Deep Learning-based Network Traffic Prediction for Secure Backbone Networks in Internet of Vehicles

XIAOJIE WANG, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.

LAISEN NIE*, School of Electronics and Information, Northwestern Polytechnical University, Xi'an, 710072, China.

ZHAOLONG NING*, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.

LEI GUO, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.

GUOYIN WANG, Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.

XINBO GAO, Chongqing Key Laboratory of Image Cognition, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.

NEERAJ KUMAR, Department of Computer Science and Engineering Thapar Institute of Engineering and Technology Patiala, India.

Internet of Vehicles (IoV), as a special application of Internet of Things (IoT), has been widely used for Intelligent Transportation System (ITS), which leads to complex and heterogeneous IoV backbone networks. Network traffic prediction techniques are crucial for efficient and secure network management, such as routing algorithm, network planning, anomaly and intrusion detection. This paper studies the problem of end-to-end network traffic prediction in IoV backbone networks, and proposes a deep learning-based method. The constructed system considers the spatio-temporal feature of network traffic, and can capture the long-range dependence of network traffic. Furthermore, a threshold-based update mechanism is put forward to improve the real-time performance of the designed method by using Q-learning. The effectiveness of the proposed method is evaluated by a real network traffic data set.

Authors' addresses: Xiaojie Wang, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.; Laisen Nie, nielaisen@nwpu.edu.cn, School of Electronics and Information, Northwestern Polytechnical University, Xi'an, 710072, China.; Zhaolong Ning, zhaolongning@gmail.com, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.; Lei Guo, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.; Guoyin Wang, Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.; Xinbo Gao, Chongqing Key Laboratory of Image Cognition, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.; Neeraj Kumar, Department of Computer Science and Engineering Thapar Institute of Engineering and Technology Patiala, India.

Manuscript submitted to ACM

Additional Key Words and Phrases: Internet of vehicles, traffic prediction, network security, deep learning

ACM Reference Format:
Xiaojie Wang, Laisen Nie, Zhaolong Ning, Lei Guo, Guoyin Wang, Xinbo Gao, and Neeraj Kumar. 2020. Deep Learning-based Network Traffic Prediction for Secure Backbone Networks in Internet of Vehicles. *ACM Trans. Internet Technol.* 1, 1, Article 1 (January 2020), 21 pages. https://doi.org/10.1145/3433548

1 INTRODUCTION

Cloud-based Internet of Vehicles (IoV), integrated by Internet of Things (IoT) and cloud computing, has been rapidly developed for Intelligent Transportation System (ITS) [6, 11, 16, 19, 23, 26, 29]. IoVs can provide reliable communications among vehicles for safety-related applications, and the network access to infrastructures and pedestrians. IoV backbone networks transmit the traffic aggregated by end-to-end network traffic generated from a large number of sensors and safety monitoring systems of On-Board Units (OBUs) [9], as shown in Fig. 1. Under this case, the IoV backbone network has become much more complex and heterogeneous. As a crucial component of an IoV, the network security and management system is necessary to provide reliable data transmissions for secure IoVs. Network management operations usually collect network states at first, and then make decisions for network management according to the reported data [5, 12, 28, 33]. End-to-end network traffic is a crucial network state, which is the basis for network planning and predictive routing [2, 31].

With the development of IoVs, more data have been generated for implementing ITS. It arises the challenges in network security. For instance, an attacker may intrude an IoV, and it can steal the information related to users' privacy leading to a tremendous economic loss for users. In this case, many network security applications, such as Intrusion Detection Systems (IDS), have been deployed in the IoV backbone network to prevent it from kinds of threats. Besides, anomaly detection techniques are leveraged to implement anomaly-based intrusion detection, and detect impossible damages in networks [8, 13, 30]. In practice, both intrusion and anomaly detection systems need end-to-end network traffic data as an input parameter to carry out the corresponding functions. For instance, an IDS can identify the Distributed Denial of Service attack (DDoS) by detecting the anomalous behaviors of end-to-end network traffic.

Thereby, precise network traffic prediction is useful for maintaining IoVs. A Traffic Matrix (TM), which shows the size of network traffic among Origin-Destination (OD) nodes, is the mathematical representation of network traffic [20]. In a TM, each row vector reveals the time-varying features of the corresponding OD flow. A TM obeys manifold statistical features, e.g., spatial, temporal and spatio-temporal features. These intricate statistical features arise the main challenges in network traffic prediction. Initially, the statistics-based methods, such as Gaussian and Poisson models, are proposed to predict network traffic. The statistical features of TM are much more complex, while supporting more services in IoVs. Then, Machine Learning (ML)-based methods emerge for network traffic prediction, such as Principal Component Analysis (PCA) [24], Convolutional Neural Network (CNN) [32] and Long Short-Term Memory (LSTM) methods [22]. These methods capture several statistical features jointly to decease the network traffic prediction error. Nevertheless, they are not appropriate for predicting network traffic in IoVs. The corresponding main challenges can be summarized as follows [7, 10, 25, 27]:

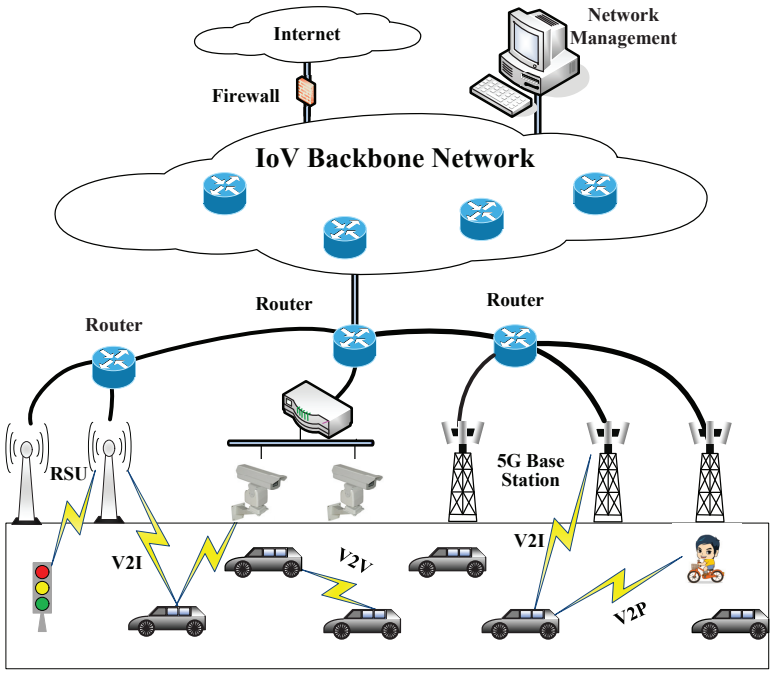

Fig. 1. An illustration of IoV backbone networks.

- The statistical features of network traffic in secure IoVs obey a superposition of various distributions. There are kinds of data in a secure IoV leading to complex network traffic statistical features. These complex statistical features result in unfaithful predictors via previous methods.
- The ML-based methods cannot meet the commands in real-time performance. For a secure IoV, the topology changes frequently due to quick movement of vehicles. Meanwhile, a secure IoV needs fast responses for any unlawful attacks. Hence, the real-time performance of network traffic prediction is crucial for secure IoVs. Nevertheless, the ML-based methods usually extract the features of network traffic by collecting many data samples. Meanwhile, to guarantee prediction accuracy, deep architectures are often updated persistently. Training these deep architectures by massive data is time-consuming for network resources (e.g., computation and communication recourses).
- It is significantly difficult for existing ML-based methods to exact the features of network traffic using few traffic samples. With limited communication resources, sampling and transmitting massive training data are uneconomical for secure IoVs.

Motivated by the above challenges, we investigate the network traffic prediction problem, and design a network traffic prediction system for secure IoVs. The proposed system uses CNN and LSTM to optimize the accuracy and real-time performance of network traffic prediction by extracting multiple statistical features of TM jointly and implementing intermittent model updates. To extract the multiple statistical features of TM jointly, a deep architecture integrating CNN and LSTM is built, in which CNN and LSTM are employed to track the spatio-temporal and time-varying features of TM. LSTM is a particular paradigm of

Recurrent Neural Network (RNN), which is able to capture the relationship between several input elements by the feedbacks within the same layer. Based on this advantage, we explore the time-varying features of network traffic through LSTM. Moreover, the TM consisting of all the OD flows, expresses spatial and spatio-temporal features caused by routing algorithms and network configurations. Hence, we take advantage of CNN to portray the spatio-temporal features of TM. To obtain the best predictors, each predictor is calculated by implementing a training in practice. However, each training of the designed deep architecture is expensive for real-time performance. Thereby, we train the deep architecture discontinuously. We propose a threshold-based mechanism to calculate the time intervals between two training processes [17].

