Deep Convolutional LSTM Network-based Traffic Matrix Prediction with Partial Information

Van An Le*, Phi Le Nguyen*, Yusheng Ji*†,
*Department of Informatics, SOKENDAI (The Graduate University for Advanced Studies), Tokyo, Japan
†National Institute of Informatics, Tokyo, Japan
Email: *{anle, nguyenle, kei}@nii.ac.jp

***Abstract*—Accurate prediction of the future network traffic plays an important role in various network problems (e.g. traffic engineering, capacity planning, quality of service provisioning, etc.). However, the modern network communication is extremely complicated and dynamic, which makes the tasks of modeling and predicting the network behavior very difficult. To this end, a common approach is to apply the traditional time series prediction techniques such as Autoregressive Integrated Moving Average or Linear Regression. Besides that, there are some studies exploiting Deep Learning techniques such as Restricted Boltzmann Machine or Recurrent Neural Network (RNN) to estimate the traffic volume. Although the prediction accuracy largely depends on the amount of historical data, measuring all the network traffic is impossible or impractical due to the monitoring resources constraints as well as the dynamics of temporal/spatial fluctuations of the traffic. Thus, the state-of-the-art proposals reveal poor performance regarding the traffic inference when lacking ground-truth input.**

In this paper, we propose a highly accurate traffic prediction algorithm by leveraging the Convolutional LSTM network (ConvLSTM), which is the integrated model of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) network, for spatiotemporal modeling and estimating the future network traffic. We also propose a technique which exploits the RNN to correct the imprecise data in the input. To evaluate the proposed algorithm, we conduct extensive experiments using the Abilene dataset which contains the real network traffic trace. The experiment results show that our proposed approach outperforms the existing algorithms in terms of several metrics including error ratio, root mean square error, and coefficient of determination, in both one-step-ahead and multi-step-ahead prediction with partial information.

I. Introduction

In recent years, the great demand for the Internet services (e.g. video streaming, VoIP, etc.) has led to an exponential growth in the backbone network traffic. Having a better knowledge about the backbone network traffic, specifically being able in predicting future traffic, thus becomes a critically important factor to perform management tasks such as traffic engineering, capacity planning and quality of service provisioning. For example, in the well-known Network Utility Maximization (NUM) [1], [2] (which usually provides a bandwidth allocation or smart routing solution by solving optimization problem), the future network traffic knowledge (e.g. user demands, link usages, end-to-end latency) is used as the input. However, due to the explosion of backbone network traffic as well as the complexity and dynamics of network communication behavior, modeling and estimating the future traffic in backbone networks becomes a significant challenge.

978-3-903176-15-7

In the literature, numerous effort has been done on predicting future traffic in data centers and cellular networks. In the traditional approaches, regression techniques (e.g., Autoregressive Integrated Moving Average (ARIMA) [3]) are exploited to obtain the predicted value. However, ARIMA has shown a poor performance due to two main reasons: 1) The communication behavior has become too dynamic and complicated to be modeled by a linear system. 2) ARIMA ignores the spatial relation between the traffic flows and processes the flows independently. However, the recent studies have shown that there is a strong relation between the flows, which can be utilized to improve the prediction accuracy [4], [5]. Recently, deep learning has been widely applied to various application domains such as image/video processing, natural language recognition, etc., and achieved breakthrough results. In traffic analysis domain, deep learning algorithms have shown superior capability in solving the modeling and predicting non-linear time series problems. Nie et al. [6] used Restricted Boltzmann Machine to accurately predict the future traffic volume. Alternatively, the studies in [4] and [7] exploited the Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) for capturing the spatial and temporal features.

Unfortunately, all of the existing algorithms proposed so far require precise historical data as the input. This requirement can be easily accomplished in the context of data centers and cellular networks since these networks are controlled through top-of-rack switches or base stations, respectively. However, collecting all the traffic data in backbone networks is impractical due to the complexity of network topology, the resource limitation of network devices and the high overhead of monitoring high-speed network. Therefore, estimating the future traffic in backbone networks suffers from the problem of missing ground-truth input where some traffic flows cannot be monitored at a particular time. A common approach to fill into the missing data is utilizing the data generated by the prediction model. However, this may cause a huge degradation in the prediction's performance.

In this paper, we address the problem of modeling and predicting the future traffic in backbone networks under the lack of historical traffic data. More specifically, we focus on predicting the future traffic matrices which represent the

traffic volume between all the origins (i.e., sources) and the destinations in the network. We leverage the so-called Convolutional LSTM (ConvLSTM, for short) network [8] (i.e., a combination of CNN and LSTM network) in extracting and modeling the spatiotemporal feature of the traffic matrix data. Besides that, since the prediction accuracy of the model strongly depends on the preciseness of feeding data, the most challenging problem is how to correct the feeding data so as to minimize the gap between the feeding data and the ground-truth. To this end, we propose two techniques. First, by constructing a backward network which processes the input in the inverse order of time, we have more information to correct the previous predicted data (which is usually imprecise) before feeding it into the model to predict the future traffic. Secondly, we construct a mechanism to sample the ground-truth at a certain rate. Specifically, by comparing the trend and the error in the historical prediction of each flow, we design a formula to determine which flows should be monitored at the next timestep. By doing so, we can balance between the monitoring overhead (in terms of flow monitoring rate) and the prediction accuracy. The contribution of this paper can be summarized as follows:

- To our best knowledge, we are the first one considering the problem of predicting future traffic in backbone networks where the historical data may be missed. Our approach is different from existing deep learning approaches in time series prediction where the model is fed with only the ground-truth input.
- We exploit the ConvLSTM in handling the spatiotemporal of the traffic matrices in backbone networks and construct a model which combines the forward and backward ConvLSTM networks. This model is able to correct the input data to improve the accuracy of the future traffic prediction. Moreover, we also design a formula to determine which flows should be measured in the future.
- We evaluate the performance of our proposed algorithms by conducting extensive experiments on real backbone network traffic dataset and comparing the results with state-of-the-art approaches.

The rest of the paper is organized as follows: we first introduce the problem of traffic prediction under the lack of precise input data and give a brief introduction about LSTM and ConvLSTM network in Section 2. Section 3 presents the motivation and the details of our proposed algorithms for correcting the input data and determining the monitored flow set. Section 4 shows extensive experimental results. We present some related works in Section 5 and conclude the paper in Section 6.

