一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

Traffic Matrix Prediction based on Bidirectional Recurrent Neural Network and Long Short-Term Memory

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
E-mail: † {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.). Measuring all the network traffic is impossible or impractical due to the monitoring resources constraints as well as the dynamic of temporal/spatial fluctuations of the traffic. To this end, a common approach is to solve the traffic matrix interpolation using compress sensing and matrix completion. Besides that, there are some studies exploiting Deep Learning techniques such as Restricted Boltzmann Machine or Recurrent Neural Network to estimate the traffic volume. However, their proposal reveals a poor performance regarding the traffic inference when the measurement data has a highly missing rate.

In this paper, we propose a highly accurate traffic prediction algorithm by leveraging the advantages of Long Short-Term Memory (LSTM) in the time series estimation and modifying the Bidirectional Recurrent Neural Networks (BRNN) in correcting the feeding data. We evaluate our model based on the Abilene Dataset which contains the real traffic matrices. The experiment results show that the proposed approach can achieve significantly better prediction accuracy in term of several metrics such as error ratio, mean absolute error and root mean square error, even when only 30% of the traffic flows in the network are measured.

Keywords traffic prediction, recurrent neural network, long short-term memory.

1. Introduction

The great demand for Internet services (e.g. video streaming, VoIP, etc.) has led to the exponential growth in the traffic. Therefore, having a better knowledge of the network traffic becomes a critically important factor which allows network operators to efficiently perform management tasks such as traffic engineering, capacity planning and quality of service provisioning. The key input of these processes is the traffic matrix (TM), which represents the traffic volume between the origins (i.e. sources) and destinations in the network and provides the essential information about the current network situation. The conventional approaches in determining the traffic matrix are to adopt direct measurement based on the traffic flows collected by built-in functions of routers [1]. In order to reduce the measurement overhead, the flows may be sampled by a certain rate. However, these approaches suffer from many critical limitations including the difficulty in dealing with rapid traffic, a high storage overhead, large bias towards the long flows, and the impossibility in capturing the flow size, etc.

To overcome the limitations of the conventional approaches, Vardi et al. [2] introduced an inference technique (named as *network tomography*) which can deduce the traffic matrix based on routing information and the link loads. However, the tomography struggles with the so-called ill-post problem when the number of flows is far more than the number of monitored links. Based on the tomography method, the authors in [3] [4] [5] exploited the compress sensing and low-ranked matrix completion techniques to achieve better results. However, by leveraging the tomography, the network routing information is implied to remain stable and the traffic matrix is stationary during the measurement interval. Besides that, these approaches addressed only traffic matrix estimation and interpolation problems, and hence they cannot be used in predicting the future traffic.

In recent years, deep learning has been widely applicable to various application domains such as image/video processing, natural language recognition, etc. 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 [7] and [8] exploited the Long Short-Term Memory (LSTM) model for capturing the temporal feature and predicting the network traffic in data centers and cellular networks,

respectively. Unfortunately, all of the existing algorithms proposed so far require the precise historical data as the input of the deep learning model. However, in backbone networks, collecting all the traffic data is impractical due to the resource limitation.

In this paper, we address the problem of modeling and predicting the future network traffic under the lack of precise historical traffic data. We propose a novel deep learning approach which leverages the advantages of LSTM in solving the problem of long-term dependencies of time series data and Bidirectional Recurrent Neural Networks (BRNN) in correcting the feeding data (i.e. the input). Since the prediction accuracy of LSTM 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, we construct a modified model of Bidirectional Recurrent Neural Network for correcting the previous predicted data 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. The contribution of this paper can be summarized as follows:

- To our best knowledge, we are the first one considering imprecise data in the input fed to deep learning model for predicting future traffic. Our approach is different from existing deep learning approaches in time series prediction where the model is fed with the ground-truth.
- We construct a model which combines the forward and backward recurrent neural network. This model is able to correct the input data to improve the accuracy of the future 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 preliminary experiments on real backbone network traffic and compare 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 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 a preliminary experiment result and Section 5 concludes the paper.