The following summarizes the main contributions of our work:

- To predict network traffic of a secure IoV backbone network accurately, we design a deep architecture utilizing CNN and LSTM to trace the temporal and spatio-temporal features of TM jointly.
- To improve the real-time performance of the proposed method, a threshold-based mechanism is proposed to update the deep architecture discretely. A Reinforcement Learning (RL) algorithm is proposed to determine the threshold.
- We evaluate the designed scheme by real network traffic data set sampled from the Abilene network and our constructed testbed which is leveraged to imitate the scene of a secure IoV. According to the evaluation, our method can capture the long-term network traffic in secure IoVs.

The following sections are organized as follows. The related work is reviewed in Section 2. Section 3 provides the system model of network traffic prediction. Section 4 presents the proposed method, and Section 5 gives the evaluation of the designed deep architecture. At last, we conclude our work and illustrate the future work in Section 6.

2 RELATED WORK

2.1 Shallow Learning-based Methods

Initially, researchers model OD traffic by simple statistical models, e.g., Gaussian and Poisson models, to extract network traffic features. Then, the problem of network traffic prediction is converted into a parameter estimation problem. However, simple models cannot model nonlinear distributions of network traffic [1, 4, 21, 24].

ML techniques have a widespread usage in network traffic prediction, which relies on its remarkable ability in modeling complex statistical distributions. Initially, kinds of shallow learning-based approaches have emerged to solve network traffic prediction problem, such as shallow Neural Network (NN) and PCA. These methods try to fit the statistical features of network traffic consisting of linear and nonlinear features. According to the analysis of network traffic in depth, current network traffic yields multiple distributions, such as multi-fractal and heavy-tailed distributions. Capturing these statistical features can enhance the accuracy of network traffic prediction remarkably. However, traditional shallow learning approaches cannot track these features effectively. For instance, the PCA method deals with the problem of network traffic prediction by singular value decomposition, and neglects some non-principal components with small singular values. Namely, it relies on the spatial correlation of TM. The NN method is mainly designed to adapt to the changes of TM with respect to the time interval.

To enhance the accuracy of network traffic prediction, hybrid model-based methods have been emerged, in which researchers model network traffic by more than two models. In [1], the authors proposed two hybrid methods based on Fanout Estimation (FE) and Iterative Proportional Fitting (IPF), respectively. The IPF method is adapted to track the time-varying features of OD network traffic. By contrast, the FE method can extract the spatial features of TM. They are combined with other state-of-the-art modeling techniques for network traffic feature extraction, i.e., tomogravity, entropy maximization and shallow NN. Besides, the authors in [18] proposed a network traffic feature extraction method using artificial NN and genetic algorithm. They first model OD flows by the autoregressive model with exogenous inputs, and then calculate the inputs (i.e., weights and biases) by genetic algorithm.

2.2 Deep Learning-base Network Traffic Prediction

Deep Learning (DL) techniques are viewed as an excellent solution to fit a complex data set. Hence, it has been referred to extract network traffic features for both estimation and prediction. For instance, a hybrid deep architecture was proposed to leverage the diversified features for network traffic prediction in cellular networks in [27]. In this deep architecture, autoencoder and LSTM are used to extract spatial and temporal features, respectively. The authors in [32] handled this kind of prediction problem in future cellular networks. A CNN-based mechanism is designed to profile the nonlinear features of traffic in wireless networks. Different from previous methods denoting OD flows by sequences, the authors defined an OD flow as an image (i.e., a matrix). From this representation of an OD flow, they design a CNN-based deep architecture with highly dense connections for network traffic prediction. In [22], the authors proposed a LSTM-based deep architecture with one hidden layer consisting of four LSTM blocks to predict the network traffic of the optical data center.

Many network traffic prediction approaches have been proposed in traditional wired and wireless networks. However, the approaches for IoV backbone network traffic prediction have not been discussed, to the best of our knowledge. Though many approaches have been put forward for traffic prediction, they are not adopted to predict the network traffic of IoV backbone networks. Consequently, we design a deep architecture by integrating CNN and LSTM to extract the spatio-temporal and temporal features of TM for decreasing prediction errors.

3 SYSTEM MODEL

As mentioned before, TM reveals the volume of traffic that flows among all OD pairs. In this paper, we denote TM by $\boldsymbol{X}$ for a secure IoV backbone network. Its element is $X_{n,t}$, where n is the index of an OD flow and t denotes time slot. Generally, t is a time interval, and then $X_{n,t}$ shows the average traffic within time interval t. If the network is made up of N nodes and K links, then $n = 1, 2, 3, ..., N^2$. Moreover, if TM contains T time slots of network traffic in a secure IoV backbone network, then $t = 1, 2, 3, ..., T$. Link load expresses the aggregation of OD network traffic, and it conforms a linear relationship with respect to TM. This relationship can be denoted by:

$$\boldsymbol{Y} = \boldsymbol{R}\boldsymbol{X}, \tag{1}$$

where $\boldsymbol{R}$ is a so-called routing matrix. The routing matrix is constructed by 0 and 1, if we consider that an OD flow is transmitted through the same path of an IoV backbone network. The element in $\boldsymbol{R}$ is denoted

by $R_{k,n}$. When $R_{k,n} = 1$, the nth OD flow passes link k ($k = 1, 2, ..., K$), otherwise $R_{k,n} = 0$. Obviously, the sizes of link load and routing matrices are $K \times T$ and $K \times N^2$, respectively. TM estimation techniques calculate TM $\boldsymbol{X}$ via $\boldsymbol{Y}$ and $\boldsymbol{R}$. Symbols $\boldsymbol{Y}$ and $\boldsymbol{R}$ are available, because they can be obtained from the simple network management protocol and routing configurations, respectively. In this paper, we utilize link load and routing matrix to build a threshold for deep architecture update.

The problem of network traffic prediction in an IoV backbone network is to calculate X_t according to previous network traffic $(X_{t-1}, X_{t-2}, ..., X_{t-E})$, which can be denoted by:

$$X_{n,t}{=}f\left(X_{n,t-1}, X_{n,t-2}, ..., X_{n,t-E}\right). \tag{2}$$

Then this problem is defined as fitting function $f(\cdot)$ with respect to $X_{t-1}, X_{t-2}, ..., X_{t-E}$. Eq. (2) mainly shows the problem of network traffic prediction according to a sequence. Namely, fitting this function mainly considers the temporal features of OD flows. By expanding to other features, the network traffic prediction problem can be defined as:

$$X_{n,t}{=}f\left(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-2}, ..., \boldsymbol{X}_{t-E}\right), \tag{3}$$

where vector $\boldsymbol{X}_{t-e}$ ($e = 1, 2, ..., E$) is a snapshot of network traffic over time slot $t - e$. Symbol E is the length of prior network traffic. Different from the problem of network traffic prediction shown by Eq. (2), we consider the spatial and spatio-temporal features of network traffic. In this case, the problem of network traffic prediction is modeled by calculating a traffic element according to previous snapshots of network traffic.

4 OUR METHODOLOGY

Traditional prediction approaches usually calculate network traffic by extracting the time-varying features of OD flows to fit the function in Eq. (2). Different from previous approaches, we combine CNN and LSTM, and design a deep architecture to predict the network traffic of secure IoVs by means of extracting two features, i.e., the spatio-temporal and temporal features. The designed deep architecture for network traffic prediction is shown in Fig. 2. It contains 6 hidden layers,i.e., a convolutional layer, a subsampling layer, an LSTM layer, two fully connected layers and a dropout layer. In our method, we first preprocess the network traffic data set to normalize and centralize it, that is:

$$X_{n,t} = \frac{X_{n,t} - \mu_n}{|X_{n,t}|}, \tag{4}$$

where μ_n is the average value of OD flow n, and can be computed by the prior of network traffic.

