large number of NN-based short-term traffic flow predictors have been developed, but most of them are shallow traffic models. The traffic features and parameters used in these models are selected according to some fundamental assumptions or are determined by applying trial-and-error methods. If we can find a way to reduce the uncertainty brought on by some of the assumptions and lower the time consumed by trial-and-error methods, the accuracy and efficiency of traffic flow forecasting can be greatly improved. Moreover, although the existing traffic flow predictors can deliver decent forecasting results, there is still some room for accuracy improvement. Inspired by the above reasons, this paper proposes a deep architecture of NN-based approach, the SAE-LM model, to reduce the uncertainty caused by some assumptions and improve the accuracy and efficiency of traffic flow forecasting. The optimized structure of the proposed model is designed using the Taguchi method, which has been successfully used to optimize NNs for traffic flow forecasting [5]. The Taguchi method uses an orthogonal array to study the effect of design factors simultaneously by a planning matrix, and it reduces the number of experiments dramatically compared with the trial-and-error method [29]. For instance, for an NN-based forecasting model with five design factors, each of which contains three levels, using a trial-and-error method requires 243 (3^5) experiments. When an orthogonal array $L_{18}(3^5)$ is used, only 18 experiments are needed to evaluate the effect of each design factor to determine the optimum condition.

III. Methodologies

In this section, the methodologies of the proposed SAE-LM model and the model design based on the Taguchi method are explained.

A. Stacked Autoencoder Levenberg–Marquardt Model

A stacked autoencoder (SAE) model uses autoencoders as building blocks to create a deep architecture [30].

The autoencoder is typically implemented as a one-hidden-layer NN, which is trained to reproduce its input. It is used as building blocks to train deep networks, where each hidden layer is associated with an autoencoder that can be trained separately. The amount of output nodes is equal to the amount of input nodes, and one input node is trained at a time. In general, an autoencoder takes a vector input x, encodes it to a hidden layer h, and decodes it to a reconstruction z. For example, given a set of data $\{x_1^{(l)}, x_2^{(l)}, \ldots, x_n^{(l)}\}$ of n training samples, where $x_i^{(l)} \in R^D$ denotes the unit i in layer l, an input $x_i^{(l)}$ is first mapped to a latent representation $h^{(l+1)}(x_i^{(l)})$ based on (1), and then the hidden representation $h^{(l+1)}(x_i^{(l)})$ is mapped back into a reconstruction $z^{(l+2)}(x_i^{(l)})$ according to (2). Equations (1) and (2) are shown as

$$h(x) = f(W_1 * x + b_1) \tag{1}$$

$$z(x) = g(W_2 * h(x) + b_2) \tag{2}$$

where W_1, W_2, b_1, and b_2 are the encoding matrix, decoding matrix, encoding bias vector, and decoding bias vector, respectively. $f(\cdot)$ and $g(\cdot)$ are the activation functions, and we follow the custom of using the same activation function for both mappings (encoding and decoding). Since the inputs in this research (i.e., the collected traffic data) are real numbers, we use the squared loss function $L(x, z)$ to measure the reconstruction error [14]. The model parameters, denoted by θ, can be obtained by minimizing the average difference between the input x and the reconstruction z (i.e., reconstruction error), as shown in

$$\theta = \arg\min_{\theta} L(x, z) = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \left\| x_i^{(l)} - z\left(x_i^{(l)}\right) \right\|^2. \tag{3}$$

There is a serious issue concerning autoencoders in that if the hidden units are the same size or greater than the input units, an autoencoder could potentially learn the identity function and become useless (e.g., just by copying the input) [14]. However, the experiment results in [30] show that the autoencoder with more hidden units than inputs can yield useful representations when trained with stochastic gradient descent. Moreover, current practice indicates that adding noise in the encoding or constraining the encoding by sparseness can successfully ensure that the above issue does not occur [31]. In this study, we choose to impose a sparsity constraint [32] on the hidden units. For a sigmoid activation function, a neuron is seen as being active if its output value is close to 1 or being inactive if its output value is close to 0 (note that different activation functions may have different output values). We constrain the neurons to be inactive most of the time. Suppose $a_j^{(l+1)}(x)$ denotes the activation of unit j in layer $l + 1$ when the network is given a specific input x. Let

$$\hat{\rho}_j = \frac{1}{n} \sum_{i=1}^{n} \left[a_j^{(l+1)}\left(x_i^{(l)}\right) \right] \tag{4}$$

be the average activation of unit j and impose the constraint

$$\hat{\rho}_j = \rho \tag{5}$$

Algorithm 1 Kullback–Leibler Divergence

Parameters: KLD(A, B)- Kullback-Leibler divergence of matrix A and B,
Matrix A- sparsity parameter,
Matrix B- average activation of each hidden neuron,
Nc()-number of columns, Nr()-number of rows,
Array Sc()- the sum of column's values on each row;

C = repeat copies of Sc(A) into a 1-by-Nc(A) block arrangement;

D = repeat copies of Sc(B) into a 1-by-Nc(B) block arrangement;

E = repeat copies of B into a Nr(A)-by-1 block arrangement;

if Nr(B) == 1

B = element-wise divide B by the sum of B;

A = element-wise divide A by C;

F = element-wise multiply A by log(E);

Replacing the not a number element in F by 0;

KLD(A,B)=Sc(F);

else if Nr(B) == Nr(A)

B = element-wise divide B by D;

A = element-wise divide A by C;

G = element-wise divide A by B;

F = element-wise multiply A by log(G);

Replacing the not a number element in F by 0;

KLD(A,B)=Sc(F);

end;

where ρ is a sparsity parameter whose value is close to zero. To achieve this, an extra penalty term that penalizes $\hat{\rho}_j$ deviating from ρ is added as follows:

$$\sum_{j=1}^{N} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{6}$$

where N is the number of neurons in the hidden layer and the index j is summing over the hidden units in the network. This penalty term is based on the concept of Kullback–Leibler (KL) divergence, where

$$\mathrm{KL}(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}. \tag{7}$$

The KL divergence penalty function (Algorithm l) has the property that $\mathrm{KL}(\rho \| \hat{\rho}_j) = 0$ if $\hat{\rho}_j = \rho$, which provides the constraint by sparseness on the coding. If the autoencoder is implemented using the backpropagation algorithm, the gradient descent will be performed exactly on the cost function $L_{\mathrm{Sparse}}(x, z)$ [32]. The overall cost function (reconstruction error with a sparsity constraint) can then be formulated as

$$L_{\mathrm{Sparse}}(x, z) = L(x, z) + \alpha \sum_{j=1}^{N} \mathrm{KL}(\rho \| \hat{\rho}_j) \tag{8}$$

where α is the weight of the sparsity term.