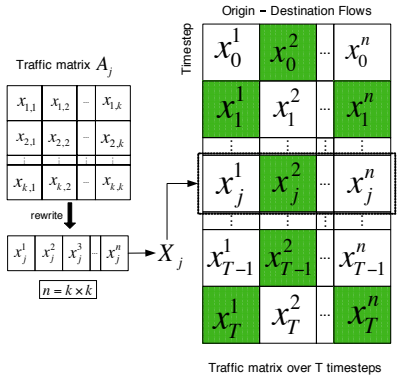


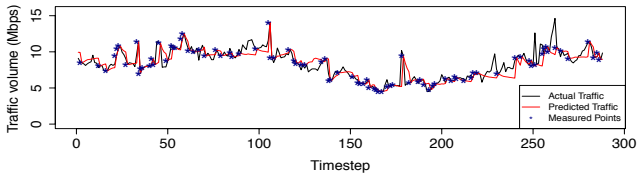
Figure 1: Construct traffic matrix over T timesteps from traffic matrices A_j $(j = 0,..,T)$.

2. Problem description

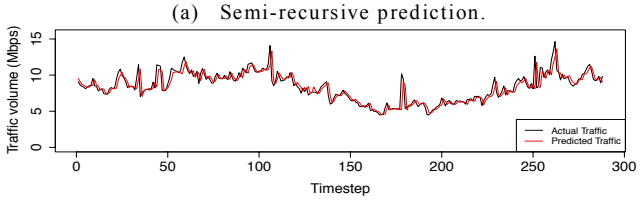
In this section, we present the problem formulation of traffic prediction under the lack of precise input traffic data. Given a network containing k origins and destinations, let A_j be the traffic matrix (with the order of $k \times k$) at the j^{th} timestep. Each element of A_j represents the traffic volume of an origin-destination flow (or an OD flow, for short). In order to examine the temporal feature of the traffic matrix over T timesteps, we rewrite A_j as a row vector X_j. Then, the traffic matrix over T timesteps can be represents as $A = \{X_1, X_2, \dots, X_T\}$ where $X_j = [x_j^1, x_j^2, \dots, x_j^n]$ $(n = k \times k)$, $x_j^i \in X_j$ is the traffic volume of the i^{th} origin-destination flow (OD_i, for short) at the j^{th} timestep (Fig.1). Although the traffic information of a particular flow can be obtained at any time by leveraging new network architectures such as Software Defined Network with a centralized controller (e.g. OpenFlow controller), collecting all the traffic information of the network results in a very high cost (in terms of computation, bandwidth, etc.). Therefore, for reducing the monitoring overhead, at each timestep, we only measure a part of the traffic flows in the network. In Fig.1 the green elements stand for the direct measurement values of traffic volume while the others are the values obtained by using traffic prediction.

Let l be a predefined parameter representing the time lags before the current timestep t. Suppose that the historical data of flow OD_i at timesteps $t, t-1, \dots, t-l$ are $x_t^i, x_{t-1}^i, \dots, x_{t-l}^i$, respectively, then the traffic prediction problem is to estimate the traffic volume y_{t+1}^i at the timestep $t+1$, by leveraging the information of $x_t^i, x_{t-1}^i, \dots, x_{t-l}^i$. This problem can be mathematically formulated using semi-recursive model as follows.

$$x_j^i = \begin{cases} o_j^i & if\ m_j^i = 1 \\ y_j^i & otherwise \end{cases} (j = t-l, \dots, t)$$
$$y_{t+1}^i = f(x_t^i, x_{t-1}^i, \dots, x_{t-l}^i) \quad (1)$$



(a) Semi-recursive prediction.



(b) Direct prediction.

Figure 2: Traffic prediction using LSTM/RNN.

Where o_j^i and y_j^i are the directly measured and predicted traffic volumes of flow OD_i at timestep j ($\forall j = t-l, \dots, t$), respectively. The binary variable $m_j^i = 1$ indicates that flow OD_i will be measured at timestep j; otherwise, the value of traffic volume is filled up by the prediction result.

3. Proposed Prediction Model

In this section, we first describe two challenges in solving the semi-recursive traffic prediction problem in Section 3.1. These two challenges then are addressed in Section 3.2 and 3.3.