4.1 Traffic Prediction Based on CNN and LSTM

CNN is a prevalent tool for 2-dimensional data feature extraction, such as pattern recognition and image processing [14, 32]. It can extract the spatio-temporal features of 2-dimensional data by using convolution kernels. For defining spatial location (i, j), $i \in \left\{I^{(l)}\right\}$ and $j \in \left\{J^{(l)}\right\}$ are the sets of input and output on layer l, respectively, and the output map of the lth convolutional layer can be denoted by:

$$\boldsymbol{a}_j^{(l)} = \sigma\left(\sum_i \left(\boldsymbol{a}_i^{(l-1)} * \boldsymbol{k}_j^{(l)}\right) + b_j^{(l)}\right), \tag{5}$$

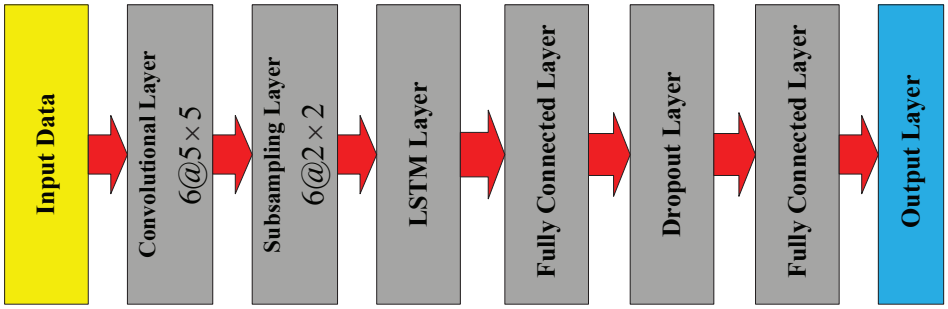

Fig. 2. Deep architecture with CNN and LSTM.

where $b_j^{(l)}$ and $\boldsymbol{k}_j^{(l)}$ are the jth convolution kernel and its bias of the lth convolutional layer, respectively. Variable $\boldsymbol{a}_i^{(l-1)}$ is the output map of the $(l-1)$th layer, and function $\sigma(\cdot)$ is the activation function.

As mentioned before, CNN is suitable for handling 2-dimensional data. Thereby, we utilize it to carry out the TM-oriented spatio-temporal feature extraction. In our deep architecture shown in Fig. 2, the convolutional layer, following the input layer, consists of 6 convolution kernels which learn 5×5 spatial dimensions. To predict network traffic, the tanh function is used as the activation function on the convolutional layer. Then, the output maps are:

$$\boldsymbol{a}_j^{(1)} = tanh\left(\left(\boldsymbol{X}^{(m)} * \boldsymbol{k}_j^{(1)}\right) + b_j^{(1)}\right), \quad j \in \{6\}, \tag{6}$$

where $\boldsymbol{X}^{(m)}$ is the training data set constructed by the prior of TM, and set $\{6\}$ denotes the set $\{1, 2, ..., 6\}$. Following the convolutional layer, a subsampling layer is built to carry out the average pooling with a factor of 2. Then, the output maps of this average pooling are:

$$\boldsymbol{a}_j^{(2)} = average\left(\boldsymbol{a}_j^{(1)}\right), \quad j \in \{6\}, \tag{7}$$

where $average(\cdot)$ denotes the average pooling process.

As a special paradigm of RNN, LSTM can learn the time-varying features of a time sequence [30, 34]. Hence, many LSTM-based algorithms have been put forward to predict a time sequence. The mathematic model of RNN can be defined by:

$$\begin{cases} f_t = \sigma\left(UX_t + Wf_{t-1} + b^f\right), \\ o_t = Vf_t + c^f, \\ a_t = \sigma\left(o_t\right), \end{cases} \tag{8}$$

where X_t is input data, and U, V and W are weights of RNN. Variables b^f and c^f are biases, and a_t is the final output map of RNN. LSTM takes advantage of three gates to control the contents of unit state. They are the forget gate, the input gate and the output gate, whose weights are denoted by W^f, W^i and W^o, respectively , and they can be denoted by:

$$\begin{cases} g_t = \sigma\left(W^f\left[f_{t-1}; X_t\right] + b^f\right), \\ i_t = \sigma\left(W^i\left[f_{t-1}; X_t\right] + b^i\right), \\ o_t = \sigma\left(W^o\left[f_{t-1}; X_t\right] + b^o\right), \end{cases} \tag{9}$$

where $[\cdot;\cdot]$ denotes the combination of two matrices. Variables b^f, b^i and b^o are the biases for three gates, respectively. Finally, the current cell state and the final output map of LSTM cell are:

$$\begin{cases} c_t = c_{t-1} g_t + i_t \tanh\left(W\left[f_{t-1}; X_t\right] + b^c\right), \\ a_t = o_t \tanh\left(c_t\right), \end{cases} \tag{10}$$

where c_{t-1} is the previous cell state.

In our method, we extract the spatio-temporal and time-varying features of TM by CNN and LSTM, respectively. As shown in Fig. 2, the output maps of subsampling layer are 6 matrices. To combine the LSTM layer, we unfold these matrices as a vector, and make it as the input of LSTM layer. After that, the output maps of LSTM can be shown as follows:

$$\begin{cases} \boldsymbol{g}_t = \sigma\left(W^f\left[f_{t-1}; \boldsymbol{a}_j^{(2)}\right] + b^f\right), \\ \boldsymbol{i}_t = \sigma\left(W^i\left[f_{t-1}; \boldsymbol{a}_j^{(2)}\right] + b^i\right), \\ \boldsymbol{o}_t = \sigma\left(W^o\left[f_{t-1}; \boldsymbol{a}_j^{(2)}\right] + b^o\right), \\ \boldsymbol{c}_t = \boldsymbol{c}_{t-1}\boldsymbol{g}_t + \boldsymbol{i}_t \tanh\left(W\left[f_{t-1}; \boldsymbol{a}_j^{(2)}\right] + b^c\right), \\ \boldsymbol{a}^{(3)} = \boldsymbol{o}_t \tanh\left(\boldsymbol{c}_t\right). \end{cases} \tag{11}$$

On the LSTM layer, there are $3\left(N^2-4\right)/2$ LSTM blocks for temporal feature extraction.

The rest of the proposed deep architecture contains two fully connected layers and one dropout layer employed between two fully connected layers. The first fully connected layer is made up of $3\left(N^2-4\right)/2$ neurons, and we also use the tanh function as the activation function for prediction. It can be denoted by:

$$a_i^{(4)} = \tanh\left(\boldsymbol{W}_i^{(4)}\boldsymbol{a}^{(3)} + b_i^{(4)}\right). \tag{12}$$

The following is the dropout layer used to prevent the proposed deep architecture from overfitting. The number of neurons on the dropout is the same as the first fully connected layer. It carries out a probabilistic dormancy mechanism for each neuron of the first fully connected layer. In other words, the output maps of the dropout layer can be denoted by:

$$a_i^{(5)} = d_i^{(5)} \tanh\left(\boldsymbol{W}_i^{(4)}\boldsymbol{a}^{(3)} + b_i^{(4)}\right), \tag{13}$$

where $d_i^{(5)}$ obeys a Bernoulli distribution $d_i^{(5)} \sim Bernoulli\,(p)$. The fully connected layer connecting with the output layer determines the length of predicted sequence. The objective of the deep architecture is to implement regression. Hence, to implement prediction, the tanh function is also utilized as activation function on this layer. The length of predicted sequence by way of a forward propagation is equal to the number of neurons on this fully connected layer.

To train the deep architecture, the backpropagation algorithm is applied to our approach. The loss function used in the backpropagation algorithm is the Mean-Squared-Error (MSE) of prediction, which is defined as:

$$loss_{MSE} = \frac{1}{I^{(L)}} \sum_{i=1}^{I^{(L)}} \left(h_i - X_{n,i}\right)^2, \tag{14}$$

where $I^{(L)}$ is the number of neurons on the output layer, and h_i is the output maps of the proposed deep architecture.

4.2 Deep Architecture Update Algorithm Based on RL

The prevalent end-to-end network traffic prediction approaches solve the problem shown by Eq. (2). In other words, OD flows are viewed as time sequences, and then they are predicted through their own prior information. In this paper, the deep architecture is designed to deal with the problem shown by Eq. (3). We use the previous network traffic of all OD flows as the prior information to predict the elements of an OD flow. This method can capture the weak relationship among nonadjacent elements that even come from different OD flows. Though it can ensure the accuracy of network traffic predictor, it actually arises a challenge in real-time performance. In detail, we employ E snapshots of TM to predict a time sequence, and then the prior data is an $N^2 \times E$ matrix. By contrast, the previous methods merely use an E-dimensional vector to predict a time sequence. Without other optimization methods, it is significantly difficult to employ the proposed deep architecture for online network traffic prediction.