In order to create a deep architecture, autoencoders are stacked one by one from the bottom layer. Having trained the

Algorithm 2 Levenberg–Marquardt Algorithm (LM)

```
Parameters: μ -control parameter for LM to train the network,
            g-gradient,
            S-sum of squared errors, H-Hessian matrix,
            I-identity matrix,
            J-Jacobian matrix, e-a vector of network errors;
  Initialize μ =0.001, μ_max=1∗10^10, μ_min=1∗10^(-10)
  % Rearrange all the weights in a vector (V)
        for i=1:max_i;
            H= J^T J; g=J^T e;
            dx=g∗(H+ μ∗I)^-1;
            V_b=V;
            count=0;
                while count ≤ 10
                   V=V_b+dx;
                      if S(i)<=S(i-1)
                         if μ > μ_max
                            μ = μ/5;
                         end;
                         if μ < μ_min
                            μ = μ∗5;
                         end;
                      end;
            count++;
        end;
```

bottom layer autoencoder [the first hidden layer $h^{(2)}(x_i^{(1)})$] to minimize the reconstruction error of the raw input, its hidden unit outputs are used as input for the next autoencoder [the second hidden layer $h^{(3)}(h^{(2)}x_i^{(1)})$]. Iterating above process until the desired number of auto-encoders is added. The last hidden layer's output is taken as input to a supervised learning algorithm to fine-tune all the parameters of this deep architecture with respect to the supervised criterion [33]. In this research, we use the LM [23] on top of the whole network to train the input generated by the last autoencoder. The LM (Algorithm 2) is adopted since it can provide a numerical solution to the nonlinear problem minimizing a function over a space of the function parameters and also it is stable and can generate good convergence [20], [23]. Fig. 1 gives an illustration of the development of the proposed SAE-LM model.

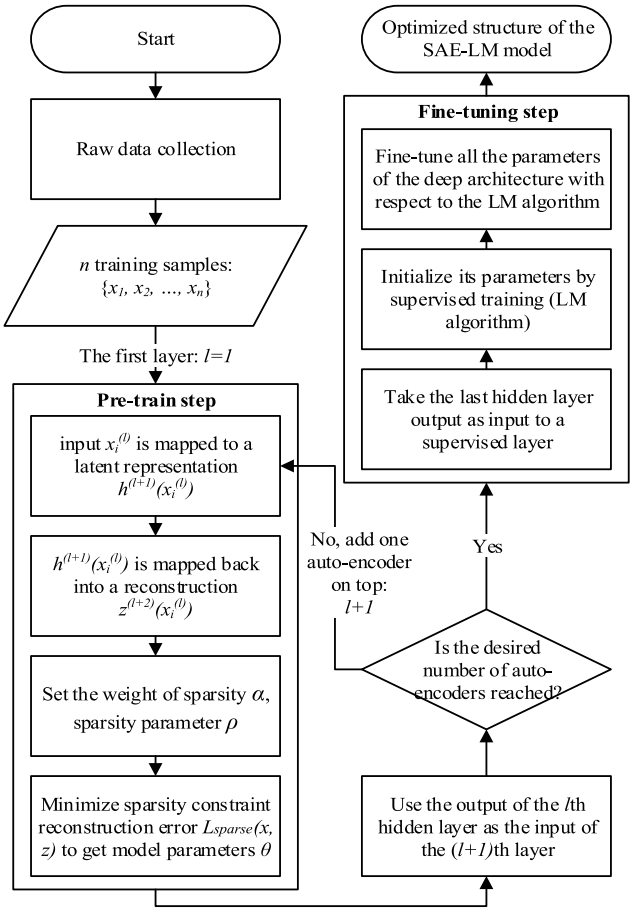

Fig. 1. Development of the optimized structure of the SAE-LM model.

B. Data Description and Forecasting Model

The traffic flow condition is sensed by inductive loops built into the road surface located at several junctions (J6, J7, J17, J18, J34, and J35) on the M6 freeway in the U.K. on the working days between March 3, 2014 and March 28, 2014 (20 working days). Both sides of the observed road sections are three lanes in width, and the road length between J6 to J7 is 7.3 km, J17 to J18 is 6.4 km, and J34 to J35 is 7.2 km. The collected traffic data were divided into six road links data sets, namely, *R1a* (J6 to J7), *R1b* (J7 to J6), *R2a* (J17 to J18), *R2b* (J18 to J17), *R3a* (J34 to J35), and *R3b* (J35 to J34). In all the six data sets, the traffic flow data collected from the first 15 working days are used to develop the forecasting model, and the remaining data (five working days) are used to evaluate the performance of the proposed model.

The sensed traffic flow condition is collected every 1 min from many sensing stations (S_1, S_2, S_3, ..., S_a), which are located at the observed road sections. S_i captures two traffic condition measures, the average vehicle speed $\hat{V}_i(t)$ and the average headway $\hat{h}_i(t)$ between time t and time $t + T_S$, where T_S is the sampling time. Since the data is collected every 1 min, T_S can be set as 1. In light of the current and past traffic conditions, future traffic conditions can be forecasted by the proposed SAE-LM model. The current traffic condition is denoted by the current average speed $\hat{V}_i(t)$ and current average headway $\hat{h}_i(t)$. The past traffic condition is indicated by the past average speed $\hat{V}_i(t-k)$ and past average headway $\hat{h}_i(t-k)$, which was collected by S_i at time $(t-k)$ with $i = 1, 2, 3, \ldots, a$ and $k = 1, 2, 3, \ldots, p$, whereas the past traffic data within p sampling time period are collected. Two data sets $\{V_1^{(l)}, V_2^{(l)}, \ldots, V_a^{(l)}\}$ and $\{h_1^{(l)}, h_2^{(l)}, \ldots, h_a^{(l)}\}$ can be generated, where $V_i^{(l)}, h_i^{(l)} \in R^D$ denotes the collected traffic conditions in layer l. Suppose the outputs of the last autoencoder are $z^{(l+2)}(V_i^{(l)})$ and $z^{(l+2)}(h_i^{(l)})$, the future traffic condition can then be calculated by the SAE-LM model, which is indicated by $\hat{h}_L(t+m)$ passing through the Lth sensing station S_L at time $(t+m)$, where the future traffic condition with m sampling time ahead is forecasted. The SAE-LM