3.1. Motivation

In order to deploy the semi-recursive traffic prediction model, we face two important challenges. The first one is the accumulative error in the historical traffic data. We have conducted an experiment to figure out the impact of imprecise input data on the prediction results (Fig.2). Fig.2a shows the traffic prediction result in which the future traffic is estimated by using both precise input data (obtained by direct measuring) and imprecise input data (the previous predicted results). As shown, the errors in the input have been accumulated over timesteps and lead to the downgrade in the prediction results. While in Fig.2b, thanks to the preciseness of the input traffic data (since all the input data are directly measured), we can capture the trend of the flow and achieve a high accuracy in the forecasted traffic volume. Therefore, in order to alleviate the accumulative error in the input data, our idea is to add a preprocessing on the imprecise input data before feeding it into the prediction model. Specifically, we propose the novel deep learning model based on bidirectional recurrent neural network (the details will be shown in Section 3.2).

The second challenge is how to determine which flows to be measured at the next timestep. As mentioned above, we can only measure a part of flows, thus choosing appropriate flows to measure becomes a significant factor

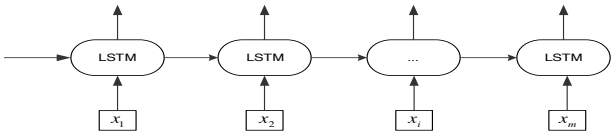


Figure 3: The unfolded RNN with LSTM unit.

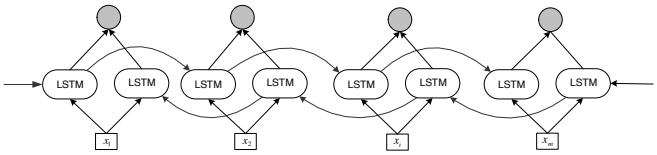


Figure 4: The BRNN model.

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 is kept to be approximately the same. However, this method may not effective since the network flows are dynamic and have various temporal fluctuations. 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 every flow (see the details in Section 3.3).

3.2 Traffic prediction and data correction

For temporal modeling and predicting the traffic of origin-destination flows, we propose to use Recurrent Neural Network (RNN), which is a generation of the feed forward neural network for modeling sequence data. However, while the network traffic has distinct temporal dependencies and the previous data has a long-term impact on the current traffic, the standard RNN is difficult to model the long-term dependencies [8]. To this end, we replace the standard RNN unit by Long Short-Term Memory (LSTM) [9], which is one of the most successful methods in capturing long-term temporal dependencies. An unfolded model of the RNN network with LSTM unit is shown in Fig.3.

Besides that, following the experiment in Section 3.1 (Fig.2a), we observe that the prediction results become more accurate if the input sequence contains more precise data. Therefore, in order to correct the imprecise data, our idea is to leverage the information of the future steps' input data which may include more precise data. Specifically, we construct a backward network (in which the input data is fed in the reverse order) and use its output to correct the imprecise data. Our approach is motivated by the Bidirectional Recurrent Neural Network (BRNN) which was introduced in [10]. Thanks to the backward network (Fig.4), BRNN can be trained using all available input information in both the past and future of a specific time

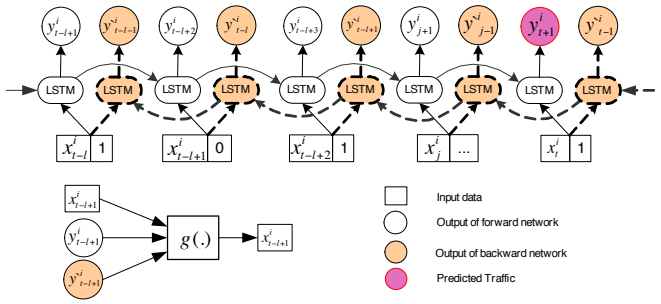


Figure 5: The proposed model in which forward and backward networks are separated. The imprecise input x^i_{t-l+1} is updated using the outputs from both forward and backward network.

frame, and hence, it has been applied in wide range of applications in Natural Language Processing (e.g. Speech Recognition, Language Translation, Speech to Text, etc.).