For a network traffic prediction approach in the IoV backbone network, its real-time performance is crucial in practice. Obviously, training and updating the deep architecture after predicting a network traffic element is really expensive. On the contrary, predicting a large number of network traffic by one training may cause a poor predictor. Aiming at dealing with the tradeoff problem, we leverage RL to build a threshold to control the prediction error and real-time performance. The threshold is based on Normalized Mean Absolute Error (NMAE) between TM and link load, which is defined as:

$$NMAE = \frac{\sum\limits_{k,t} \left| \boldsymbol{R}\hat{\boldsymbol{X}} - \boldsymbol{Y} \right|}{\sum\limits_{k,t} \left| \boldsymbol{R}\hat{\boldsymbol{X}} \right|}, \tag{15}$$

where $\hat{\boldsymbol{X}}$ is the predictor of TM, and $\boldsymbol{Y}$ is the corresponding aggregation traffic (i.e., link load). Comparing with other metrics, NMAE can measure the spatial and temporal errors of network traffic prediction. Therefore, we use NMAE as the metric of the threshold-based mechanism. In our method, when the NMAE between a snapshot of link load and its predictor is larger than the threshold, and then we train the deep architecture. Otherwise, we predict traffic elements over the next time interval using previous trained deep architecture.

The threshold is computed by RL to obtain the optimal tradeoff between real-time performance and accuracy. RL has an excellent ability of dealing with the decision problem, in which it can compute an optimal policy with the maximum reward by implementing iterative actions. It leverages a sample of the environment to guide the following action from the current state to the next, and the relative reward can be gained in the meantime. Furthermore, the environment can be updated in the light of the obtained reward. The RL can be regarded as an MDP represented by $\langle S, A, \boldsymbol{P}, R, \gamma \rangle$. In detail, S and A are the state and action spaces, respectively. Symbol $\boldsymbol{P}$ is the transition probability matrix, and γ is the discount factor. R is the immediate reward from the current state to the next according to the given policy with environment.

To explore the optimal policy, many algorithms have been proposed. Recently, the Q-learning, known as an off-policy learning algorithm, has been brought in RL and widely developed. In Q-learning algorithm, when an agent with state s ($s \in S$) moves to state $\tilde{s}$ ($\tilde{s} \in S$) after carrying out action a ($a \in A$) according to given policy $\pi(s, a)$, immediate reward $R(s, a)$ can be obtained. The reward with respect to current state s by taking action a, denoted by $Q(s, a)$, is a weight sum of immediate reward $R(s, a)$ of moving to the next

state $\tilde{s}$, which can be denoted by:

$$Q(s,a) = R(s,a) + \gamma \left(\max_{\tilde{a} \in \tilde{A}} Q(\tilde{s}, \tilde{a}) \right), \tag{16}$$

where $\tilde{a} \in \tilde{A}$ is the action set under the next state $\tilde{s}$. The Q-learning algorithm can be regarded as an optimization problem that finds an optimal policy to maximize the reward $Q(s,a)$. For each iterative process of researching the maximum reward, the reward can be updated according to a weighted sum denoted by:

$$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha \left(R(s,a) + \gamma \left(\max_{\tilde{a} \in \tilde{A}} Q(\tilde{s}, \tilde{a}) \right) \right), \tag{17}$$

where $\alpha \in [0,1]$ is a constant step-size parameter. During each iteration, the sampling of state can be chosen by way of exploration-only and exploitation-only approaches. The former prefers to carry out sampling with large range in value, and determine a state sampling by the uniform distribution. By contrast, the latter selects the state sampling with the maximum cumulative reward currently.

The detail process of computing the threshold based on Q-learning is shown in Algorithm 1. The state s represents the error threshold. The state space is set according to previous prediction errors, i.e., it is equal to the maximum NMAE among previous predictors. We define the number of updates with respect to state s as $N(s)$. Similarly, the NMAE with respect to state s can be denoted by $NMAE(s)$. They can be obtained by counting the previous data set. Obviously, $N(s)$ and $NMAE(s)$ are monotone decreasing and increasing functions with respect to s, respectively. In practice, we hope that $NMAE(s)$ and $N(s)$ are as small as possible. Therefore, we define the initialized policy as follows:

$$\pi(s,a) = \frac{1}{NMAE(s) N(s)}. \tag{18}$$

By Algorithm 1, we can gain optimal policy $\pi^*(s,a)$, and then set the threshold according to $\pi^*(s,a)$.

Algorithm 1 Residual-based Dictionary Learning

Require: action space A
initial state s_0
discount factor γ
step-size parameter α
Ensure: $\pi^*(s,a)$
1: $Q(s,a) \leftarrow 0$
2: $\pi(s,a) \leftarrow \frac{1}{NMAE(s)N(s)}$
3: **for** $z = 1, 2, ..., Z$ **do**
4: Choose an action from S using ε-greedy algorithm
5: Obverse immediate reward $R(s,a)$ and $\tilde{s}$
6: Update $Q(s,a)$ according to Eq. (17)
7: $s \leftarrow \tilde{s}$
8: **end for**

Up to now, we have presented the proposed deep architecture to solve the traffic prediction problem and the corresponding optimization method to optimize the real-time performance. The details of prediction method are shown by Algorithm 2. We first construct the training data set from previous TM $\boldsymbol{X}'$ which is known for us. $\boldsymbol{X}^{(m)}$ is the mth sampling, and it is an $N^2 \times E$ matrix. Symbol $\boldsymbol{X}'_{*\,,\,(m:(m+E-1))}$ denotes

intercepting partial columns of $\boldsymbol{X}'$, i.e., from columns m to $m+E-1$. Similarly, $\boldsymbol{X}'_{n\ ,\ (m:(m+L_p-1))}$ means intercepting partial columns (columns m to $m+L_p-1$) from row n of $\boldsymbol{X}'$. Obviously, the output maps of convolutional and subsampling layers are $\left(N^2-4\right)\times(E-4)$ and $\frac{\left(N^2-4\right)}{2}\times\frac{(E-4)}{2}$ matrices. The unfolding layer reshapes 6 output maps of the subsampling layer as a sequence which is a $\left(\frac{3\left(N^2-4\right)(E-4)}{2}\right)\times 1$ vector. In each output map $\boldsymbol{a}_j^{(2)}$, it is unfolded as a row vector consisting of $\frac{\left(N^2-4\right)}{2}$ rows of $\boldsymbol{a}_j^{(2)}$ in turn. Aiming at each OD flow, we build the corresponding training data set for prediction. Then, we train the proposed deep architecture by the backpropagation algorithm independently for each OD flow prediction. The predictor can be achieved by forward propagation over trained deep architecture. After predicting all OD flows, we update the prior TM $\boldsymbol{X}'$ for the next prediction. Meanwhile, we can calculate the NMAE according to link load, routing information and predictor $\hat{\boldsymbol{X}}$. Besides, we compute a threshold by means of Algorithm 1. If the NAME is greater than or equal to the threshold, we train the deep architecture by updated prior TM for the network prediction. Otherwise, we implement the next prediction by the forward propagation without re-training the proposed deep architecture. This method is able to decease the number of training and optimize the real-time performance of prediction.