TABLE I
DESIGN FACTORS FOR OPTIMIZING THE STRUCTURE OF THE PROPOSED SAE-LM MODEL

Design factors		Level			
		1	2	3	4
i	Sampling time (minutes)	30	60	240	600
ii	Activation functions of the auto-encoders	Sigmoid	Tanh	LogSig	SoftSign
iii	The number of hidden nodes	$\log_2(N)$	$2\times\log_2(N)$	$3\times\log_2(N)$	50
iv	The number of hidden layers	3	4	5	6
v	Activation functions of the latest hidden set, *C(.)*	Logsig	Tansig	Purelin	Same as the design factor ***ii***

forecasting model is formulated as

$$\hat{h}_L(t+m) = \sum_{i=1}^{a}\sum_{j=1}^{M}\left[\beta_{j,i}^{V} C\left(\gamma_{0,j,i}^{V}\sum_{k=1}^{p}\gamma_{k,j,i}^{V} z^{(l+2)}\left(V_i^{(l)}\right)\right) + \beta_{j,i}^{h} C\left(\gamma_{0,j,i}^{h}\sum_{k=1}^{p}\gamma_{k,j,i}^{h} z^{(l+2)}\left(h_i^{(l)}\right)\right)\right] + \alpha_0 \quad (9)$$

where M is the number of nodes in the hidden layer, $\boldsymbol{C}(\cdot)$ is the activation function of the hidden set, $V_i^{(1)} = \hat{V}_i(t-k)$ and $h_i^{(1)} = \hat{h}_i(t-k)$, and $\beta_{j,i}^{V}$, $\beta_{j,i}^{h}$, $\gamma_{0,j,i}^{V}$, $\gamma_{0,j,i}^{h}$, $\gamma_{k,j,i}^{V}$, $\gamma_{k,j,i}^{h}$, and α_0 are the parameters of the proposed forecasting model.

C. Taguchi Method and Experiment Design

The Taguchi method is applied to obtain the optimized structure of the proposed traffic flow forecasting model, the SAE-LM model, which is trained with the SAE and LM algorithms to establish a short-term traffic forecasting predictor. The Taguchi method is used to optimize the topology of the SAE-LM model by the following three sections.

D. Identification of Design Factors

In the design of the optimized structure of the SAE-LM model, the considered design factors and their corresponding levels are as follows and are presented in Table I.

1) *Design Factor i:* The number of training nodes is determined by the sampling time period. Hence, in this research, we consider sampling times of 30, 60, 240, and 600 min as Levels 1, 2, 3, and 4, respectively.
2) *Design Factor ii:* For the activation functions of the autoencoders, $f(\cdot)$ and $g(\cdot)$, the most commonly used transfer functions for deep structures are Sigmoid, Tanh, LogSig, and SoftSign [4], [34]. We follow the practice of using the same activation function for $f(\cdot)$ and $g(\cdot)$, and Sigmoid, Tanh, LogSig, and SoftSign functions are set as Levels 1, 2, 3, and 4, respectively.
3) *Design Factor iii:* The number of hidden nodes and the number of hidden layers are important since they determine the size of the SAE-LM model for the short-term traffic flow predictor. The minimum number of hidden nodes recommended by [35] is $\log_2(N)$, where N is the number of training nodes. In addition, the maximum number of hidden nodes suggested by [36] to solve modeling problems is 50. Therefore, we set $\log_2(N)$, $2\times\log_2(N)$, $3\times\log_2(N)$, and 50 as Levels 1, 2, 3, and 4, respectively.
4) *Design Factor iv:* The number of hidden layers is a significant design factor, and in this research, it depends on how many autoencoders are stacked. For example, if two autoencoders are stacked, the number of hidden layers is three (i.e., the number of autoencoders plus the last one hidden layer, which is trained by LM). The accuracy rate can be improved with more autoencoders in the forecasting model; however, the computational time will increase dramatically. With a different number of autoencoders used in the forecasting model, the relationship between the accuracy rate and computational time is illustrated in Fig. 2. Since the accuracy rate and computational time are both important in traffic forecasting, we select the number of autoencoders that can generate predictions with more than 80% accuracy rate in less than 5 s. Hence, according to the results, three, four, five, and six hidden layers (two, three, four, and five autoencoders) are considered as Levels 1, 2, 3, and 4, respectively.
5) *Design Factor v:* For the activation functions of the last hidden set, $C(\cdot)$, Logsig, Tansig, and Purelin functions are commonly used for traffic forecasting networks [5], [21], [37]. Therefore, Logsig, Tansig, Purelin, and the function that is the same as the one used for the autoencoders (design factor ii) are set as Levels 1, 2, 3, and 4, respectively. Note: Levels 1 and 4 of design factor v are the same (Logsig) when Level 3 of design factor ii is used.

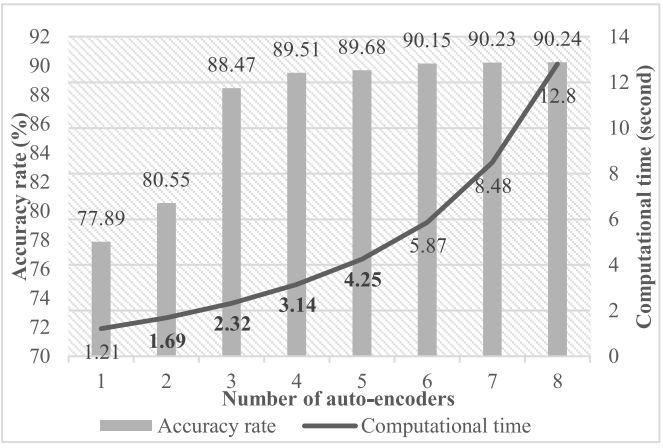

Fig. 2. Relationship between accuracy rate and computational time for a different number of autoencoders. The traffic forecasting model is trained with a 120-min sampling time, the SoftSign activation function for all hidden layers, and ten hidden nodes. The computational time is measured on a 3.40-GHz Intel i7 CPU, 64-b operating system, and 16.0-GM RAM.

E. Specification of Performance Measure

To evaluate the effectiveness of the SAE-LM model for traffic forecasting, we use two performance indexes, the MAPE and variance absolute percentage error (VAPE).

MAPE and VAPE indicate the mean and variance of the difference between the actual and predicted values, respectively. Mathematical models for MAPE and VAPE are defined as

$$\text{MAPE}(\%) = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \times 100\% \tag{10}$$

$$\text{VAPE}(\%) = \text{Var}\left(\left|\frac{A_t - F_t}{A_t}\right|\right) \times 100\% \tag{11}$$

where A_t is the observed traffic flow condition, F_t is the predicted traffic flow condition, and n is the number of forecasted points. The lower the value of MAPE and VAPE, the higher the accuracy of the predicted result.

F. Trial Design and Results Analysis

An orthogonal array $L_{16}(4^5)$ is used for the trial design due to the fact that there are five design factors and each one contains four levels. Based on the combination of the levels of design factors, 16 main trials are conducted to obtain the optimized structure of the SAE-LM model. For each main trial corresponding to each row of the adopted orthogonal array $L_{16}(4^5)$, the training set of the collected traffic data in the traffic data sets (*R1a*, *R1b*, *R2a*, and *R2b*) is used to develop the SAE-LM model. The results of three random days (March 24, 2014; March 27, 2014; and March 28, 2014) and the average results of the testing set with respect to the 16 main trials are shown in Table II.