II. Preliminaries

In this section, we first present the problem formulation of traffic matrix prediction under the lack of ground-truth input. Then, we give a brief introduction about the Long Short-Term Memory and Convolutional LSTM networks which are used in our approach.

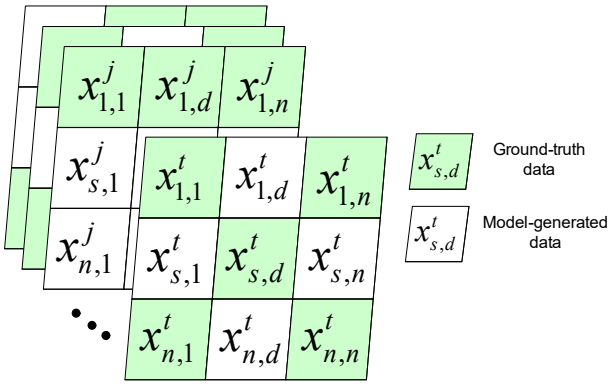


Fig. 1: A sequence of J previous traffic matrices including both ground-truth and imprecise data.

A. Problem Description

We suppose that the traffic monitoring and predicting tasks are executed periodically in every *timestep* and denote t as the current timestep. Given a backbone network in which $\mathcal{N}$ is the set of nodes ($|\mathcal{N}| = n$), let $X_j \in \mathbb{R}^{n \times n}$ be the traffic matrix at the timestep j. Each element $x^j_{s,d} \in X_j$ represents the traffic volume of a flow from a source s to a destination d in the network (flow (s, d), for short) at timestep j. Thus, the traffic matrix prediction problem is to estimate the traffic matrices of next L timesteps (denoted by $\widetilde{X}_{t+1}, ..., \widetilde{X}_{t+L}$), given the previous J measurements ($L, J \geq 1$):

$$\widetilde{X}_{t+1}, ..., \widetilde{X}_{t+L} = \underset{X_{t+1},...,X_{t+L}}{\operatorname{argmax}} \; p(X_{t+1}, ..., X_{t+L} | X_{t-J+1}, ..., X_t) \quad (1)$$

However, in this paper, we consider the case of backbone networks where we cannot obtain all the J previous traffic matrices by directly monitoring all the flows. Although in recent years, advance network architectures such as Software-Defined Networking (SDN) [9] and Network Function Virtualization (NFV) have enabled many alternative ways to measure the network flow information, collecting all the traffic statistics with low overhead (in terms of computational complexity, bandwidth, ...) still remains as a challenging task. Therefore, to reduce the monitoring overhead, we measure only a part of the traffic flows at each timestep and use the data generated by the prediction model to fill into the missed historical data. Accordingly, the traffic matrix prediction problem under highly missing ground-truth data can be formulated as follows:

Input

$$x^j_{s,d} = \begin{cases} o^j_{s,d} & \text{if } m^j_{s,d} = 1 \\ \widetilde{x}^j_{s,d} & \text{otherwise} \end{cases} \quad (2)$$
$$\forall s, d \in N; j = t - J + 1, ..., t$$

Output

$$\widetilde{X}_{t+1}, ..., \widetilde{X}_{t+L} = \underset{X_{t+1},...,X_{t+L}}{\operatorname{argmax}} \; p(X_{t+1}, ..., X_{t+L} | X_{t-J+1}, ..., X_t) \quad (3)$$

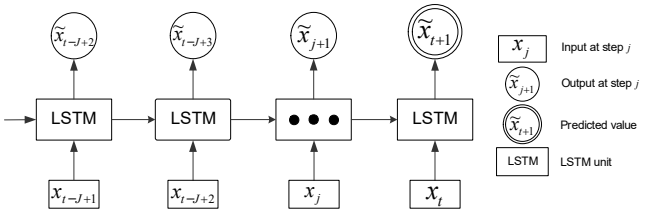


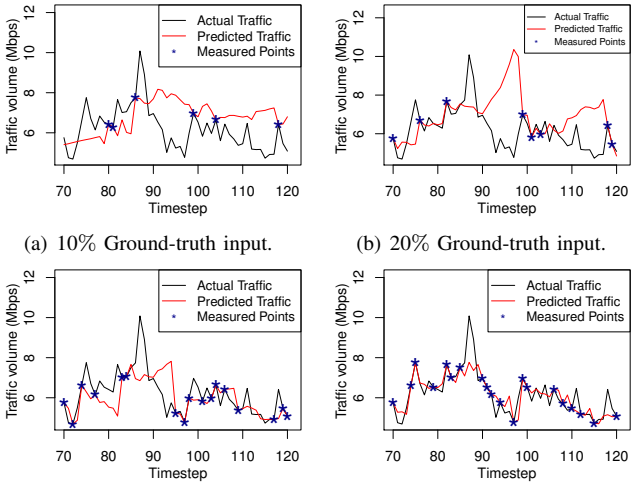
Fig. 2: The unfolded model of the LSTM network.

where $o^j_{s,d}$ and $\widetilde{x}^j_{s,d}$ denote the observed and predicted value of flow (s,d) at timestep j, respectively. The binary variable $m^j_{s,d}$ depends on the monitoring policy, where $m^j_{s,d}=1$ indicates that flow (s,d) is monitored at timestep j; otherwise, the value of traffic volume is filled up by the prediction result. Fig.1 shows the sequence of J previous traffic matrices which is used as the input for predicting the future traffic. The green elements are the ground-truth data obtained by monitoring the flows directly, and the white elements represent the predicted values.

B. LSTM and Convolutional LSTM network for spatiotemporal sequence modeling

Long Short-Term Memory network is a special Recurrent Neural Network which replaces the standard RNN units by the LSTM units. LSTM network has been proved to be stable and powerful for modeling long-range dependencies in various problem domains, thus it is well-suited for processing and making predictions based on time series or sequence data. Indeed, LSTM has been applied in many real-life sequence modeling problems. The unfolded model of LSTM network (Fig.2) shows that the input is processed step-by-step and the outputs of previous steps are used as the input for the next step. This architecture along with the advantages of the memory cell in LSTM units make LSTM network especially suitable for solving the series based predictions.