Algorithm 2 Network traffic prediction by way of DL and RL

Require: previous TM $\boldsymbol{X}'$ with M time slots
length of training data set E
length of predicted sequence L_p
Ensure: $\hat{\boldsymbol{X}}$
for $n=1,2,...,N^2$ **do**
for $m=1,2,...,(M-E-L_p)$ **do**
$\boldsymbol{X}^{(m)} \leftarrow \boldsymbol{X}'_{*\ ,\ (m:(m+E-1))}$
$\boldsymbol{X}_n^{(m)} \leftarrow \boldsymbol{X}'_{n\ ,\ \left(m:\left(m+L_p-1\right)\right)}$
Train the deep architecture shown in Fig. 2 by the training data $\left(\boldsymbol{X}^{(m)}, \boldsymbol{X}_n^{(m)}\right)$
Predict $\left(X_{n,M+1}, X_{n,M+1}, ..., X_{n,M+L_p}\right)$ from $\boldsymbol{X}'_{*\ ,\ ((M-E):M)}$ by forward propagation
end for
end for
Update previous TM by predicted TM $\hat{\boldsymbol{X}}$ with L_p time slots
Calculate NMAE according to $\hat{\boldsymbol{X}}$ by Eq. (15)
Calculate threshold by Algorithm 1
if $NAME \geq threshold$ **then**
Go to Step 1
else
Go to Step 7
end if

5 NUMERICAL RESULTS

5.1 Network Traffic Data Set

To evaluate the designed approach, we implement it by the real data set sampled via the Abilene network and our testbed [21]. The Abilene network includes 12 nodes, and these nodes are connected by 54 internal and external links. Hence, the number of OD flows is 144 in Abilene. We use one week of TM to evaluate

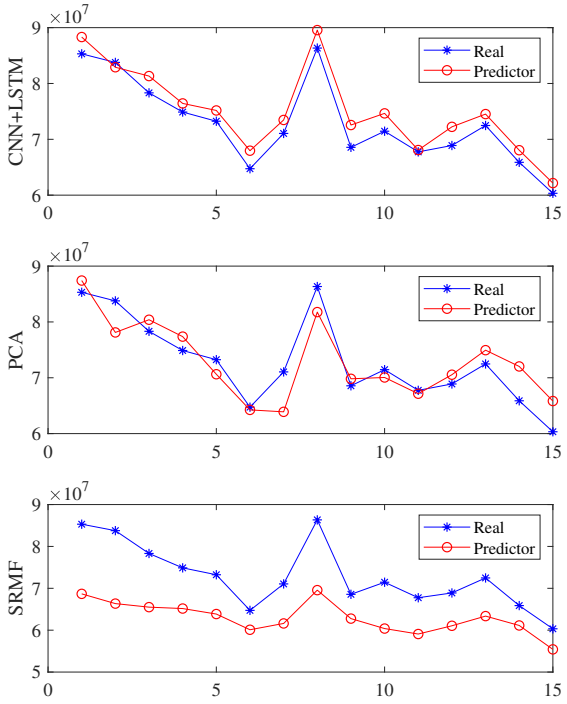

Fig. 3. Real network traffic and its predictor for large OD flow in Abilene.

the proposed approach. The network traffic is sampled by 10-minute interval. Namely, the TM contains 2016 time slots. Besides, we construct a testbed, which consists of 1 RSU and 12 OBUs, to imitate the scene of a secure IoV [15]. The topology of our testbed is built according to the open shortest path first algorithm, in which weights are defined as the general urban path loss model [3]. Meanwhile, the distance between two OBUs is random. Furthermore, the services provided by the testbed include video and radio. We also collect end-to-end network traffic with 2016 time slots by using Wireshark. Moreover, two state-of-the-art network traffic prediction approaches are leveraged for comparison, i.e., the PCA method and the Sparsity Regularized Matrix Factorization (SRMF) method. The PCA method is a ML-based approach, and leverages singular value decomposition to capture the spatial features of TM [24]. SRMF is a matrix interpolation algorithm in fact. It is put forward to reconstruct the missing data during network traffic direct measurement process. Besides, by setting special parameters, it is also an available network traffic prediction approach by extracting the spatio-temporal features of TM [21].

In our evaluations, the first 2000 time slots of TM are used as the previous TM to build the training data set, and the rest is for test. Furthermore, we set $M = 150$, $E = 50$, $L_p = 4$, $\alpha = 0.5$, $\gamma = 0.01$ in our evaluations. To train the proposed deep architecture, we set the batch equal to 2 and $p = 0.5$ for the Bernoulli distribution on the dropout layer. We set these parameters empirically to obtain the lowest prediction error. We use MATLAB R2019a to carry out all the evaluations. The evaluations are conducted on a 64-bit Windows 7 machine running on an Intel Xeon W-2102 (2.9 GHz) and a 32 GB RAM.

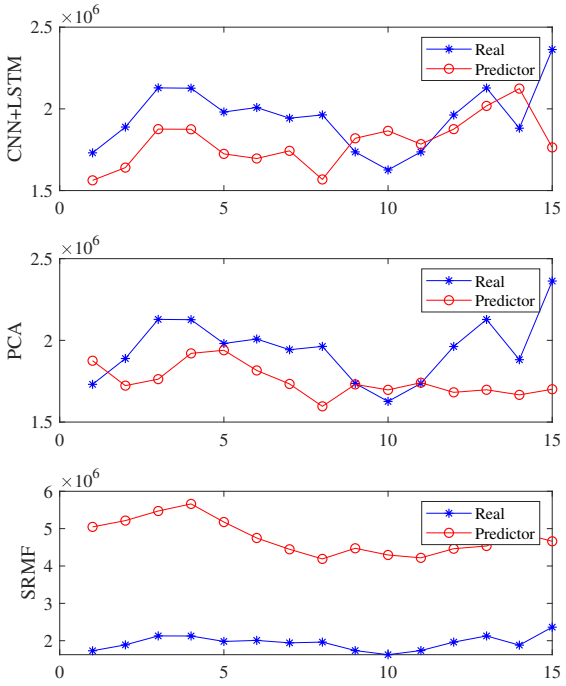

Fig. 4. Real network traffic and its predictor for small OD flow in Abilene.

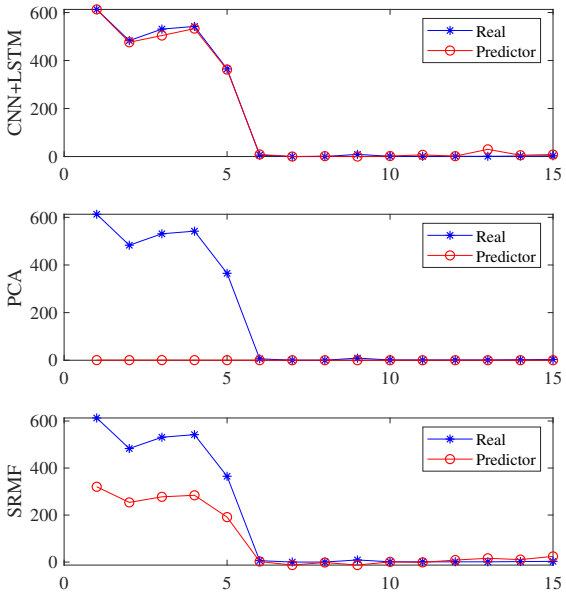

Fig. 5. Real network traffic and its predictor for large OD flow in testbed.

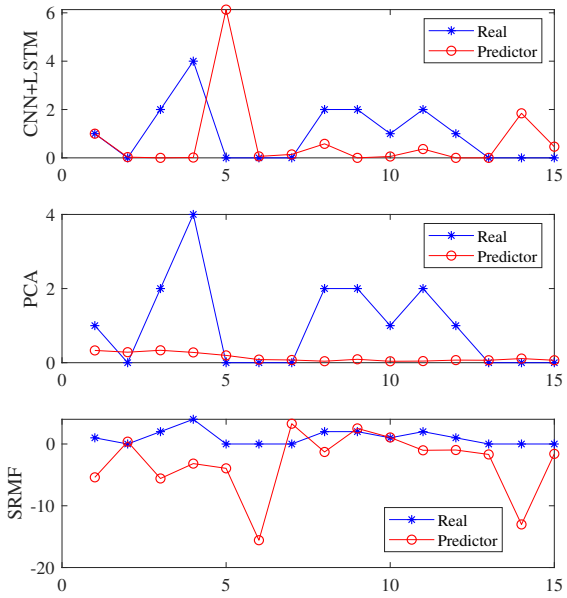

Fig. 6. Real network traffic and its predictor for small OD flow in testbed.