A total of 1280 trials, 16 main trials over 20 days using four traffic data sets (*R1a*, *R1b*, *R2a*, and *R2b*), were conducted to evaluate the effectiveness and robustness of the SAE-LM model using the Taguchi method design. If the full factorial design is used, 81 920 ($4^5 \times 20 \times 4$) trials need to be carried out, where four levels in each of the five design factors over the 20 observed days using four traffic data sets are used for the traffic forecasting model design. Compared with the number of trials required by the Taguchi method with the full factorial design, 80 640 trials (81 920-1280) can be saved in the design of the SAE-LM model. This demonstrates that using the Taguchi method in this research is reasonable and effective.

As shown in Table II, the tenth main trial with a 240 min sampling time, Tanh function for the autoencoders, 50 hidden nodes, 5 hidden layers, and Logsig function for the last hidden set achieves the smallest MAPE and VAPE. It is much smaller compared with the 11th and 12th main trials, which utilize the same sampling time. The top three ranks are the 5th, 10th, and 16th main trials, which denotes that the SAE-LM model with five hidden layers can achieve better predicted results than using three, four, or six hidden layers. Moreover, the 1st, 7th, 12th, and 14th main trials show that MAPE and VAPE perform poorly, which indicates that applying only three hidden layers (two autoencoders) cannot generate accurate forecasting results. Comparing the results of the 3rd, 5th, 10th, and 16th main trials, we found that when the SAE-LM model with five hidden layers, the sampling time was 240 min. The accuracy decreases greatly when reducing the sampling time from 240 to 30 min. The above results clearly demonstrate

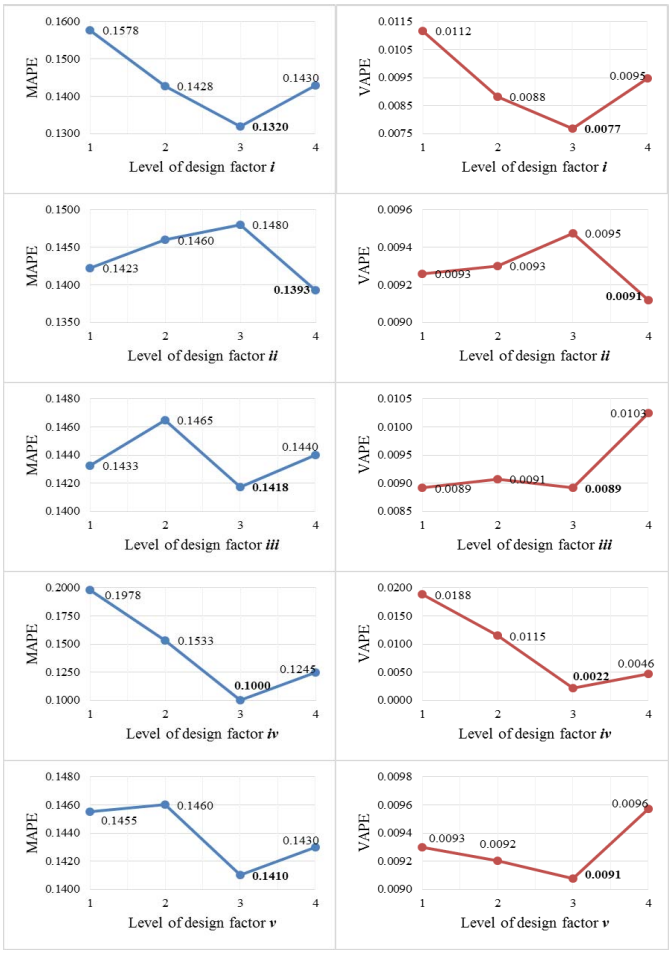

Fig. 3. Effects of each design factor at each of the four levels. The level with the largest effect (bold) of each design factor are Level 3 for design factor i, Level 4 for design factor ii, Level 3 for design factor iii, Level 3 for design factor iv, and Level 3 for design factor v.

that the proposed SAE-LM model with a 240-min sampling time and five hidden layers (four autoencoders) can generate the predicted results with high accuracy.

The effects of each design factor can be separated since the combinations of design factors of each trial are orthogonal [38]. The effects of each design factor at each of the four levels are calculated by taking the average from Table II for a design factor at a given level. For example, the Level 2 of the design factor iii is in the 2nd, 5th, 12th, and 15th main trials, and the average of this level is 0.1465 (MAPE) and 0.0091 (VAPE). Since a lower value of MAPE and VAPE indicates better performance, the lower the average value obtained means the larger the effect. The effects of each design factor at each of the four levels are displayed in Fig. 3. An intuitive method, range analysis, is applied to express the sensitivity of the design factors to the performance values. In this research, this is computed by the difference between the values with the largest and the smallest effect for each design factor, and the results are shown in Table III. The order of the sensitivity of the five design factors is iv > i > ii > v > iii in MAPE and iv > i > iii > v > ii in VAPE. From the above order, we can find that design factor iv and