However, in many problems such as precipitation nowcasting [8] or images/videos based action prediction where the sequence data has a strong spatial relation, LSTM reveals many limitations. Therefore, to deal with more general spatiotemporal sequence forecasting problems, authors in [8] have proposed an extension of LSTM called ConvLSTM. The ConvLSTM layer has convolutional structures in both the input-to-state and state-to-state transitions which can exploit both temporal and spatial features of the input sequence. To obtain the spatiotemporal feature, the ConvLSTM network takes a 3D tensor X_t as input in the processing step t, for example, X_t can be a $32\times32\times3$ image whose the last dimension represents the color of the image (RGB). The key equations of the ConvLSTM are shown in (4), where i_t, f_t, o_t are the input, forget, output gates, respectively; C_t, H_t are the cell and the final state of the LSTM unit, respectively; '$*$' denotes the convolution operator and '$\circ$' denotes the



(a) 10% Ground-truth input. (b) 20% Ground-truth input.

(c) 30% Ground-truth input. (d) 40% Ground-truth input.

Fig. 3: The effect of ground-truth input on prediction accuracy. (One-step-ahead prediction using the LSTM network.)

Hadamard product [8]:

$$\begin{aligned} i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\ f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\ C_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\ o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \\ H_t &= o_t \circ \tanh(C_t) \end{aligned} \tag{4}$$

The main difference between the LSTM unit and ConvLSTM unit is the convolution operation (i.e., '$*$'). While the LSTM network only takes a 1D array as the input at each processing step, the ConvLSTM takes a 3D tensor as the input and uses multiple filters to extract the spatial information. The filter size (also called as kernel size) is $k\times k\times d$, where k is an integer number (normally, $k=3,5,7,etc.$) and d equals to the last dimension of the input. These filters are shifted across the 3D input, and hence, can extract the spatial features of the data.

III. Proposed Prediction Model

In this section, we first describe two challenges in solving the traffic matrix prediction problem with partial information in Section III-A. These two challenges then are addressed in Section III-B and III-C, respectively.

A. Motivation

In order to deploy the traffic matrix prediction model, we face two important challenges. The first one is the accumulative error in the prediction results over timesteps. We have conducted an experiment to figure out the impact of imprecise input data on the results of one-step-ahead prediction (i.e., $L=1$) using the LSTM network. Fig.3 shows the one-step-ahead prediction results with various settings of the percentage of ground-truth data in the input. As shown, the accuracy decreases over the timesteps between two consecutive measurement points (i.e., Fig.3(a), Fig.3(b)). While in Fig.3(d),

thanks to the higher percentage of ground-truth data in the input, we can capture the trend of the flow and achieve a high accuracy in forecasting the future traffic. Therefore, in order to alleviate the accumulative error while remaining the low monitoring overhead (which is proportional to the portion of the ground-truth data), our idea is to add a preprocessing on the imprecise data before feeding it into the prediction model. Specifically, we propose a novel deep learning model based on the bidirectional recurrent neural network (the details will be shown in Section III-B).

The second challenge is how to determine which flows to be measured at the next timestep. As mentioned above, in order to reduce the monitoring overhead we do not measure all network flows, thus choosing appropriate flows to monitor in each timestep becomes a critical factor in reducing the prediction error. One of the common approaches in choosing monitored flow set is to satisfy the fairness among all the flows. Specifically, the gaps between every two consecutive monitored timesteps of every flow are kept to be approximately the same. However, this method may not be effective since the network flows are dynamic and have various temporal fluctuation patterns. To deal with this problem, we propose a novel scheme for selecting the next monitored flows, which exploits the previous prediction results and the fluctuation characteristic of all flows (see the details in Section III-C).

B. Spatiotemporal traffic matrix prediction and input data correction

The ConvLSTM network has shown the effective capability in processing the spatial and temporal time series data where the inputs are 3D tensors such as images or video frames. Therefore, to apply the ConvLSTM network in our problem, we first consider the traffic matrices as 2D images with the dimension of $n \times n$, and then we transform them into 3D tensors by adding extra binary matrices (called as measurement matrices). Specifically, the measurement matrix M_j, which is added to the traffic matrix at timestep j, is a matrix whose each element $m^j_{s,d} \in M_j$ indicates whether the corresponding value in the traffic matrix is an observed value (i.e., $m^j_{s,d} = 1$) or a predicted value (i.e., $m^j_{s,d} = 0$). Thus, the input of our ConvLSTM network is a sequence of J 3D tensors whose dimension is $n \times n \times 2$. In order to predict multi-step traffic matrices, we apply the Iterated Multi-Step estimation (IMS) approach [10]. Specifically, we first use the many-to-many model to predict the traffic matrix of the next timestep and then, iteratively feed the generated data into the model to get the multi-step-ahead prediction. Fig.4 describes the many-to-many model which takes J 3D tensors from timestep $t-J+1$ to t as the input and predicts the next traffic matrix $\widetilde{X}_{t+1}$.

Now, we describe our technique for alleviating the error in the input data. Following the experiment in Section III-A (Fig.3), we observe that the prediction results become more accurate if the input sequence contains more precise data. Besides that, the accuracy of the predicted traffic at a timestep whose previous timestep's data is ground-truth data is better than the others. Based on the above observations, we propose

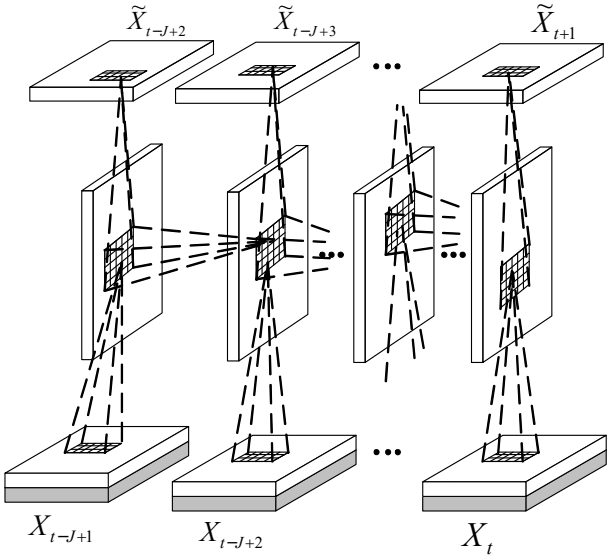


Fig. 4: The many-to-many model of ConvLSTM network for traffic matrix prediction.