5.2 Prediction Error Evaluation

We first evaluate our approach in tracking the traces of OD flows. Hence, we demonstrate the real network traffic and its predictor as shown in Figs. 3 and 4, respectively. The x-axis and y-axis are time slot order and the volume of traffic element (i.e., the number of packets during a time slot), respectively. For a prediction algorithm, the elephant (large) flows are easier to be predicted. On the contrary, it is difficult to predict the mice (small) flows. Thereby, to guarantee the comprehensiveness of evaluation, we assess two OD flows with large and small averages of network traffic from 144 OD flows. Fig. 3 shows the real network traffic of large OD flow and its predictor in Abilene. Obviously, three approaches can faithfully capture the profile of this OD flow. PCA and our method (CNN+LSTM) appear over-estimate and under-estimate more or less. By contrast, SRMF exhibits a consistent under-estimate. For small OD flow shown in Fig. 4, we find that the prediction errors of three approaches are higher than that of large OD flow. For an approach extracting spatial or spatio-temporal features to predict network traffic, the prediction errors of small OD flows are usually influenced by large OD flows. The spatial feature of TM means that two adjacent elements that belong to different OD flows in TM are similar in size, and the definition of spatio-temporal feature is similar. Hence, SRMF shows obvious over-estimate in Fig. 4, though it still can track the profile of this OD flow. PCA and CNN+LSTM have lower errors. Nevertheless, PCA cannot capture the profile of this OD flow at all. For a network management function, it has a fault-tolerant ability for network traffic prediction error. Sometimes, predicting the profile of an OD flow are much more important for many network management functions. Thereby, SRMF is not a failure prediction. Figs. 5 and 6 show the predictors of our testbed. The traffic flows in our testbed have many irregular fluctuations, which raises the difficulty level of network traffic prediction. For instance, in Fig. 5, the traffic flow appears a sharp damping. Our approach can capture this

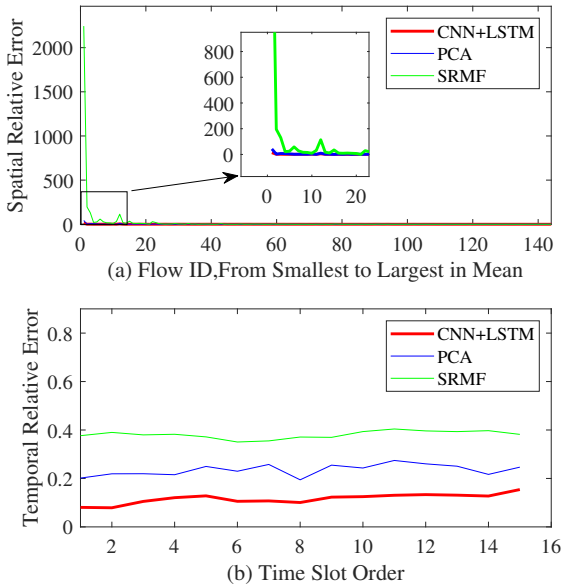

Fig. 7. Evaluation for SREs and TREs in Abilene.

sharp damping faithfully. By contrast, SRMF has a low error, and PCA cannot track this trace at all. For small traffic flows, all approaches occur large prediction error. Specially, SRMF has negative predictors, and PCA shows a consistent predictor. Our approach can pursue the profile of this traffic flow from time slots 6 to 13.

To provide a quantitative analysis of three approaches, we leverage the Spatial Relative Error (SRE) and Temporal Relative Error (TRE) defined as:

$$\begin{cases} SRE\,(n) = \frac{\left\|\hat{X}_{n,t} - X_{n,t}\right\|_2}{\left\|X_{n,t}\right\|_2} \\ TRE\,(t) = \frac{\left\|\hat{X}_{n,t} - X_{n,t}\right\|_2}{\left\|X_{n,t}\right\|_2} \end{cases}, \tag{19}$$

where $X_{n,t}$ and $\hat{X}_{n,t}$ are real network traffic and its predictor, respectively. Fig. 7 shows the SREs and TREs of three approaches. The x-axis in Fig. 7(a) is the index of each OD flow, and it is arranged according to the means of OD flows from the smallest to the largest ones. We can obtain similar conclusion for Figs. 3 and 4. SRMF has the largest SREs, specially for small OD flows. The means of SREs of CNN+LSTM, PCA, and SRMF are 0.47, 1.03 and 22.01, respectively. Fig. 7(b) exhibits the TREs of three approaches. Our method obtains the lowest TRE consistently. SRMF shows the largest TRE in three approaches. Fig. 8 displays the Cumulative Distribution Functions (CDF) of SREs and TREs. From Fig. 8(a), for CNN+LSTM, PCA and SRMF, the SREs are less than 1.63, 1.80 and 15.38 respectively, with respect to 90% of all the OD flows. Meanwhile, in Fig. 8(b), the TREs are less than 0.13, 0.25 and 0.39 respectively for 80% of time slots. Similarly, the means of SREs of three methods are 2.34, 6.63, and 4.92, as shown in Fig. 9. The TREs of our method occur some fluctuations, though it is the lowest one of three approaches. These fluctuations of TRE can be exhibited clearly in Fig. 10(b).

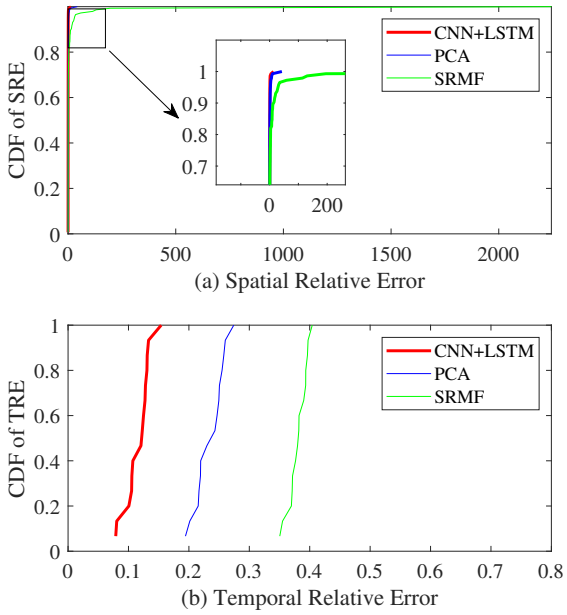

Fig. 8. Evaluation for CDF of SREs and TREs in Abilene.

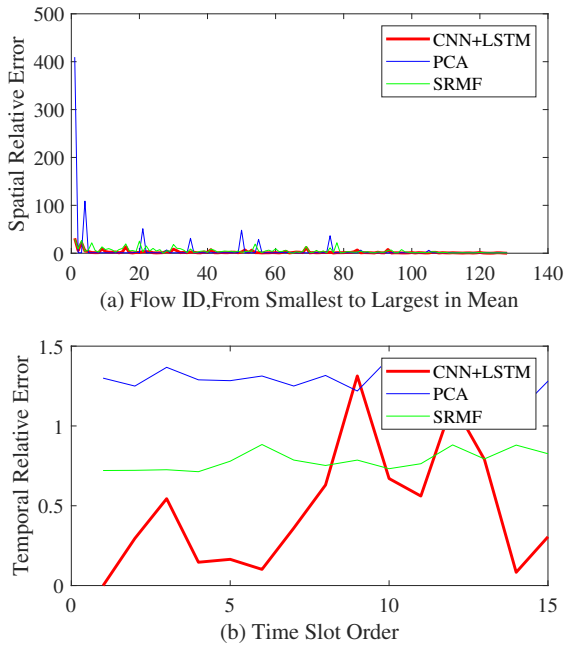

Fig. 9. Evaluation for SREs and TREs in testbed.

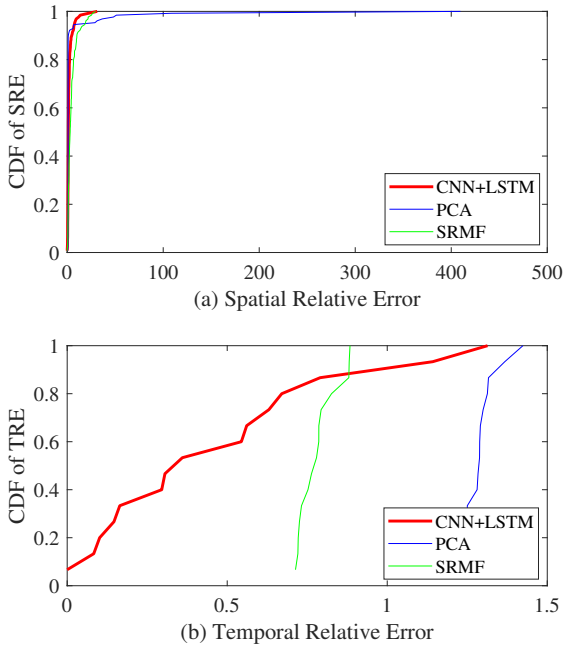

Fig. 10. Evaluation for CDF of SREs and TREs in testbed.