TABLE II

ORTHOGONAL ARRAY $L_{16}(4^5)$ AND EXPERIMENTAL RESULTS

Main Trials	Design factor/Level					Day	***R1a***		***R1b***		***R2a***		***R2b***		Average results of 5 working days (24th to 28th March 2014)			
	i	***ii***	***iii***	***iv***	***v***		MAPE	VAPE	MAPE	VAPE	MAPE	VAPE	MAPE	VAPE	MAPE	Rank	VAPE	Rank
1	1	1	1	1	1	24th	0.194	0.019	0.205	0.020	0.204	0.019	0.208	0.021	0.211	16	0.020	16
						27th	0.199	0.018	0.208	0.020	0.200	0.020	0.209	0.020				
						28th	0.201	0.018	0.192	0.019	0.202	0.018	0.205	0.020				
2	1	2	2	2	2	24th	0.172	0.013	0.173	0.013	0.175	0.014	0.173	0.013	0.174	12	0.013	12
						27th	0.174	0.013	0.173	0.013	0.171	0.013	0.174	0.013				
						28th	0.176	0.014	0.172	0.013	0.173	0.013	0.174	0.013				
3	1	3	3	3	3	24th	0.111	0.003	0.114	0.004	0.113	0.004	0.112	0.004	0.113	5	0.004	6
						27th	0.112	0.003	0.115	0.004	0.112	0.004	0.112	0.003				
						28th	0.114	0.004	0.113	0.004	0.109	0.003	0.114	0.004				
4	1	4	4	4	4	24th	0.134	0.008	0.131	0.009	0.132	0.009	0.129	0.008	0.133	8	0.008	8
						27th	0.134	0.009	0.132	0.009	0.133	0.009	0.130	0.009				
						28th	0.132	0.008	0.130	0.008	0.132	0.009	0.131	0.009				
5	2	1	2	3	4	24th	0.095	0.001	0.094	0.001	0.099	0.002	0.096	0.002	0.099	3	0.002	3
						27th	0.098	0.002	0.095	0.002	0.096	0.002	0.101	0.002				
						28th	0.097	0.002	0.094	0.002	0.094	0.001	0.106	0.002				
6	2	2	1	4	3	24th	0.122	0.004	0.121	0.004	0.123	0.004	0.120	0.003	0.122	6	0.004	5
						27th	0.124	0.004	0.120	0.003	0.122	0.004	0.121	0.004				
						28th	0.119	0.003	0.120	0.003	0.124	0.004	0.121	0.003				
7	2	3	4	1	2	24th	0.201	0.020	0.200	0.020	0.198	0.019	0.206	0.020	0.203	15	0.019	15
						27th	0.206	0.020	0.202	0.020	0.206	0.020	0.200	0.020				
						28th	0.208	0.019	0.199	0.019	0.206	0.019	0.202	0.020				
8	2	4	3	2	1	24th	0.144	0.010	0.141	0.010	0.139	0.011	0.145	0.011	0.147	10	0.011	10
						27th	0.142	0.011	0.146	0.012	0.143	0.012	0.145	0.011				
						28th	0.141	0.010	0.150	0.012	0.142	0.011	0.150	0.012				
9	3	1	3	4	2	24th	0.114	0.002	0.108	0.002	0.109	0.003	0.107	0.002	0.111	4	0.003	4
						27th	0.110	0.003	0.108	0.003	0.112	0.003	0.108	0.003				
						28th	0.109	0.002	0.111	0.003	0.113	0.003	0.110	0.003				
10	**3**	**2**	**4**	**3**	**1**	**24th**	**0.091**	**0.001**	**0.089**	**0.002**	**0.091**	**0.001**	**0.093**	**0.002**	**0.092**	**1**	**0.002**	**1**
						27th	**0.094**	**0.002**	**0.095**	**0.002**	**0.088**	**0.002**	**0.094**	**0.002**				
						28th	**0.098**	**0.002**	**0.095**	**0.002**	**0.090**	**0.001**	**0.094**	**0.001**				
11	3	3	1	2	4	24th	0.140	0.009	0.137	0.009	0.136	0.010	0.144	0.011	0.144	9	0.010	9
						27th	0.138	0.009	0.137	0.010	0.141	0.010	0.142	0.010				
						28th	0.138	0.010	0.137	0.009	0.141	0.010	0.143	0.011				
12	3	4	2	1	3	24th	0.168	0.017	0.170	0.019	0.182	0.017	0.168	0.017	0.181	13	0.017	13
						27th	0.173	0.018	0.170	0.019	0.189	0.018	0.165	0.016				
						28th	0.171	0.017	0.169	0.017	0.186	0.018	0.170	0.017				
13	4	1	4	2	3	24th	0.140	0.010	0.142	0.011	0.141	0.013	0.148	0.013	0.148	11	0.012	11
						27th	0.143	0.010	0.145	0.011	0.145	0.012	0.147	0.012				
						28th	0.143	0.011	0.148	0.011	0.143	0.012	0.152	0.013				
14	4	2	3	1	4	24th	0.199	0.018	0.198	0.019	0.194	0.019	0.193	0.019	0.196	14	0.019	14
						27th	0.201	0.019	0.199	0.019	0.196	0.019	0.196	0.019				
						28th	0.194	0.018	0.196	0.018	0.195	0.019	0.198	0.020				
15	4	3	2	4	1	24th	0.131	0.005	0.129	0.005	0.132	0.005	0.133	0.005	0.132	7	0.005	7
						27th	0.130	0.005	0.130	0.005	0.132	0.004	0.134	0.005				
						28th	0.130	0.005	0.128	0.004	0.134	0.005	0.131	0.004				
16	4	4	1	3	2	24th	0.099	0.001	0.090	0.001	0.100	0.002	0.099	0.002	0.096	2	0.002	2
						27th	0.099	0.002	0.096	0.002	0.097	0.002	0.097	0.001				
						28th	0.100	0.002	0.093	0.001	0.098	0.002	0.098	0.002				

The results of 3 random days (24th, 27th and 28th March 2014) and the average results of the last 5 working days (24th to 28th March 2014) in the ***R1a***, ***R1b***, ***R2a***, and ***R2b*** datasets with respect to the 16 main trials.

TABLE III

EFFECTS AND THE SENSITIVITY OF EACH DESIGN FACTOR

Design factor	MAPE					VAPE				
	i	***ii***	***iii***	***iv***	***v***	***i***	***ii***	***iii***	***iv***	***v***
Level 1	0.1578	0.1423	0.1433	0.1978	0.1455	0.0112	0.0093	0.0089	0.0188	0.0093
Level 2	0.1428	0.1460	0.1465	0.1533	0.1460	0.0088	0.0093	0.0091	0.0115	0.0092
Level 3	<u>0.1320</u>	0.1480	<u>0.1418</u>	<u>0.1000</u>	<u>0.1410</u>	<u>0.0077</u>	0.0095	<u>0.0089</u>	<u>0.0022</u>	<u>0.0091</u>
Level 4	0.1430	<u>0.1393</u>	0.1440	0.1245	0.1430	0.0095	<u>0.0091</u>	0.0103	0.0046	0.0096
Sensitivity	0.0258	0.0087	0.0048	**0.0978**	0.0050	0.0035	0.0004	0.0013	**0.0166**	0.0005

The underlined values denote the largest effect in the corresponding design factor (the lower the value, the larger the effect), and the **Bold** values denote the greatest sensitivity in all 5 design factors.

design factor i are always the primary factors in the proposed SAE-LM forecasting model.

According to Fig. 3 and Tables II and III, the optimized structure design of the SAE-LM model in traffic flow forecasting can be inferred as design factor i with Level 3, design factor ii with Level 4, design factor iii with Level 3, design factor iv with Level 3, and design factor v with Level 3. That is, the SAE-LM model with a

TABLE IV
FORECASTING ACCURACY OF SAE-LM, EXP-LM, PSONN, AND RBFNN

Traffic forecasting model	Forecasting accuracies											
	R3a						***R3b***					
	24th March	25th March	26th March	27th March	28th March	5-days average	24th March	25th March	26th March	27th March	28th March	5-days average
SAE-LM	**90.23%**	**90.00%**	**90.31%**	**90.50%**	**90.74%**	**90.36%**	**89.55%**	**90.62%**	**90.18%**	**90.39%**	**91.15%**	**90.38%**
EXP-LM	86.00%	83.54%	83.70%	85.27%	84.64%	84.63%	84.99%	83.74%	85.21%	85.10%	83.47%	84.50%
PSONN	84.20%	82.10%	81.77%	82.98%	81.58%	82.53%	83.60%	83.27%	84.13%	82.76%	85.29%	83.81%
RBFNN	84.32%	82.23%	82.10%	83.68%	83.41%	83.15%	84.20%	84.06%	84.52%	83.33%	83.86%	83.99%

Values in **Bold** refer to the highest accuracy rate on each of the observed days.