an algorithm which uses the outputs from each processing step of the ConvLSTM network (i.e., $\widetilde{X}_{t-J+2}, ..., \widetilde{X}_t$) and the measurement matrices to correct the imprecise data. To be more specific, considering an example where we predict the traffic matrix X_{t+1} by taking the sequence $\{X_{t-J+1}, ..., X_t\}$ as the input. As using the many-to-many model, after the prediction, we also obtain the outputs $\{\widetilde{X}_{t-J+2}, ..., \widetilde{X}_{t+1}\}$ whose each element corresponds to one processing step of the ConvLSTM network. Suppose that $x^j_{s,d} \in X_j$ $(t - J + 2 \leq j \leq t)$ is an imprecise data and $\widetilde{x}^j_{s,d} \in \widetilde{X}_j$ is its corresponding output in the ConvLSTM network. Because of the forward direction in the processing step of the ConvLSTM network, the output $\widetilde{x}^j_{s,d}$ is generated by taking only the input from timestep $t - J + 1$ to $j - 1$. If the input sequence from timestep $t - J + 1$ to $j - 1$ contains almost ground-truth data, we can replace $x^j_{s,d}$ by $\widetilde{x}^j_{s,d}$. However, we face with the problem when the input sequence from timestep $t - J + 1$ to $j - 1$ contains only a few ground-truth data. To this end, our idea is to leverage the data at the next timesteps (from $j+1$ to t) which may include more precise data.

In order to implement the above idea, besides the current ConvLSTM network (which we call as the *forward network*), we construct an extra network (called as the *backward network*) in which the input data is fed in the reverse order. Our approach is motivated by the Bidirectional Recurrent Neural Network (BRNN) which was introduced in [11]. Thanks to adding a backward network, BRNN can be trained by all available input information in both the past and the future of a specific time frame. Different from the standard BRNN, in each processing step, instead of aggregating the outputs of the forward and backward networks, we keep them separately (as shown in Fig.5), and use the outputs from the backward network to update the incorrect data in the input matrices.

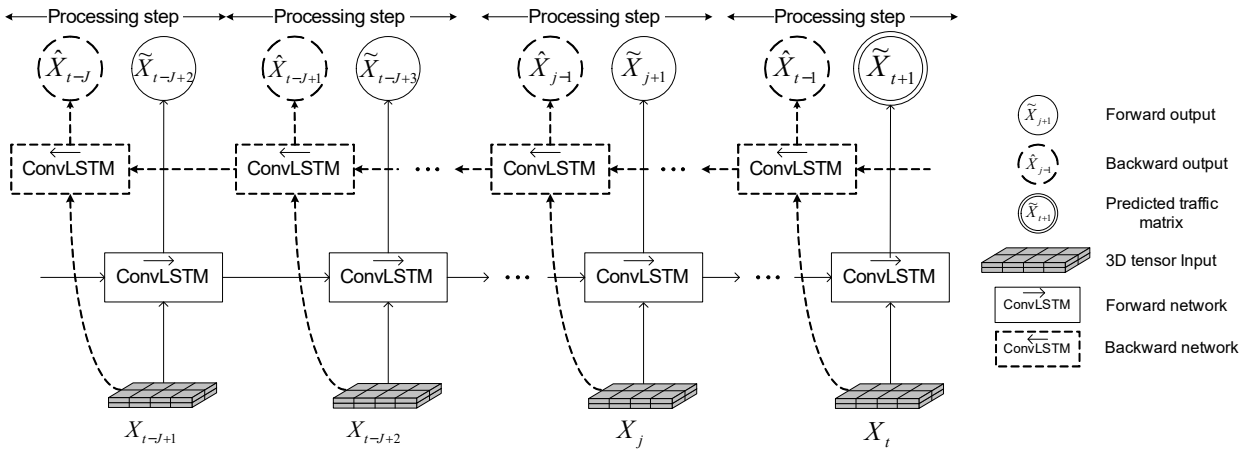


Fig. 5: The forward and backward ConvLSMT networks.

Therefore, $x^j_{s,d}$ now can be updated by using the output of the forward network (i.e., $\widetilde{x}^j_{s,d}$) or the output of the backward network (i.e., $\widehat{x}^j_{s,d}$). In order to determine the contribution of the old value (i.e., $x^j_{s,d}$), the outputs of the forward and backward networks (i.e., $\widetilde{x}^j_{s,d}$ and $\widehat{x}^j_{s,d}$) in updating imprecise data, we define parameters $\alpha_{s,d,t}$, $\beta^j_{s,d,t}$ and $\gamma^j_{s,d,t}$ which are the confidence factors of $x^j_{s,d}$, $\widetilde{x}^j_{s,d}$ and $\widehat{x}^j_{s,d}$, respectively ($\alpha_{s,d,t}+\beta^j_{s,d,t}+\gamma^j_{s,d,t}=1$). The details of the imprecise data correction algorithm is described in Algorithm 1. Note that since the outputs of the forward network are from timestep $t-J+2$ to $t+1$ while that of the backward network are from $t-J$ to $t-1$, we can only apply Algorithm 1 for correcting the inputs from timestep $t-J+2$ to $t-1$. In what follows, we describe the main idea behind Algorithm 1.