Moreover, we leverage the prediction bias and the relative sample Standard Deviation (SD) to evaluate the availability, which are denoted by:

$$\begin{cases} bias(n) = \frac{1}{T}\sum_{t=1}^{T}\left(\hat{X}_{n,t} - X_{n,t}\right) \\ SD(n) = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}\left(error(n) - bias(n)\right)^2} \end{cases}, \tag{20}$$

where $error\,(n) = \hat{X}_{n,t} - X_{n,t}$. Fig. 11 shows the biases and SD of three approaches. In Fig. 11(a), we sort all OD flows by the descending order according to their averages. From Fig. 11(a), we find that all the approaches show an under-estimate or over-estimate. Specially, the CNN+LSTM has smaller biases than the others. Meanwhile, the biases are decreased with the means of OD flows decrease. The biases of SRMF are not consistently related to flow size, as in CNN+LSTM and PCA. It has prominent under-estimates and over-estimates. For small OD flows, SRMF appears a remarkable under-estimate. The biased estimators sometimes may have lower SD, and then predictors are closer to the real value than those of unbiased predictor. Thereby, we refer to the SD (or variance) of bias for further evaluation. In Fig. 11(b), we find that CNN+LSTM and PCA have lower bias, but higher SD. An approach with low bias and high SD tends to predict the long-term traffic [24]. Otherwise, it prefers to predict the short-term traffic, when it has high bias and low SD. Hence, CNN+LSTM and PCA are suitable for long-term traffic prediction, and SRMF for short-term traffic prediction. As mentioned before, the SRMF is a matrix interpolation algorithm used to reconstruct the missing data in TM. Generally, the missing probability of each element is independent identically distributed. To extending this matrix interpolation algorithm to predict network traffic, it assumes

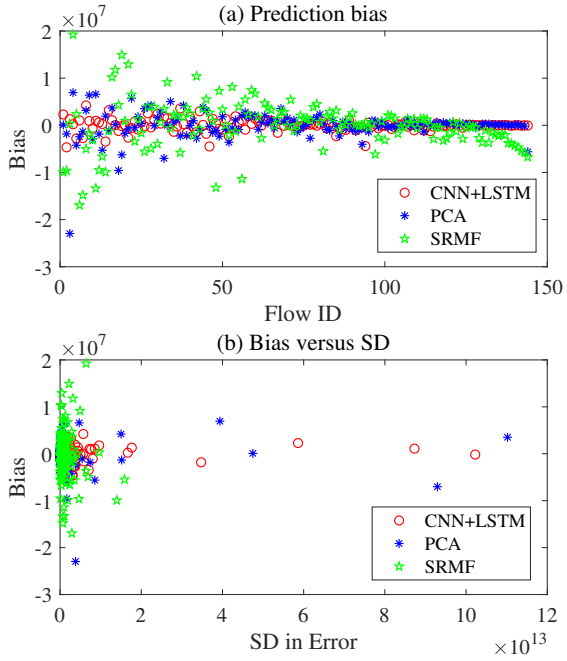

Fig. 11. Prediction bias and its SD in Abilene.

that the missing elements are a series of columns in TM. Under this case, SRMF predicts an element from a snapshot of TM. Hence, it is good at predicting short-term traffic. On the contrary, PCA and CNN+LSTM predict a network traffic element by using a great number of snapshots of TM. For instance, the singular value decomposition is a matrix-oriented operation. We use CNN as the first hidden layer, and employ a matrix as the input of deep architecture. As a result, PCA and our method are suitable for long-term traffic prediction. We can obtain the similar conclusion according to the evaluations in Fig. 12.

Finally, we evaluate the performance improvement ratio for our approach versus PCA and SRMF. The performance improvement ratio is expressed by:

$$PIR = \frac{\sum_{n=1}^{N^2}\sum_{t=1}^{T}\left|\hat{X}_{n,t}^{A} - X_{n,t}\right| - \sum_{n=1}^{N^2}\sum_{t=1}^{T}\left|\hat{X}_{n,t}^{B} - X_{n,t}\right|}{\sum_{n=1}^{N^2}\sum_{t=1}^{T}\left|\hat{X}_{n,t}^{A} - X_{n,t}\right|}, \tag{21}$$

where $\hat{X}_{n,t}^{A}$ and $\hat{X}_{n,t}^{B}$ are the predictors obtained via algorithms A and B, respectively. According to Eq. (8), the performance ratios of CNN+LSTM versus PCA and SRMF are 43.34% and 71.47% in Abilene. Moreover, they are 60.93% and 64.54% in our testbed, respectively.

6 CONCLUSIONS AND FUTURE WORK

This paper investigates the problem of end-to-end network traffic prediction in the IoV backbone network. Aiming at minimizing the prediction error and improving real-time performance of network traffic prediction, we design a deep architecture based on CNN and LSTM. This hybrid deep architecture is built by considering

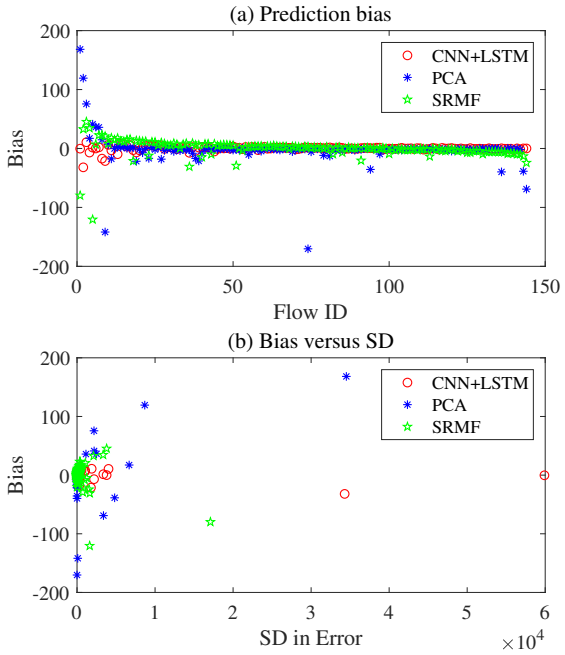

Fig. 12. Prediction bias and its SD in testbed.

two features of TM, i.e., the temporal and spatio-temporal features. Convolutional and subsampling layers are used for extracting the spatio-temporal features of TM. Meanwhile, an LSTM layer is leveraged to capture the temporal features. To improve the real-time performance of the proposed deep architecture for prediction, we propose a threshold-based deep architecture update policy. Furthermore, we propose a Q-learning algorithm to gain this threshold, in which link loads and routing information are employed to generate NMAE. In this case, the threshold can be calibrated to obtain an optimal tradeoff between real-time performance and prediction error. The proposed approach is evaluated by real network traffic data set from the Abilene backbone network and our constructed testbed. From these evaluations, the proposed approach can track the trace of end-to-end network traffic precisely.

The 5G-enabled communication network, in which infrastructures are employed densely, has been involved in IoVs to provide adequate communication resources. Hence, network traffic prediction approaches for large-scale and heterogeneous IoV backbone networks are necessary. Under this case, the computational complexity of a prediction approach is significantly important for online network traffic prediction. Thereby, a lightweight network traffic prediction approach is the main work in the future.

7 ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grant 2018YFE0206800, in part by the National Natural Science Foundation of China under Grants 61936001, 61971084, 62001073 and 61701406, in part by the National Natural Science Foundation of Chongqing under Grants cstc2019jcyj-cxttX0002 and cstc2019jcyj-msxmX0208.

REFERENCES

[1] T. O. Adelani and A. S. Alfa. 2010. Hybrid Techniques for Large-Scale IP Traffic Matrix Estimation. In *2010 IEEE International Conference on Communications*. IEEE, 1–6.

[2] Nikolaos Athanasios Anagnostopoulos, Saad Ahmad, Tolga Arul, Daniel Steinmetzer, Matthias Hollick, and Stefan Katzenbeisser. 2020. Low-Cost Security for Next-Generation IoT Networks. *ACM Trans. Internet Technol.* 20, 3, Article 30 (Sept. 2020), 31 pages. https://doi.org/10.1145/3406280

[3] J. Andrusenko, R. L. Miller, J. A. Abrahamson, N. M. M. Emanuelli, R. S. Pattay, and R. M. Shuford. 2008. VHF general urban path loss model for short range ground-to-ground communications. *IEEE Transactions on Antennas and Propagation* 56, 10 (2008), 3302–3310.