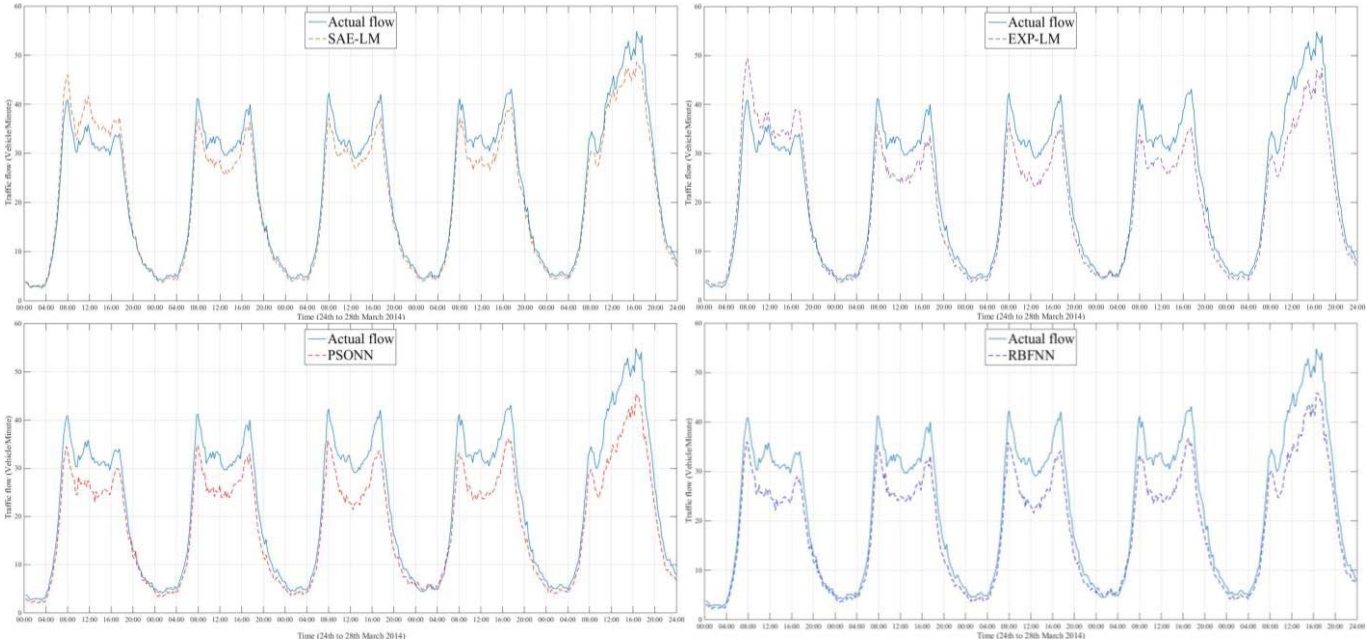

Fig. 4. Performance evaluation of SAE-LM, EXP-LM, PSONN, and RBFNN traffic flow forecasting predictors for the R3a data set. The value of traffic flow is represented by the vehicle/minute, and the time index starts at 00:00 on March 24, 2014 and ends at 24:00 on March 28, 2014. Since the performance of each model for R3a is similar to R3b, we display only the forecasting results for R3a.

240-min sampling time, SoftSign function for the autoencoders, $3 \times \log_2(N)$ hidden nodes, five hidden layers (four autoencoders), and Purelin function for the last hidden set generates the most accurate predicted results.

IV. PERFORMANCE EVALUATION OF THE SAE-LM MODEL WITH AN OPTIMIZED STRUCTURE

This section demonstrates the accuracy and efficiency of the SAE-LM model with the optimized structure for traffic flow forecasting by comparing the predictors introduced in Section II, which are hybrid exponential smoothing and the LM algorithm with NNs (EXP-LM), particle swarm optimization algorithm with NNs (PSONN), and RBFNNs.

Performance Evaluation: The traffic data in *R3a* and *R3b* are used for performance evaluation. Table IV shows the accuracy rate of the traffic prediction on each of the observed days. The results show that the proposed SAE-LM model is more accurate than the EXP-LM, PSONN, and RBFNN models for the prediction of the traffic flow. Importantly, SAE-LM is able to generate forecasting results with approximately 90% accuracy, which is 5% better on most of the observed days compared with the other predictors. For the SAE-LM, the prediction performance is relatively stationary from 89.55% to 91.15%. The difference between the maximum and minimum accuracy rates for SAE-LM is 1.6%, which is better than 2.53% for EXP-LM, 3.71% for PSONN, and 2.42% for RBFNN.

Fig. 4 presents the output of the SAE-LM, EXP-LM, PSONN, and RBFNN models for traffic flow prediction on March 24–28, 2014. The actual (observed) traffic flow is also included in Fig. 4 for comparison. As can be seen in Fig. 4, the predicted traffic flow of all four models has similar traffic patterns to the actual traffic flow. However, an obvious difference occurs between the values of the actual and predicted traffic flows at 07:00 to 18:00 on each of the observed days, especially in the forecast results generated by the EXP-LM, PSONN, and RBFNN models. This phenomenon implies that the proposed SAE-LM model can forecast more accurately than the other models when the traffic flow is rising and falling rapidly over a short period of time. In contrast, in the period during which the traffic flow does not change

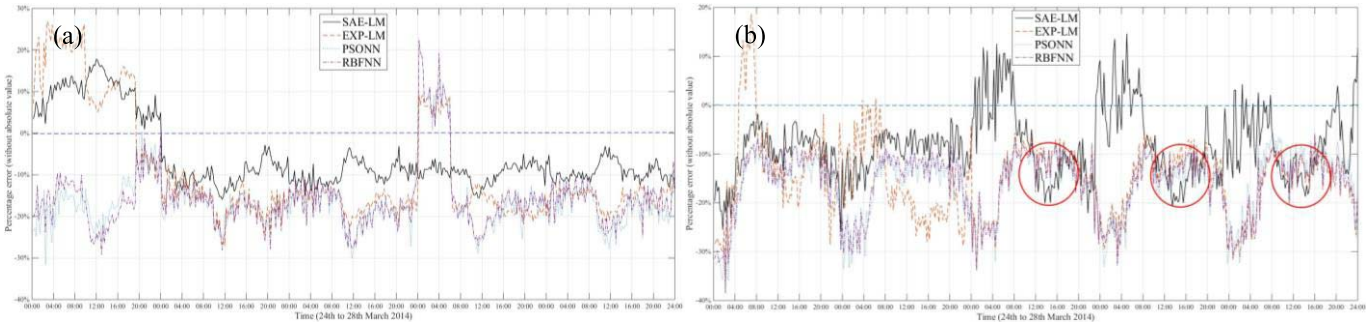

Fig. 5. Percentage error of traffic flow forecasting results for (a) R3a and (b) R3b. The percentage error is calculated without absolute value. The ovals in (b) indicate the time periods when the forecasting result of SAE-LM has a larger percentage error than the other models. The closer the percentage error value to 0%, the less the difference between the predicted and actual values.

sharply (18:00 to 24:00 and 00:00 to 07:00), all four models generate forecasting results with high accuracy. The reason for this phenomenon is that some hidden statistical information and correlations in the forecasted data set cannot be revealed. It is thought that none of the traffic flow predictors is able to learn well from the input except for the proposed SAE-LM model, which reconstructs the observed traffic data through an encoding and decoding process to predict traffic flow.