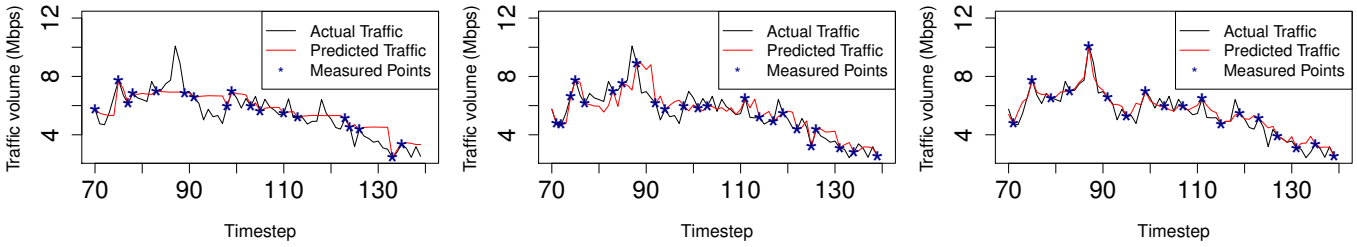
Obviously, the less the ground-truth data contained in the inputs, the less precise the predicted values, thus $x^j_{s,d}$ should not be updated if the historical data is highly missed. Accordingly, the confidence factor $\alpha_{s,d,t}$ in Algorithm 1 is determined by the portion of the flows that are not monitored (line 2). $\beta^j_{s,d,t}$ and $\gamma^j_{s,d,t}$ are determined based on two terms named *forward loss* and *backward loss*. The forward and backward losses of a flow (s,d) (denoted as $l^f_{s,d,t}$ and $l^b_{s,d,t}$, respectively) are defined to assess the performance of the forward and backward ConvLSTM networks after predicting the traffic at current timestep t. Specifically, $l^f_{s,d,t}$ and $l^b_{s,d,t}$ are defined as the root squared errors between the outputs and the ground-truth elements in the input (line 4, 5). Note that, if the input contains no ground-truth data, then the losses are assigned to ξ which is a very large positive number (line 8). Intuitively, the smaller $l^f_{s,d,t}$ (resp. $l^b_{s,d,t}$), the smaller the gap between $\widetilde{x}^j_{s,d}$ (resp. $\widehat{x}^j_{s,d}$) and the ground-truth. Therefore, if $l^f_{s,d,t} < l^b_{s,d,t}$, then $\widetilde{x}^j_{s,d}$ should contribute more than $\widehat{x}^j_{s,d}$ in correcting the imprecise input, otherwise, $\widehat{x}^j_{s,d}$ should contribute more than $\widetilde{x}^j_{s,d}$. Accordingly, the confidence factors $\beta^j_{s,d,t}$ is designed so that flows with the small $l^f_{s,d,t}$ and the large $l^b_{s,d,t}$ will have the large $\beta^j_{s,d,t}$. Inversely, flows with the small $l^b_{s,d,t}$ and the large $l^f_{s,d,t}$ will have the large $\gamma^j_{s,d,t}$. Moreover, with the observation that the more ground-truth elements the inputs contain, the more accurate the prediction result is, $\beta^j_{s,d,t}$ (resp. $\gamma^j_{s,d,t}$) is also designed so that the greater the percentage of the ground-truth data in the input at timesteps from $t-J+1$ to $j-1$, i.e., denoted as $\rho^j_{s,d,t}$ (resp. from $j+1$ to t, i.e., denoted as $\mu^j_{s,d,t}$), the greater the $\beta^j_{s,d,t}$. Consequently, $\beta^j_{s,d,t}$ and $\gamma^j_{s,d,t}$ are calculated based on both the network losses and the percentage of ground-truth data in the input (line 12, 13). Finally, $x^j_{s,d}$ is updated by the formulation described in line 14.

Algorithm 1: Forward and backward data correction

Input : X_k: the previous traffic matrix at timestep k
$\widetilde{X}_k$: the outputs of forward network
$\widehat{X}_k$: the outputs of backward network
M_k: the measurement matrix of X_k
$(k = t-J+2, ..., t-1)$

Output : The updated traffic matrices

1 **for** $s, d \in \mathcal{N}$ **do**
2 $\quad \alpha_{s,d,t} \leftarrow 1 - \frac{\sum_{i=t-J+1}^{t} m^i_{s,d,t}}{J}$;
3 $\quad$ **if** $(\sum_{i=t-J+1}^{t} m^i_{s,d} \neq 0)$ **then**
4 $\quad\quad l^f_{s,d,t} \leftarrow \sqrt{\sum_{j=t-J+2}^{t-1} m^j_{s,d} \times (\widetilde{x}^j_{s,d} - x^j_{s,d})^2}$;
5 $\quad\quad l^b_{s,d,t} \leftarrow \sqrt{\sum_{j=t-J+2}^{t-1} m^j_{s,d} \times (\widehat{x}^j_{s,d} - x^j_{s,d})^2}$;
6 $\quad$ **end**
7 $\quad$ **else**
8 $\quad\quad l^f_{s,d,t} \leftarrow \xi; \quad l^b_{s,d,t} \leftarrow \xi$;
9 $\quad$ **end**
10 $\quad$ **for** $j = t-J+2$ ***to*** $t-1$ **do**
11 $\quad\quad \rho^j_{s,d,t} \leftarrow \frac{\sum_{i=t-J+1}^{j-1} m^i_{s,d,t}}{j-t+J}; \quad \mu^j_{s,d,t} \leftarrow \frac{\sum_{i=j+1}^{t} m^i_{s,d,t}}{t-j}$;
12 $\quad\quad \beta^j_{s,d,t} \leftarrow \frac{(l^b_{s,d,t}+\rho^j_{s,d,t})(1-\alpha_{s,d,t})}{l^f_{s,d,t}+l^b_{s,d,t}+\rho^j_{s,d,t}+\mu^j_{s,d,t}}$;
13 $\quad\quad \gamma^j_{s,d,t} \leftarrow \frac{(l^f_{s,d,t}+\mu^j_{s,d,t})(1-\alpha_{s,d,t})}{l^f_{s,d,t}+l^b_{s,d,t}+\rho^j_{s,d,t}+\mu^j_{s,d,t}}$;
14 $\quad\quad x^j_{s,d} \leftarrow \alpha_{s,d,t} \times x^j_{s,d} + \beta^j_{s,d,t} \times \widetilde{x}^j_{s,d} + \gamma^j_{s,d,t} \times \widehat{x}^j_{s,d}$;
15 $\quad$ **end**
16 **end**
17 **return** X_k $(k = t-J+2, ..., t-1)$



(a) ARIMA (b) The LSTM network (c) Our proposal

Fig. 6: Prediction results and actual values of a random flow.

