Applying Deep Learning Approaches for Network Traffic Prediction

Vinayakumar R*, Soman KP* and Prabaharan Poornachandran†
*Centre for Computational Engineering and Networking (CEN), Amrita School of Engineering, Coimbatore
†Center for Cyber Security Systems and Networks, Amrita School of Engineering, Amritapuri
Amrita Vishwa Vidyapeetham, Amrita University, India
Email: vinayakumarr77@gmail.com

***Abstract*—Network traffic prediction aims at predicting the subsequent network traffic by using the previous network traffic data. This can serve as a proactive approach for network management and planning tasks. The family of recurrent neural network (RNN) approaches is known for time series data modeling which aims to predict the future time series based on the past information with long time lags of unrevealed size. RNN contains different network architectures like simple RNN, long short term memory (LSTM), gated recurrent unit (GRU), identity recurrent unit (IRNN) which is capable to learn the temporal patterns and long range dependencies in large sequences of arbitrary length. To leverage the efficacy of RNN approaches towards traffic matrix estimation in large networks, we use various RNN networks. The performance of various RNN networks is evaluated on the real data from GÉANT backbone networks. To identify the optimal network parameters and network structure of RNN, various experiments are done. All experiments are run up to 200 epochs with learning rate in the range [0.01-0.5]. LSTM has performed well in comparison to the other RNN and classical methods. Moreover, the performance of various RNN methods is comparable to LSTM.**

***Index Terms*—Network traffic matrix, Prediction, Deep Learning**

I. Introduction

In modern society, the Internet and its applications has become a primary communication tool for all types of users to carry out daily activities. This can create a large volume of network traffic. Understanding the traffic matrix is essential for individual network service providers mainly due to the reason that their inferences are enormous. Traffic matrix provides the abstract representation of the volume of traffic flows from all possible sets of origin to a destination point in a network for definite time interval. The source and destination point can be routers, points-of-presence, internet protocol (IP) prefixes and links [1]. Traffic matrix facilitates network service providers to make various network management decisions such as network maintenance, network optimization, routing policy design, load balancing, protocol design, anomaly detection and prediction of future traffic trends [2], [3]. A detailed study on network traffic matrix is reported in [4]. A network service provider has to know the future trends of network parameters, routers and other devices information in order to proceed with the traffic variability. To deal with the problem of predicting the future trends of network parameters, routers and other devices information in real time network, prediction approaches have been employed.

Most commonly used approaches to obtain a traffic matrix are tomogravity [1], principal component analysis (PCA), route change [5]. These techniques have their own disadvantages [6]. Network tomography is the well-known technique used to get a traffic matrix. This facilitates to know the link counts and routing information. This generates an ill-posed feature in estimating the traffic matrix [6]. Ideally, the network traffic parameters prediction is estimated based on their statistical characteristics mainly because they exists extreme association relationships between the sequential ordered values. The statistical features have changed a lot in the current complex and diverse network architecture and its applications [7]. These new statistical features are not the right approach for current network traffic information due to the fact that they were inadequately modeled by Poisson and Gaussian models [5]. This paper is more devoted to network traffic prediction by using the existing data from GÉANT backbone networks.

There are various methods exist for network traffic forecasting. Generally, these can be classified in to linear and non-linear prediction. The most commonly used linear mechanism is ARMA/ARIMA model [8], [9], [10] and HoltWinters algorithm [8]. The most commonly used non-linear mechanism is neural networks [3], [11], [12]. In [13] claimed that the non-linear approach based traffic prediction was most appropriate. The performance of various linear approaches such as ARMA, ARAR and HW and non-linear approach such as neural networks was evaluated. In all experimental settings, the non-linear technique has performed well in comparison to the linear approaches. Generally, the best forecasting approach is chosen based on considering the factors like chracteristics of the traffic matrix, less mean suwarred error and less compuational cost. [13] showed that the Feed Forward neural network (FFN) predictor with multiresolution learning approach as considered as best forecasting technique with careful consideration of precision and model complexity.

Recurrent neural network (RNN) is an enhanced model of traditional FFN [14]. This contains a self-recurrent loop which facilitates to carry out information from one time step to another. This characteristic of RNN made suitable for time series and sequence modelling tasks. LSTM is a type of RNN which was proposed to alleviate the vanishing and exploding gradient issue of traditional RNN [15]. With the aim to reduce the computational cost of LSTM, [16] introduced GRU. [17] showed RNN with ReLU and proper initialization of identity matrix to a recurrent weight matrix is capable to perform closer in the performance of LSTM. This was shown by evaluating the 4 standard experiments. This type of RNN is called as identity-RNN (IRNN). The family of RNN techniques has performed well in various long-standing tasks like handwritten recognition, speech recognition, various tasks related to natural language processing (NLP) [18]. Towards transforming the efficacy of family of RNN techniques towards the large scale traffic matrix prediction, in this paper we apply and evaluate the effectiveness of various RNN networks on GÉANT backbone networks.