[4] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibaez. 2018. Internet of Vehicles: Architecture, Protocols, and Security. *IEEE Internet of Things Journal* 5, 5 (2018), 3701–3709.

[5] Giuseppe Faraci, Christian Grasso, and Giovanni Schembra. 2020. Fog in the Clouds: UAVs to Provide Edge Computing to IoT Devices. *ACM Trans. Internet Technol.* 20, 3, Article 26 (Aug. 2020), 26 pages. https://doi.org/10.1145/3382756

[6] Qiang He, Xingwei Wang, Zhencheng Lei, Min Huang, Yuliang Cai, and Lianbo Ma. 2019. TIFIM: A Two-stage Iterative Framework for Influence Maximization in Social Networks. *Appl. Math. Comput.* 354 (2019), 338 – 352. https://doi.org/10.1016/j.amc.2019.02.056

[7] Z. Hu, Y. Qiao, and J. Luo. 2018. ATME: Accurate Traffic Matrix Estimation in Both Public and Private Datacenter Networks. *IEEE Transactions on Cloud Computing* 6, 1 (2018), 60–73.

[8] B. Hussain, Q. Du, A. Imran, and M. A. Imran. 2020. Artificial Intelligence-Powered Mobile Edge Computing-Based Anomaly Detection in Cellular Networks. *IEEE Transactions on Industrial Informatics* 16, 8 (2020), 4986–4996.

[9] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, and A. Ksentini. 2020. Named Data Networking in Vehicular Ad Hoc Networks: State-of-the-Art and Challenges. *IEEE Communications Surveys & Tutorials* 22, 1 (2020), 320–351.

[10] T. Li, J. Yuan, and M. Torlak. 2018. Network Throughput Optimization for Random Access Narrowband Cognitive Radio Internet of Things (NB-CR-IoT). *IEEE Internet of Things Journal* 5, 3 (June 2018), 1436–1448. https://doi.org/10.1109/JIOT.2017.2789217

[11] M. Liu, L. Liu, H. Song, Y. Hu, Y. Yi, and F. Gong. 2020. Signal Estimation in Underlay Cognitive Networks for Industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 16, 8 (2020), 5478–5488.

[12] X. Liu, L. Che, K. Gao, and Z. Li. 2020. Power System Intra-Interval Operational Security Under False Data Injection Attacks. *IEEE Transactions on Industrial Informatics* 16, 8 (2020), 4997–5008.

[13] Marcin Luckner, Maciej Grzenda, Robert Kunicki, and Jaroslaw Legierski. 2020. IoT Architecture for Urban Data-Centric Services and Applications. *ACM Trans. Internet Technol.* 20, 3, Article 29 (July 2020), 30 pages. https://doi.org/10.1145/3396850

[14] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. 2017. Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification. *IEEE Transactions on Geoscience and Remote Sensing* 55, 2 (2017), 645–657. https://doi.org/10.1109/TGRS.2016.2612821

[15] L. Nie, Z. Ning, X. Wang, X. Hu, Y. Li, and J. Cheng. 2020. Data-Driven Intrusion Detection for Intelligent Internet of Vehicles: A Deep Convolutional Neural Network-based Method. *IEEE Transactions on Network Science and Engineering* (2020), 1–1.

[16] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, and R. Y. K. Kwok. 2020. Intelligent Edge Computing in Internet of Vehicles: A Joint Computation Offloading and Caching Solution. *IEEE Transactions on Intelligent Transportation Systems* (2020), 1–14.

[17] Z. Ning, K. Zhang, X. Wang, M. S. Obaidat, L. Guo, X. Hu, B. Hu, Y. Guo, B. Sadoun, and R. Y. K. Kwok. 2020. Joint Computing and Caching in 5G-Envisioned Internet of Vehicles: A Deep Reinforcement Learning-Based Traffic Control System. *IEEE Transactions on Intelligent Transportation Systems* (2020), 1–12.

[18] A. Omidvar and H. S. Shahhoseini. 2011. Intelligent IP traffic matrix estimation by neural network and genetic algorithm. In *2011 IEEE 7th International Symposium on Intelligent Signal Processing*. IEEE, 1–6.

[19] Carlo Puliafito, Enzo Mingozzi, Francesco Longo, Antonio Puliafito, and Omer Rana. 2019. Fog Computing for the Internet of Things: A Survey. *ACM Trans. Internet Technol.* 19, 2, Article 18 (April 2019), 41 pages. https://doi.org/10.1145/3301443

[20] C. Qiu, Y. Zhang, Z. Feng, P. Zhang, and S. Cui. 2018. Spatio-Temporal Wireless Traffic Prediction With Recurrent Neural Network. *IEEE Wireless Communications Letters* 7, 4 (Aug 2018), 554–557. https://doi.org/10.1109/LWC.2018.2795605

[21] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu. 2012. Spatio-temporal compressive sensing and Internet traffic matrices (extended version). *IEEE Transactions on Networking* 20, 3 (2012), 662–676.

[22] S. K. Singh and A. Jukan. 2018. Machine-learning-based prediction for resource (Re)allocation in optical data center networks. *IEEE/OSA Journal of Optical Communications and Networking* 10, 10 (Oct 2018), D12–D28. https://doi.org/10.1364/JOCN.10.000D12

[23] S. Sinha and C. S. R. Murthy. 2005. Information theoretic approach to traffic adaptive WDM networks. *IEEE/ACM Transactions on Networking* 13, 4 (2005), 881–894.

[24] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot. 2005. Traffic matrices: balancing measurements, inference and modeling. In *Proceedings of SIGMETRICS 2005*. IEEE, 362–373.

[25] D. A. Tedjopurnomo, Z. Bao, B. Zheng, F. Choudhury, and A. K. Qin. 2020. A Survey on Modern Deep Neural Network for Traffic Prediction: Trends, Methods and Challenges. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.

[26] D. Wang, J. Fan, Z. Xiao, H. Jiang, H. Chen, F. Zeng, and K. Li. 2019. Stop-and-Wait: Discover Aggregation Effect Based on Private Car Trajectory Data. *IEEE Transactions on Intelligent Transportation Systems* 20, 10 (Oct 2019), 3623–3633. https://doi.org/10.1109/TITS.2018.2878253

[27] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang. 2017. Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. IEEE, 1–9.

[28] X. Wang, Z. Ning, S. Guo, and L. Wang. 2020. Imitation Learning Enabled Task Scheduling for Online Vehicular Edge Computing. *IEEE Transactions on Mobile Computing* (2020), 1–1.

[29] J. Weng, J. Weng, Y. Zhang, W. Luo, and W. Lan. 2019. BENBI: Scalable and Dynamic Access Control on the Northbound Interface of SDN-Based VANET. *IEEE Transactions on Vehicular Technology* 68, 1 (Jan 2019), 822–831. https://doi.org/10.1109/TVT.2018.2880238

[30] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li. 2020. LSTM Learning With Bayesian and Gaussian Processing for Anomaly Detection in Industrial IoT. *IEEE Transactions on Industrial Informatics* 16, 8 (2020), 5244–5253.

[31] S. Yang, Y. Su, Y. Chang, and H. Hung. 2019. Short-Term Traffic Prediction for Edge Computing-Enhanced Autonomous and Connected Cars. *IEEE Transactions on Vehicular Technology* 68, 4 (April 2019), 3140–3153. https://doi.org/10.1109/TVT.2019.2899125

[32] C. Zhang, H. Zhang, D. Yuan, and M. Zhang. 2018. Citywide Cellular Traffic Prediction Based on Densely Connected Convolutional Neural Networks. *IEEE Communications Letters* 22, 8 (Aug 2018), 1656–1659. https://doi.org/10.1109/LCOMM.2018.2841832

[33] L. Zhang and Y. Liang. 2019. Joint Spectrum Sensing and Packet Error Rate Optimization in Cognitive IoT. *IEEE Internet of Things Journal* 6, 5 (Oct 2019), 7816–7827. https://doi.org/10.1109/JIOT.2019.2907993

[34] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu. 2017. LSTM network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems* 11, 2 (2017), 68–75.