C. Determining the monitored flow set

Our main idea is that at each timestep t, we calculate a weight $w_{s,d,t}$ for each flow (s,d) and choose at most k flows which have the lowest weights to monitor at the next timestep (i.e., timestep $t+1$). The maximum number of flows that can be monitored (i.e., k) depends on the network monitoring policy. In the following, before going to the details of the weight's formula, we will first describe our theoretical basis. To ease the presentation, we call the periods between the consecutive measured timesteps of a flow as non-monitored periods of that flow.

Following the experiment results shown in Section III-A, we note that the longer the non-monitored periods, the larger the difference between the predicted results and the actual traffic. Thus, in order to reduce the prediction error, we should decrease the non-monitored periods of all flows. More specifically, the flows that have not been monitored for a long period should be chosen to be monitored at the next timestep. To this end, for each flow (s,d), we define a term named *consecutive missing* (denoted as $c_{s,d,t}$) which is the number of the timesteps from when (s,d) was last monitored till the current timestep t. The weight should be designed so that it will decline when the consecutive missing gets high.

Moreover, since our imprecise data correction algorithm is based on the outputs of the forward and backward networks, we need to keep the losses of these networks (i.e., $l^f_{s,d,t}$ and $l^b_{s,d,t}$, respectively) as smaller as possible. Accordingly, the flows with higher forward and backward losses should be chosen to be monitored at the next timestep.

Finally, we see that the flows with high fluctuation tend to have high prediction error. Therefore, these flows should be monitored more frequently. In order to measure the unsteadiness of flow (s,d), we use the standard deviation of the traffic volume of the flow (s,t) from timestep $t-J+1$ to t (denoted as $\eta_{s,d,t}$).

Consequently, the weight is calculated based on the flows' consecutive missing, backward loss, forward loss, and fluctuation as follows.

$$w_{s,d,t} = \frac{1}{\lambda_1 \times l^f_{s,d,t} + \lambda_2 \times l^b_{s,d,t} + \lambda_3 \times c_{s,d,t} + \lambda_4 \times \eta_{s,d,t}} \quad (5)$$

TABLE I: The configurations of ConvLSTM networks.

Convolutional layers	2	Recurrent dropout	0.2
No. of filters	8/layer	J	26
Filter size	(3, 3, 2)	L	12
Padding	'same'	No. of trained epoches	50
CNN dropout	0.0	Batch size	256
λ_1, λ_2	2.0, 1.0	λ_3, λ_4	5.0, 0.4

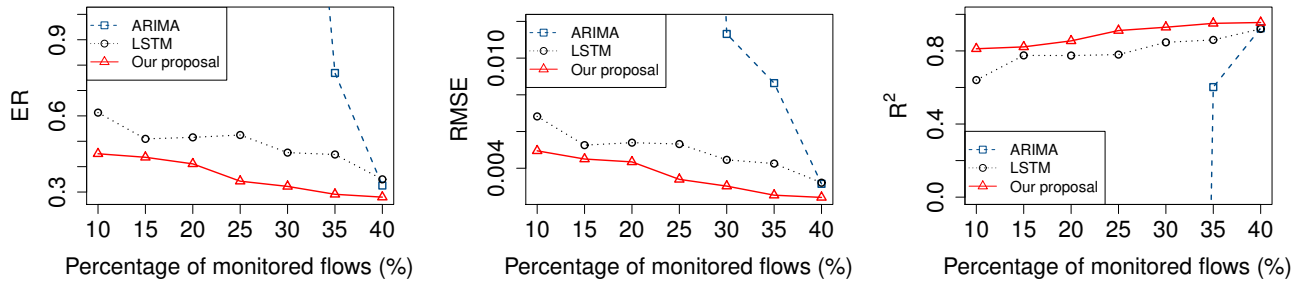
where λ_1, λ_2, λ_3 and λ_4 are hyper-parameters which are chosen by experiments.

At the end of the timestep t, the weights of all flows are calculated, and the first k flows with the lowest weights are chosen to be measured at the next timestep (i.e., timestep $t+1$).

IV. Performance Evaluation

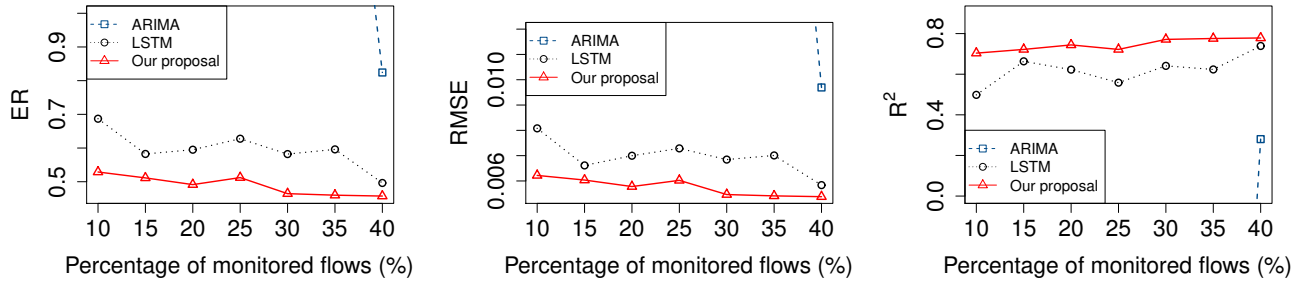
A. Settings

We evaluated the performance of our proposed approach by conducting extensive experiments on the Abilene dataset (available at [12]). The dataset contains the real trace data from the backbone network located in North America which contains 12 nodes ($n = 12$). The Abilene dataset, which includes averages over 5 minutes interval of 144 origin-destination flows from March 1 to September 11, 2004, has been widely used for performance evaluation in many traffic matrix interpolation and prediction studies [13], [5]. In the experiments, we separated the dataset into 60% for training, 20% and 20% for testing and validating, respectively. We compared our proposal with ARIMA [3] and the standard LSTM model. For ARIMA, we use the historical data of one month as the input for predict future traffic at each timestep. The performance metrics used including Error Ratio (ER), Root Mean Square Error ($RMSE$) and Coefficient of Determination (denoted as R^2 score) which are defined in (6). The ER and $RMSE$ are used for measuring the prediction error and the standard deviation of the differences between predicted values and ground-truth values, respectively (the lower values is better). The R^2 score determines how well the predicted values are generated by the model. The maximum value of R^2 is 1 and it can be a negative number. Moreover,



(a) Error Ratio (b) Root Mean Square Error (c) Coefficient of Determination

Fig. 7: Performance comparison in one-step-ahead prediction.