Since the standard BRNN is only appropriate to solve the problem where the complete input (e.g. the whole sentence) are available before processing, we have to modify the BRNN in order to apply to our model as follows. The outputs from each step of the network are not aggregated while the fed inputs are the same for both forward and backward network (Fig.5). Moreover, besides the traffic data, we also add another binary feature to the input where "1" indicates the measured value and "0" is the predicted value. Specifically, considering the prediction process at timestep t of OD_i flow, let we denote $\{x^i_t, x^i_{t-1}, \dots, x^i_{t-l}\}$ as the input sequence, $\{m^i_t, m^i_{t-1}, \dots, m^i_{t-l}\}$ as the measured binary variable, and $\{y^i_{t+1}, y^i_{t1}, \dots, y^i_{t-l+1}\}$, $\{y'^i_{t-1}, y'^i_{t-2}, \dots, y'^i_{t-l-1}\}$ as the outputs of the forward and the backward network, respectively. Then the output of backward network (i.e., which may be processed based on more precise input) is used to update the current input as follow.

$$x^i_j = g\left(x^i_j, y^i_j, y'^i_j\right) = \alpha \times x^i_j + \beta \times y^i_j + \gamma \times y'^i_j \quad (j = t-l+1, \dots, t-1) \quad \alpha + \beta + \gamma = 1 \tag{2}$$

The parameters α, β, γ are defined as the *confidence factors* of the previous timestep prediction, the forward and backward RNNs prediction, respectively. We will discuss more details about these confidence factors by first define two terms named forward loss and backward loss. The forward and backward loss of a flow (denoted as $l^f_{i,t}$ and $l^b_{i,t}$, respectively) are defined to assess the performance of the forward and backward RNNs after each current timestep t. The $l^f_{i,t}$ and $l^b_{i,t}$ are calculated based on the differences between the output of forward, backward RNN and the observed input traffic as follows.

$$l^f_{t,i} = \begin{cases} +\infty & if \sum_{j=t-l}^{t} m^i_j = 0 \\ \sqrt{\sum_{j=t-l}^{t} m^i_j * \left(y^i_j - x^i_j\right)^2} & otherwise \end{cases}$$

$$l^b_t = \begin{cases} +\infty & if \sum_{j=t-l}^{t} m^i_j = 0 \\ \sqrt{\sum_{j=t-l-1}^{t-1} m^i_j * \left(y'^i_j - x^i_j\right)^2} & otherwise \end{cases} \tag{3}$$

According to the forward and backward loss after each prediction, we calculated the confidence factors α, β, and γ by using the following formulas:

$$\alpha = 1 - \eta^i_t;\ \beta = \frac{l^b_{i,t}(1-\alpha)}{{l_{i,t}}^f + l^b_{i,t}};\ \gamma = \frac{l^f_{i,t}(1-\alpha)}{l^f_{i,t} + l^b_{i,t}} \tag{4}$$

where $\eta^i_t = \begin{cases} \frac{\sum_j m^i_j}{l} & if\ \sum_j m^i_j > 0 \\ \epsilon & otherwise \end{cases} (j = t-l, \dots, t)$ is the monitored ratio of the input sequence of OD_i flow at timestep t (l is the length of the input sequence). In case all the data involved in that sequence are the predicted values, then $\eta^i_t = \epsilon \approx 0$.

3.3 Determining the monitored flow set

Our main idea is that at each timestep t, for each flow OD_i we define a weight w^i_t which indicates the priority of OD_i in being chosen to be monitored at the next timestep (i.e., timestep $t+1$). The flows with lower weights will be more likely to be chosen. In the following, before going to the detail 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 3.1, we notice that the longer the non-monitored periods, the larger the errors 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 OD_i, we define a term named "*consecutive loss*" (denoted as $l^c_{i,t}$), which indicates how long OD_i has not been monitored. Specifically, $l^c_{i,t}$ is the number of timesteps from when OD_i was last monitored till the current timestep t. The weight w^i_t then is designed to be inversely proportional with $l^c_{i,t}$.

Moreover, we also observe that the prediction error becomes larger when the predicted timestep goes farther

from the measured points (i.e., the timestep where the traffic is measured directly). The reason is because the error in the prediction results (which have been used as the input data) has been accumulated overtime. Therefore, in order to alleviate this accumulative error, the flows with high backward loss and forward loss (defined in Section 3.2) 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 OD_i, we define the term flow fluctuation (denote as $s_{i,t}$) which is the standard deviation of the traffic volume of the flow OD_i from timestep $t-l$ to t. The weight w_t^i then is designed to be inversely proportional with $s_{i,t}$.