The paper is structured as follows. Section II discusses the related work. Section III provides the background information of RNN networks and traffic matrix prediction. Section IV discusses the details of data sets and experiments related to network parameters and network structure identification. Section V provides the detailed evaluation results. Finally, the conclusion and future work directions are placed at Section VI.

II. Related work

There are different methods exist for prediction of network traffic. [13] discussed the various linear models such as ARMA, ARAR and HW and non-linear approach such as neural network with multi-resolution learning. In all configurations of experiments, non-linear model was performed well in comparison to all the other linear models. [19] proposed FARIMA predictors using RNN and stable to capture the non-gaussian of the self-similar traffic. [20] was done comparative study of various prediction methods such as mode prediction, key element with matrix prediction and prediction of principal component. prinicipal component prediction was resulted in less average prediction error for origin-destination (OD) flows. TMP-KEC was decreased the prediction error for most important elements and total matrix.

III. Background

This section discusses the idea behind the traffic matrix prediction and the mathematical details of artificial neural network (ANN).

A. Artificial neural network (ANN)

An Artificial neural network (ANN) is a computational mechanism in machine learning that is influenced by the attributes of biological neural networks. This forms a directed graph in which nodes represent biological neurons and edges represent synapses. Feed forward networks (FFN) and recurrent neural network (RNN) are two types of ANN.

B. Feed forward network (FFN)

A feed-forward network (FFN) forms a directed graph without formation of a cycle, as shown in Fig 1. The directed graph consists of a set of neurons, called as units

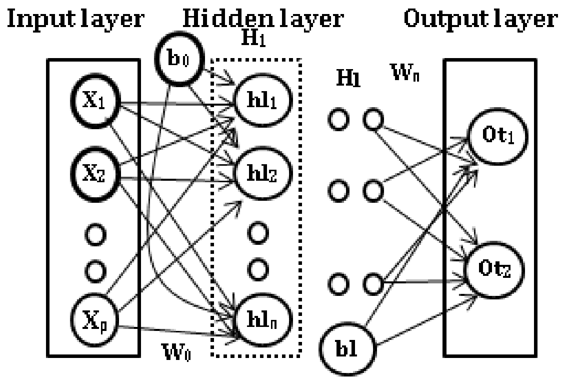

Fig. 1: Architecture of Multi-layer perceptron which has inputs $x = x_1, x_2, \cdots, x_{p-1}, x_p$ and outputs $ot = ot_1, ot_2$, all connections and hidden layers and its units are not shown.

in mathematical terms and these neurons are connected by an edge. Multi-layer perceptron (MLP) is one of the well-known algorithms of FFN. It contains an input layer, output layer and the number of hidden layer is parameterized. FFN follows unidirectional mechanism to pass information from one layer to the others. So the weights in the hidden layers are not shared. MLP is a parameterized function that can be defined as $ot : R^i \times R^j$ where i and j are the size of the input vector $x = x_1, x_2, \cdots, x_{i-1}, x_i$ and output vector $ot = ot_1, ot_2, \cdots, ot_{j-1}, ot_j$ respectively. Each hidden layer (hl_i) computation is defined mathematically as follows

$$hl_i(x) = f({w_i}^T x + b_i) \tag{1}$$

$hl_i : R^{d_i-1} \to R^{d_i}$, $f : R \to R$, $w_i \in R^{d \times d_{i-1}}$, $b \in R^{d_i}$, and f is either logistic sigmoid or hyperbolic tangent non-linear activation function. Logistic sigmoid have values between [0, 1] whereas [1,-1] range of values for hyperbolic tangent.