We compare the percentage error of the traffic flow forecasting results of each model, as illustrated in Fig. 5. As can be seen in Fig. 5(a) and (b), the percentage error value of the SAE-LM model is closer to 0% compared with those of the other models in almost all time periods. Only in a few time periods, especially those shown by the ovals in Fig. 5(b) (12:00 to 16:00 on March 26–28, 2014), the SAE-LM model has the worst percentage error result of all four traffic predictors. The raw traffic data in *R3b* shows a highly smooth distribution between 08:00 to 12:00 on March 26–28, 2014. Since the SAE-LM model uses a 240-min sampling time to predict the traffic flow, we can infer that the SAE-LM model may not be suitable for forecasting the traffic flow using data samples with a highly smooth distribution. However, the results described in the previous paragraph show that the SAE-LM model is effective in dealing with lumpy data. In fact, the actual traffic flow data is always lumpy. Therefore, based on the results in Figs. 4 and 5 and Table IV, the SAE-LM model is demonstrated to be more effective and promising for traffic flow prediction in practice than the other models.

In terms of efficiency concerns [39], [40], the computational time is also an important factor in traffic flow forecasting. We measure the computational time on a 3.40-GHz Intel i7 CPU, 64-b operating system, and 16.0-GM RAM. The computational time of each traffic predictor in the training and forecasting stages is shown in Table V. The RBFNN is the fastest compared with the other predictors in both training and forecasting. However, although the SAE-LM consumes slightly more time (2.25 s in the training stage and 0.53 s in the forecasting stage), it produces more accurate results than RBFNN (average increase of 7.33% for *R3a* and 6.39% for *R3b*).

TABLE V
COMPUTATIONAL TIME COMPARISON IN SECONDS

Traffic forecasting model	Training (Seconds)	Forecasting (Seconds)	Total (Seconds)
SAE-LM	5.98	1.25	7.23
EXP-LM	3.79	0.74	4.53
PSONN	5.39	1.09	6.48
RBFNN	**3.73**	**0.72**	**4.45**

Four hours of forecasted traffic flow.

V. CONCLUSION

In recent times, research has focused on developing and optimizing traffic flow predictors in order to obtain more accurate predictions and solve traffic congestion problems based on many computational models. However, most of them are shallow traffic models and the computed traffic flow forecasting results need to be more accurate. Therefore, this paper proposes a novel predictor, SAE-LM, with an optimized structure. Due to the fact that the trial-and-error method is very time consuming with too many design factors involved, the Taguchi method is adopted to improve the effectiveness of the design of traffic flow forecasting model. The proposed SAE-LM with an optimized structure is applied to real-world data collected from the M6 freeway in the U.K. and compared with the existing traffic predictors, EXP-LM, PSONN, and RBFNN. The evaluation results indicate that the SAE-LM model with an optimized structure is an accurate and efficient approach to traffic flow forecasting. In addition, it has superior performance (about 90% accuracy rate) in traffic flow forecasting and is the most suitable approach to deal with the lumpy data in this research. Some significant findings include the following.

1) Although the accuracy of traffic flow forecasting may be increased by applying more autoencoders, the time consumed in the training and forecasting stages increases dramatically. With limited computational time, the SAE-LM model with five hidden layers (four autoencoders) can generate the most accurate predicted results.
2) The SAE-LM is able to generate forecasting results with approximately 90% accuracy rate, which is 5% more than the other predictors. In addition, the prediction

performance of the SAE-LM is relatively stationary (only 1.6% difference), which is 0.93% better than the EXP-LM, 2.11% better than the PSONN, and 0.82% better than the RBFNN.

3) The disadvantage of the SAE-LM model is that it may not be able to perform well if the observed traffic data has a highly smooth distribution. However, when lumpy traffic data are collected, the SAE-LM has superior performance in traffic flow forecasting. Since the observed traffic flow data are always lumpy in nature, the SAE-LM model is demonstrated to be more effective and promising for traffic flow prediction in practice than the other models.

Future research will focus on the following areas: improving the accuracy and efficiency of the SAE-LM, adopting other learning algorithms to the last hidden set, selecting the appropriate imputation strategy to deal with missing data, and solving congestion problems that occur in complex roadways.

REFERENCES

[1] M. E. Ben-Akiva, S. Gao, Z. Wei, and Y. Wen, "A dynamic traffic assignment model for highly congested urban networks," *Transp. Res. C, Emerg. Technol.*, vol. 24, pp. 62–82, Oct. 2012.

[2] A. Hamilton, B. Waterson, T. Cherrett, A. Robinson, and I. Snell, "The evolution of urban traffic control: Changing policy and technology," *Transp. Planning Technol.*, vol. 36, no. 1, pp. 24–43, 2013.

[3] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[4] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[5] K. Y. Chan, S. Khadem, T. S. Dillon, V. Palade, J. Singh, and E. Chang, "Selection of significant on-road sensor data for short-term traffic flow forecasting using the Taguchi method," *IEEE Trans. Ind. Informat.*, vol. 8, no. 2, pp. 255–266, May 2012.

[6] J. Nahar, Y.-P. P. Chen, and S. Ali, "Kernel-based naive Bayes classifier for breast cancer prediction," *J. Biol. Syst.*, vol. 15, no. 1, pp. 17–25, 2007.

[7] D. W. Tjondronegoro and Y.-P. P. Chen, "Knowledge-discounted event detection in sports video," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 5, pp. 1009–1024, Sep. 2010.

[8] A. Stathopoulos, M. Karlaftis, and L. Dimitriou, "Fuzzy rule-based system approach to combining traffic count forecasts," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2183, pp. 120–128, Dec. 2010.

[9] K. Y. Chan, T. Dillon, E. Chang, and J. Singh, "Prediction of short-term traffic variables using intelligent swarm-based neural networks," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 1, pp. 263–274, Jan. 2013.

[10] S. Kalyaanamoorthy and Y.-P. P. Chen, "Quantum polarized ligand docking investigation to understand the significance of protonation states in histone deacetylase inhibitors," *J. Molecular Graph. Model.*, vol. 44, pp. 44–53, Jul. 2013.

[11] T. Lillesand, R. W. Kiefer, and J. Chipman, *Remote Sensing and Image Interpretation*. New York, NY, USA: Wiley, 2014.

[12] P. Gibson, *Introductory Remote Sensing Principles and Concepts*. Evanston, IL, USA: Routledge, 2013.