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

$$w_t^i = \frac{1}{l_{i,t}^f + l_{i,t}^b + l_{i,t}^c + s_{i,t}} \quad (5)$$

where $l_{i,t}^f$, $l_{i,t}^b$, $l_{i,t}^c$ and $s_{i,t}$ are the forward, backward, consecutive loss and the flow fluctuation of OD_i at timestep t, respectively.

At the end of the timestep t, the weights of all flows are calculated, and the first d flows with the lowest weights are chosen to be measured at the next timestep (i.e., timestep $t+1$) (d is the maximum number of flows can be measured depend on the network monitoring policy).

4. Performance evaluation

4.1. Dataset and metrics

We evaluate the performance of our proposed approach by conducting preliminary experiments on the real dataset Abilene [11]. The dataset contains the real trace data from the backbone network located in North America. Abilene dataset, which includes averages over 5 minutes interval of 144 origin-destination flows from March 1st to September 11st, 2004, was widely used for performance evaluation in many traffic matrix completion and prediction studies [3] [4] [5]. In our experiments, we separated the dataset into 70% for training and 30% for testing. We compare our results with ARIMA [12] and the standard RNN model. ARIMA is one of the most popular linear models for time series forecasting and the widely used method for time series analysis [13] [14]. The used metrics include Error Ratio (E_r), Mean Absolute Error (MAE) and Root Mean Square Error ($RMSE$) have been defined as follows.

$$E_r = \frac{\sqrt{\sum_{i,j}\left(o_j^i - x_j^i\right)^2}}{\sqrt{\sum_{i,j}(o_j^i)^2}} \;\forall\, m_j^i = 0;$$

$$MAE = \frac{1}{N}\sum_{i,j}\left|o_j^i - x_j^i\right| \;;\; RMSE = \sqrt{\frac{1}{N}\sum_{i,j}\left(o_j^i - x_j^i\right)^2} \quad (6)$$

$$(j = 0,..,N)$$

where o_j^i is the actual traffic volume at timestep j of flow OD_i, and N is the size of the testing dataset.

4.2. Performance comparison

We conducted two experiments. In the first one, at each timestep, we measure 30% flows of the network flows. In the second one, we let consecutive measurements over 4 hours all loss from 10 am in everyday and then calculate the errors of each model.

The results of the first experiment are plotted in Fig. 6 and 7. Fig.6 compares the traffic volume predicted by our algorithm and the actual values of a flow chosen randomly. As shown, our algorithm can capture the traffic trend smoothly. Specifically, the largest gap between the predicted values and the actual value is …. Recalling the results shown in Figure 2a, it can be seen that our input data correction algorithm has improved the performance significantly.

Fig.7 shows the performance comparison among ARIMA, standard RNN and our proposed approach in terms of *Error Ratio*, *MAE* and *RMSE*. We can see that our approach outperforms ARIMA and the standard RNN model in terms of all the metrics. Specifically, the proposed approach results in the *Error Ratio* that is less than 48.3% that of the standard RNN. The MAE and *RMSE* of the proposed approach are less than 24.2% and 62.9% that of the standard RNN, respectively. Furthermore, comparing with the ARIMA, our approach achieves the *Error Ratio*, *MAE* and *RMSE* that are less than 59.5%, 90,9% and 71.5% of ARIMA.

The results of the second experiment are presented in Fig.8. As shown, while ARIMA is affected significantly by the consecutive loss (i.e., increasing more than 105, 9 and 106 times in term of *Error Ratio*, *MAE* and *RMSE*, respectively, compared to the results of the first

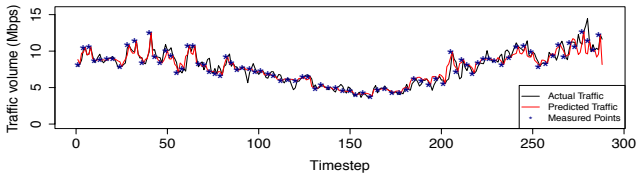


Figure 6: Prediction results and actual values of a random flow.