$$sigmoid = \sigma(x) = \frac{1}{1+e^{-z}} \tag{2}$$

$$hyperbolic \tan gent = \tanh(z) = \frac{e^{2z}-1}{e^{2z}+1} \tag{3}$$

If a network consists of l layers then the combined representation of them can be generally defined as

$$hl(x) = hl_l(hl_{l-1}(hl_{l-2}(\cdots(hl_1(x))))) \tag{4}$$

$$d_0 = i \tag{5}$$

MLP is a parameterized function, thus optimal parameters have to be selected to achieve an acceptable detection rate. These parameters are dynamic in nature, which means it changes with the data. Even, finding optimal parameters to a certain data is considered as a research study. We use mean squared error as loss function in MLP to achieve a better detection rate. Loss function is a difference between the expected (ep) and predicted value (pr) and it is defined with a list of corrected input-output set $it\_ot = (s_1, t_1), (s_2, t_2), \cdots, (s_n, t_n)$ as follows

$$loss(s_n, t_n) = \frac{1}{n} \sum_{i=1}^{n} d(t_i, f(s_i)) \tag{6}$$

Given a loss function, $Train_{i\_o}(\theta) \equiv J_{i\_o}(\theta) = \frac{1}{n} \sum_{i=1}^{n} d(t_i, f_\theta(s_i))$ where $\theta = (w_1, b_1, \cdots, w_k, b_k)$ and we need to find the appropriate value of $\theta \in R^d$ to minimize the loss function $J_{i\_o}(\theta)$. This includes the estimation of $f_\theta(s_i)$ and $\nabla f_\theta(s_i)$ at the cost $|it\_ot|$

$$\min_{\theta} \ J(\theta) \tag{7}$$

Gradient descent is one of the most commonly used techniques to minimize the loss function. It follows a repeated estimation of a gradient to update its parameters. A single update on training data is mathematically defined as follows.

$$\theta^{new} = \theta_{old} - \alpha \nabla_\theta J(\theta) \tag{8}$$

α is the learning rate, it is comparatively very small. There is no standard approach to select a particular value for learning rate. The optimal value is chosen by following the trial and error mechanism during training phase. Learning rate has a direct impact on the speed of the optimization, lesser the learning rate consumes more time to its convergence. Traditional machine learning classifiers follows the gradient descent approach in which the problem surface is convex. In the case of neural networks the problem space is non-convex and a detailed analysis of non-convex training mechanism is discussed by [21]. They reported that training deep networks maps the data to a high dimensional space from a layer to another layer. Thus the critical points might grow exponentially. Sometimes critical points might end up in saddle point or a local optimum. To avoid this, we use backpropogation or backward propogation of errors algorithm. Stochastic gradient descent (SGD) is a mechanism used by many deeper networks. It facilitates to estimate the gradient on a random selection of very few training samples by forming a mini-batch. As a result, SGD is very faster in comparison to GD in reaching the minimum , while this will not be an efficient approximation in comparison to standard gradient method. The deeper networks of this paper have used SGD and its update rule is given by

$$\theta^{new} = \theta_{old} - \alpha \nabla_\theta J(\theta; s^{(i)}, t^{(i)}) \tag{9}$$

where $(s^{(i)}, t^{(i)})$ are input-output set of training samples. Let us take three-layer network and formulate the problem in mathematical way.

$$hl_1 = g1(x; \theta_1) \tag{10}$$

$$hl_2 = g2(hl_1; \theta_2) \tag{11}$$

$$pr = g3(hl_2; \theta 3) \tag{12}$$

where $i\_o = (s_1, t_1), (s_2, t_2), \cdots, (s_n, t_n)$ and to minimize the error between the predicted value (pr) and an expected value (ep), we use loss function.

To compute partial derivatives for the above case, we use the chain rule

$$\frac{\partial L}{\partial pr} = 2 * (pr - ep) \tag{13}$$

$$\frac{\partial L}{\partial hl_2} = \left(\frac{\partial L}{\partial pr}\right) * \left(\frac{\partial pr}{\partial hl_2}\right) \tag{14}$$

$$\frac{\partial L}{\partial hl_1} = \left(\frac{\partial L}{\partial hl_2}\right) * \left(\frac{\partial hl_2}{\partial hl_1}\right) \tag{15}$$

$$\frac{\partial L}{\partial \theta_3} = \left(\frac{\partial L}{\partial pr}\right) * \left(\frac{\partial pr}{\partial \theta_3}\right) \tag{16}$$

$$\frac{\partial L}{\partial \theta_2} = \left(\frac{\partial L}{\partial hl_2}\right) * \left(\frac{\partial hl_2}{\partial \theta_2}\right) \tag{17}$$

$$\frac{\partial L}{\partial \theta_1} = \left(\frac{\partial L}{\partial hl_1}\right) * \left(\frac{\partial hl_1}{\partial \theta_1}\right) \tag{18}$$

C. Recurrent neural network (RNN)

Recurrent neural network (RNN) is an enhanced model of traditional FFN, as shown in Fig 2. A RNN unit contains a self-recurrent connection that helps to carry out information across time-steps. This characteristic of RNN facilitates to learn the temporal dependencies. This computational flow of RNN is mathematically formulated as follows.