[13] J. N. Liu, Y. Hu, Y. He, P. W. Chan, and L. Lai, "Deep neural network modeling for big data weather forecasting," in *Information Granularity, Big Data, and Computational Intelligence*. Switzerland: Springer, 2015, pp. 389–408.

[14] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[15] H.-C. Shin, M. R. Orton, D. J. Collins, S. J. Doran, and M. O. Leach, "Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1930–1943, Aug. 2013.

[16] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 1562–1566.

[17] Y. Wang, M. Papageorgiou, and A. Messmer, "A real-time freeway network traffic surveillance tool," *IEEE Trans. Control Syst. Technol.*, vol. 14, no. 1, pp. 18–32, Jan. 2006.

[18] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach," *Transp. Res. C, Emerg. Technol.*, vol. 13, no. 3, pp. 211–234, 2005.

[19] E. I. Vlahogianni and M. G. Karlaftis, "Testing and comparing neural network and statistical approaches for predicting transportation time series," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2399, pp. 9–22, Dec. 2013.

[20] B. M. Wilamowski, "Neural network architectures and learning algorithms," *IEEE Ind. Electron. Mag.*, vol. 3, no. 4, pp. 56–63, Dec. 2009.

[21] 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.

[22] K. Y. Chan and C. K. F. Yiu, "Development of neural network based traffic flow predictors using pre-processed data," in *Optimization and Control Methods in Industrial Engineering and Construction*. Dordrecht, The Netherlands: Springer, 2014, pp. 125–138.

[23] B. M. Wilamowski and H. Yu, "Improved computation for Levenberg–Marquardt training," *IEEE Trans. Neural Netw.*, vol. 21, no. 6, pp. 930–937, Jun. 2010.

[24] G.-C. Chen and J.-S. Yu, "Particle swarm optimization algorithm," *Inf. Control*, vol. 34, no. 3, p. 318, 2005.

[25] I. Okutani and Y. J. Stephanedes, "Dynamic prediction of traffic volume through Kalman filtering theory," *Transp. Res. B, Methodol.*, vol. 18, no. 1, pp. 1–11, 1984.

[26] S.-H. Ling, F. H. F. Leung, H. K. Lam, Y.-S. Lee, and P. K. S. Tam, "A novel genetic-algorithm-based neural network for short-term load forecasting," *IEEE Trans. Ind. Electron.*, vol. 50, no. 4, pp. 793–799, Aug. 2003.

[27] J. Z. Zhu, J. X. Cao, and Y. Zhu, "Traffic volume forecasting based on radial basis function neural network with the consideration of traffic flows at the adjacent intersections," *Transp. Res. C, Emerg. Technol.*, vol. 47, pp. 139–154, Oct. 2014.

[28] H.-G. Han and J.-F. Qiao, "Adaptive computation algorithm for RBF neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 342–347, Feb. 2012.

[29] G. Taguchi and Y. Yokoyama, *Taguchi Methods: Design of Experiments* (Taguchi Methods Series), vol. 4. Amer. Supplier Inst., Nov. 1993.

[30] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19. 2007, p. 153.

[31] Y.-L. Boureau and Y. L. Cun, "Sparse feature learning for deep belief networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1185–1192.

[32] A. Ng, "Sparse autoencoder," *CS294A Lect. Notes*, vol. 72, pp. 1–19, 2011.

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

[34] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, Jun. 2014.

[35] N. Wanas, G. Auda, M. S. Kamel, and F. Karray, "On the optimal number of hidden nodes in a neural network," in *Proc. IEEE Can. Conf. Elect. Comput. Eng.*, May 1998, pp. 918–921.

[36] B. Choi, J.-H. Lee, and D.-H. Kim, "Solving local minima problem with large number of hidden nodes on two-layered feed-forward artificial neural networks," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3640–3643, 2008.

[37] A. Nagare and S. Bhatia, "Traffic flow control using neural network," *Traffic*, vol. 1, no. 2, pp. 50–52, Jan. 2012.

[38] M. S. Phadke, *Quality Engineering Using Robust Design*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995.

[39] C. Yan *et al.*, "Efficient parallel framework for HEVC motion estimation on many-core processors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 12, pp. 2077–2089, Dec. 2014.

[40] C. Yan *et al.*, "A highly parallel framework for HEVC coding unit partitioning tree decision on many-core processors," *IEEE Signal Process. Lett.*, vol. 21, no. 5, pp. 573–576, May 2014.

Hao-Fan Yang received the master's degree in information technology (software architecture) from the Queensland University of Technology, Brisbane, QLD, Australia, in 2010. He is currently pursuing the Ph.D. degree with La Trobe University, Melbourne, VIC, Australia.

He has been undertaking research on data mining to more efficiently improve the performance of prediction models in many areas. His current research interests include short-term traffic forecasting and real-time congestion management.

Tharam S. Dillon (M'74–SM'87–F'98–LF'10) received the Ph.D. degree in electrical and computer systems engineering from Monash University, Melbourne, VIC, Australia, in 1974.

He has published over 750 papers in international conferences and journals. He has authored five books and edited five books. His current research interests include Web semantics, ontologies, Internet computing, e-commerce, hybrid neurosymbolic systems, neural nets, software engineering, database systems, and data mining.

Prof. Dillon is a fellow of the Australian Computer Society and Institution of Engineers in Australia. He is the Editor-in-Chief of the *International Journal of Computer Systems Science and Engineering* and *Engineering Intelligent Systems*. He is the Head of the International Federation for Information Processing (IFIP) International Task Force WG2.12/24 on Semantic Web and Web Semantics, and the Chairman of the IFIP WG12.9 on Computational Intelligence, the IEEE Industrial Electronics Society Technical Committee on Industrial Informatics, and the IFIP Technical Committee 12 on Artificial Intelligence.

Yi-Ping Phoebe Chen (M'04–SM'07) received the B.Inf.Tech. (Hons.) and Ph.D. degrees in computer science (bioinformatics) from the University of Queensland, Brisbane, QLD, Australia.

She is currently a Professor and the Chair and Director of Research with the Department of Computer Science and Computer Engineering, La Trobe University, Melbourne, VIC, Australia. She is the Chief Investigator with the ARC Centre of Excellence in Bioinformatics, Brisbane. She is involved in knowledge discovery technologies, and is especially interested in their application to genomics and biomedical science. She has been involved in the areas of bioinformatics, health informatics, multimedia databases, query systems, and systems biology. She has co-authored more than 200 research papers, with many published in top journals and conferences. Her current research interests include to find the best solutions for mining, integrating, and analyzing complex data structures and functions for scientific and biomedical applications.

Prof. Chen is the Steering Committee Chair of the Asia-Pacific Bioinformatics Conference (Founder) and the International Conference on Multimedia Modelling. She has been on the program committees of more than 100 international conferences, including top ranking conferences, such as ICDE, ICPR, ISMB, and CIKM.