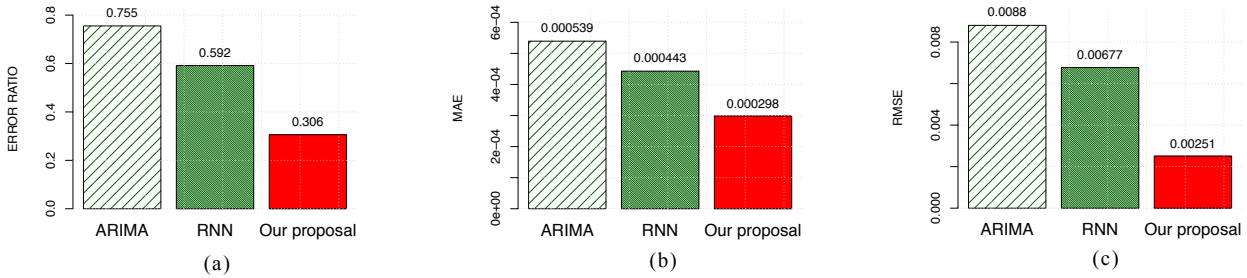


Figure 7: Prediction errors without of consecutive loss.

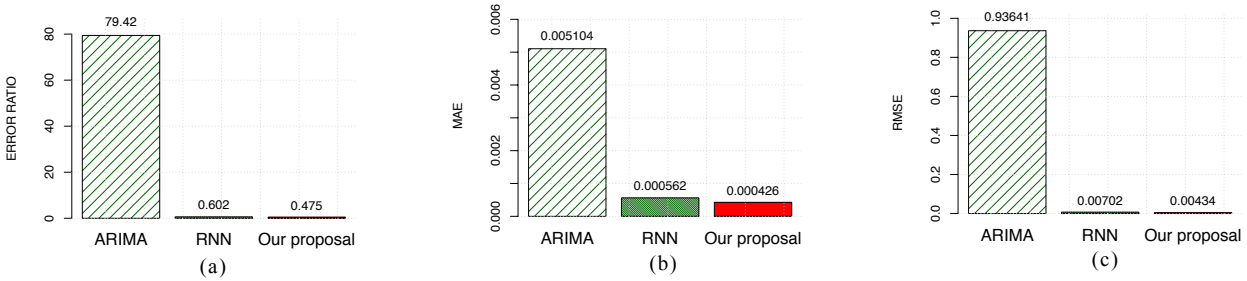


Figure 8: Prediction errors under consecutive loss.

experiment), our proposed approach and standard RNN are not. Our proposed approach shows the superiority in dealing with the imprecise as the *Error Ratio* always remains under 0.5 (Fig.8).

5. Conclusion

In this paper, we defined the semi-recursive traffic prediction problem where the future traffic is estimated under the lack of precise historical data. We proposed the novel deep learning model and techniques which leverage the forward and backward Recurrent Neural Network for input data correction and monitored flows determination after each timestep. We conducted preliminary experiments for evaluating our proposed approach. The results demonstrated that the proposed approach can achieve better performance compared with the well-known methods in time series analysis field.

Although our model is good at capturing the long-term dependencies temporal feature and predicting the future traffic even in high missing observation data, we have abandoned the relation between the *OD* flows in the network when treating the flows independently. In the future work, we will examine the relation between the flows in the traffic matrix and capture not only the temporal but also the spatial feature of the traffic matrix.

References

[1] "Cisco Netflow," [Online]. Available: http://www.cisco.com/go/netflow.

[2] Vardi and Yehuda, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American statistical association,* vol. 91, no. 433, pp. 365-377, 1996.

[3] M. Malboubi, L. Wang, C.-N. Chuah and P. Sharma, "Intelligent sdn based traffic (de) aggregation and measurement paradigm (istamp)," in *IEEE INFOCOM*, 2014.

[4] 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 INFOCOM*, 2015.

[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 INFOCOM*, 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,* vol. 76, pp. 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,* vol. 6, pp. 5276 - 5289, 2017.

[8] 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 INFOCOM*, 2017.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[10] S. Mike and P. K. K, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing,* vol. 45, no. 11, pp. 2673-2681, 1997.

[11] "Abilene dataset," [Online]. Available: http://abilene.internet2.edu/observatory/data-collections.html.

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

[13] B. Zhou, D. He, Z. Sun and W. H. Ng, "Network traffic modeling and prediction with ARIMA/GARCH," in *HET-NETs*, 2005.

[14] Y. Shu, M. Yu, J. Liu and O. W. Yang, "Wireless traffic modeling and prediction using seasonal ARIMA models," in *IEEE International Conference on Communications*, 2003.