(a) Error Ratio (b) Root Mean Square Error (c) Coefficient of Determination

Fig. 8: Performance comparison in multi-step-ahead prediction.

a higher R^2 is better.

$$
\begin{aligned}
ER &= \frac{\sqrt{\sum_{s,d\in\mathcal{N}}\sum_{i=1}^{T}(1-m_{s,d}^{i})\times(o_{s,d}^{i}-x_{s,d}^{i})^2}}{\sqrt{\sum_{s,d\in\mathcal{N}}\sum_{i=1}^{T}(1-m_{s,d}^{i})\times(o_{s,d}^{i})^2}} \\
RMSE &= \sqrt{\frac{1}{D}\sum_{s,d\in\mathcal{N}}\sum_{i=1}^{T}(o_{s,d}^{i}-x_{s,d}^{i})^2} \\
R^2 &= 1-\frac{\sum_{s,d\in\mathcal{N}}\sum_{i=1}^{T}(o_{s,d}^{i}-x_{s,d}^{i})^2}{\sum_{s,d\in\mathcal{N}}\sum_{i=1}^{T}(o_{s,d}^{i}-\overline{o}_{s,d}^{i})^2}
\end{aligned}
\tag{6}
$$

where T is the number of timesteps, $D = T \times n \times n$ is the size and $\overline{o}_{s,d}^{i} = \frac{1}{D}\sum_{s,d\in\mathcal{N}}\sum_{i=1}^{T} o_{s,d}^{i}$ is the mean of the testing dataset. The experiments have been conducted on a computer which has Intel i7-6900K CPU @ 3.20GHz, 48 GB memory and two NVIDIA GeForce GTX 1080Ti. The detail configurations of the ConvLSTM networks (the forward and backward networks have the same configurations) are listed in Table I.

We conducted two experiments. First, we evaluated the performance of all the three algorithms in one-step-ahead traffic matrix prediction ($L = 1$). In the second experiment, we conducted a multi-step-ahead traffic matrix forecasting by predicting the traffic matrices of one hour ahead of the current timestep ($L = 12$). We denote p as the percentage of the monitored flows per timestep. In each experiment, we conducted different scenarios in which p is varied from 10% to 40% (i.e., the maximum number of monitored flows per timestep is $k = p \times n \times n$).

In the following, we will show the numerical results. Note that since the values regarding ARIMA are extremely large and lie outside the boundaries of the figures, they are also presented in Table II.

B. One-step-ahead traffic prediction results

In this section, we present the performance evaluation in one-step-ahead traffic prediction. First, we evaluate how well the algorithms can capture the traffic trend (Fig.6). The figure shows the difference between the predicted values and the actual values of a flow chosen randomly in about 6 hours (with 30% flows are monitored). As shown, our algorithm can capture the traffic trend much more smoothly compared with the other two algorithms. Besides that, thanks to the monitoring policy (described in Section III-C), our proposal monitors the traffic at more spikes than ARIMA and LSTM (which use simple random policies in deciding monitored flows).

Fig.7 shows the performance comparison among ARIMA, LSTM and our proposed approach in terms of the Error Ratio, Root Mean Square Error and Coefficient of Determination. It can be seen that our proposal achieves the best performance regarding all the metrics. LSTM shows the second best performance and ARIMA is the worst. Comparing LSTM and our proposal, we can see that our approach achieves a high accuracy in most of the experiment scenarios. Specifically, in the best case (i.e., $p = 35\%$ flows are monitored), the ER and $RMSE$ of our algorithm are 35.0% and 40.5% less than that of LSTM, respectively. In terms of R^2 score, our proposal achieves 26.7% better than the LSTM approach in the best case (i.e., $p = 10\%$). Moreover, it can be seen that our proposal in case $p = 25\%$, achieves better performance (regarding all metrics) as LSTM in case $p = 40\%$. In addition,

TABLE II: The prediction results of ARIMA

p	ER (exp1|exp2)	$RMSE$ (exp1|exp2)	R^2 (exp1|exp2)
10%	2.45e10 | 7.29e10	3.27e8 | 1.05e9	−5.72e20 | −5.8e21
15%	3.5e6 | 1.001e7	4.5e4 | 1.4e5	−1.1e13 | −1.09e14
20%	40220 | 158861	323.75 | 1406	−1.48e9 | −2.8e10
25%	7.27 | 17.98	0.07 | 0.21	−41.23 | −341.6
30%	2.89 | 11.5	0.032 | 0.087	−13.6 | −148.2
35%	0.77 | 2.04	0.008 | 0.03	0.62 | −3.58
40%	0.33 | 0.83	0.003 | 0.009	0.92 | 0.28

Fig.7 indicates that while the performance of our proposal is improved gradually when increasing the number of monitored flows, LSTM does not. The fluctuation in the results of LSTM can be explained by the random policy in choosing flows to be monitored after each timestep.

The ratio of ground-truth input has a massive impact on the prediction accuracy of ARIMA. As shown in Table II, when $p < 25\%$, we see the dramatical increase of ER and $RMSE$. Specifically, when $p = 10\%$, ER and $RMSE$ become extremely large, i.e., $ER = 2.45\text{e}10$, $RMSE = 3.27\text{e}8$. Similarly, the R^2 score also makes a huge decrease from -41.23 to $-5.27\text{e}20$ when reducing p from 25% to 10%. However, when the percentage of ground-truth data in the input is sufficiently large (i.e., $p = 40\%$), ARIMA can performs very well and its performance is close to that of our algorithm. This results strongly emphasizes the advantage of our model in predicting the future traffic with only a small portion of ground-truth information.

C. Multi-step-ahead traffic prediction results

In this section, we present the results of multi-step-ahead traffic prediction under various settings of the ratio of the monitored flows. In each timestep, our proposed algorithm and LSTM predict the traffic matrices of one hour ahead by applying the IMS approach, while ARIMA uses the Direct Multi-Step approach.

Fig.8 shows the performance evaluation of all the three algorithms in terms of ER, $RMSE$ and R^2 score. In general, similar to the first experiment, our approach achieves the best performance in all scenarios, followed by LSTM and ARIMA. In comparison with LSTM, in the best case (i.e., 10% of flows are monitored), the ER, $RMSE$ of our algorithm are about 23.0% less than that of LSTM. Moreover, our algorithm results in the R^2 score that is 41.2% less than LSTM. Similar to the first experiment, ARIMA also shows very poor performance compared to the other two approaches.

Comparing the results of the two experiments, it can be seen that the performance of the three algorithms in the second experiment is worse than that in the first experiment. However, the degradation of ARIMA is much more larger than that of our approach and LSTM. Specifically, in the best case (i.e., the percentage of the monitored flows is 10%), the prediction errors of our approach and LSTM increase by only 17.3% and 12.5%, respectively, while that of ARIMA is 197.6%.

In summary, by applying the Convolutional LSTM network and the forward/backward data correction technique, our proposal shows the superiority in handling the missing ground-truth in the historical data. Specifically, our proposed algorithm reduces the ER, $RMSE$ and increases the R^2 significantly compared to LSTM and ARIMA.

V. Related work

Traffic matrix estimation and prediction, especially in backbone networks, have attracted a lot of attention in the research community. To solve the traffic matrix estimation problem, the authors in [14] exploited the compressive sensing and network tomography to recover the missing data in the traffic matrices. Moreover, the studies [5] and [15] considered the spatial and temporal features by proposing a new structure of the traffic matrices (3-way tensor). The above methods can reduce the monitoring overhead by using only a small number of measurement data (under 30%). However, these approaches focused on the traffic interpolation problem while many network applications and management tasks require to forecast the future traffic usage or demand.

In the traffic matrix prediction problem, originally, researchers referred to some simple statistical models such as ARIMA or Gaussian model [16]. However, such simple models cannot handle the complexity and dynamics of the communication behavior in modern networks. To this end, deep learning techniques have been exploited more in predicting traffic [4], [7], [6]. In [4] and [7], the authors utilized the Long Short-Term Memory and Convolutional Neural Network for capturing the spatiotemporal feature and predicting the network traffic in data center and cellular networks, respectively. In backbone network, Nie et al. [6] used Restricted Boltzmann Machine to capture the dynamic features of network. The authors then proposed two separated deep belief network models for solving the traffic estimation and prediction, independently. More recent, the results in [17] and [18] showed the superiority of LSTM network in modeling the temporal feature and the long-range dependencies of network traffic. Unfortunately, all the approaches proposed so far assumed that the input contains only the ground-truth data. To the best of our knowledge, there is no existing work addressing the problem of traffic prediction under missing ground-truth data in backbone network.

VI. Conclusion

In this paper, we addressed the traffic matrix prediction problem in backbone networks where the future traffic is estimated under the lack of precise historical data. We exploited the Convolutional LSTM network for extracting and modeling the spatiotemporal feature of the traffic matrices. We proposed a novel deep learning model and techniques which leverage the forward and backward ConvLSTM networks to correct input data and determine flows monitored at the next timestep. We conducted extensive experiments for evaluating our proposed approach. The results demonstrated that the proposed approach outperforms other well-known methods for time series analysis.

Acknowledgment

This research is supported in part by JSPS KAKENHI Project No. JP16H02817.

REFERENCES

[1] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang. Experience-driven networking: A deep reinforcement learning based approach. *arXiv preprint arXiv:1801.05757*, 2018.

[2] S. H. Low and D. E. Lapsley. Optimization flow controli: basic algorithm and convergence. *IEEE/ACM Transactions on Networking (TON)*, 7(6):861–874, 1999.

[3] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[4] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang. Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach. In *IEEE Conference on Computer Communications (INFOCOM'17)*, pages 1–9, 2017.

[5] K. Xie, L. Wang, X. Wang, G. Xie, J. Wen, and G. Zhang. Accurate recovery of internet traffic data: A tensor completion approach. In *IEEE Conference on Computer Communications (INFOCOM'16)*, pages 1–9, 2016.

[6] L. Nie, D. Jiang, L. Guo, and S. Yu. Traffic matrix prediction and estimation based on deep learning in large-scale ip backbone networks. *Journal of Network and Computer Applications*, 76:16–22, 2016.

[7] X. Cao, Y. Zhong, Y. Zhou, J. Wang, C. Zhu, and W. Zhang. Interactive temporal recurrent convolution network for traffic prediction in data centers. *IEEE Access*, 6:5276–5289, 2018.

[8] S. Xingjian, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[10] X. Shi and D. Yeung. Machine learning for spatiotemporal sequence forecasting: A survey. *arXiv preprint arXiv:1808.06865*, 2018.

[11] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[12] Yin Zhang. Abilene dataset. *URL http://www.cs.utexas.edu/yzhang/research/AbileneTM*, 2011.

[13] K. Xie, L. Wang, X. Wang, G. Xie, G. Zhang, D. Xie, and J. Wen. Sequential and adaptive sampling for matrix completion in network monitoring systems. In *IEEE Conference on Computer Communications (INFOCOM'15)*, pages 2443–2451, 2015.

[14] Z. Hu and J. Luo. Cracking network monitoring in dcns with sdn. In *IEEE Conference on Computer Communications (INFOCOM'15)*, pages 199–207, 2015.

[15] Kun Xie, Can Peng, Xin Wang, Gaogang Xie, and Jigang Wen. Accurate recovery of internet traffic data under dynamic measurements. In *IEEE Conference on Computer Communications (INFOCOM'17)*, pages 1–9, 2017.

[16] H. Zhou, D. Zhang, and K. Xie. Accurate traffic matrix completion based on multi-gaussian models. *Computer Communications*, 102:165–176, 2017.

[17] A. Azzouni and G. Pujolle. Neutm: A neural network-based framework for traffic matrix prediction in sdn. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, 2018.

[18] R. Vinayakumar, K. Soman, and P. Poornachandran. Applying deep learning approaches for network traffic prediction. In *Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on*, pages 2353–2358, 2017.