$$hl_t = SG(w_{xhl} x_t + w_{hlhl} hl_{t-1} + b_{hl}) \tag{19}$$

$$ot_t = sf(w_{hlot} hl_t + b_{ot}) \tag{20}$$

where w represents weight matrices, b terms denotes bias vectors, SG and SF is an element wise non-linear $sigmoid$, $softmax$ activation function, hl acts as a short-term memory to the RNN network. The RNN network is typically unfolded across time-steps and the process of applying backpropogation on the unfolded RNN is called as backpropogation through time (BPTT). RNN with BPTT generates vanishing and exploding gradient issue in backpropogating error gradient across many time-steps [22]. To alleviate, [15] introduced LSTM. LSTM contains memory block as shown in Fig 2 that helps to reduce vanishing and exploding gradient issue. A memory block is a complex processing unit that contains input gate, output gate, forget gate and one or more memory cells, as shown in Fig 2. A memory cell acts as a short term memory to store dependencies across time-steps with the control from the different gates such as input gate, forget gate and output gate.

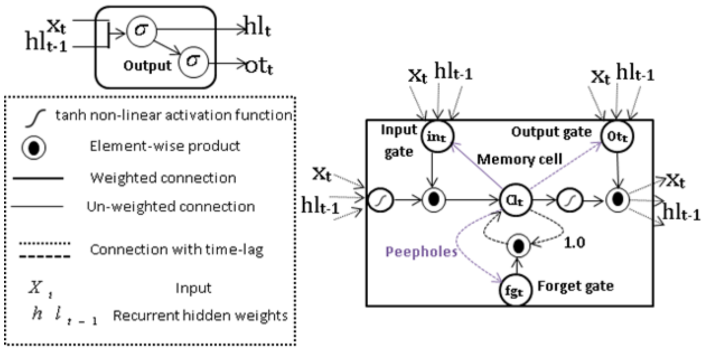

Fig. 2: Architecture of multi-layer perceptron which has inputs $x = x_1, x_2, \cdots, x_{p-1}, x_p$ and outputs $ot = ot_1, ot_2$, all connections and hidden layers and its units are not shown.

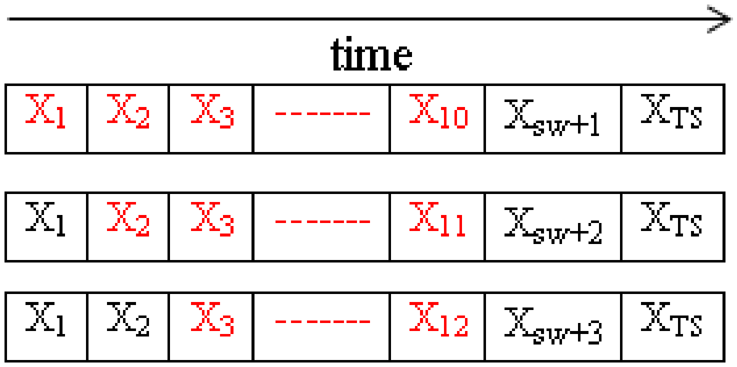

Fig. 3: Sliding window

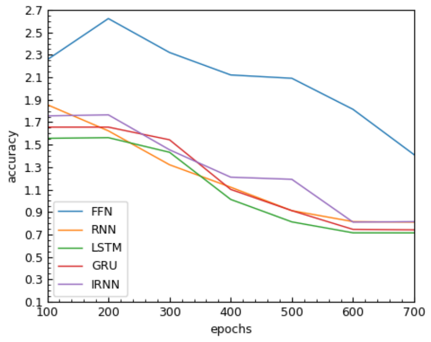

Fig. 4: MSE across hidden layers in the range [100-700]

IV. Experiments

All experiments of RNN and LSTM network are trained using BPTT on Graphics processing unit (GPU) enabled TensorFlow [23] computational framework in Ubuntu 14.04. Experiments of shallow networks are done using Scikit-learn [24].

A. *Description of traffic matrix data set*

To evaluate the effectiveness of various RNN networks, FFN and other classical method, we use the publically available and most well-known data set such as GÉANT backbone networks [25]. The GÉANT network includes 23 peer nodes and 120 undirected links [3]. 2004-timeslot traffic data is sampled from the GÉANT networks by 15 minute time interval. From the 10,772 traffic matrices, we choose arbitrarily 1200 traffic matrices. Each traffic matrix TM is transformed to vector of size 23*23 = 529 TMV . These vectors are concatenated and formed a new traffic matrix TM of size 1200*529. The traffic matrix TM is randomly divided in to two matrices such as training TM_{train} 900*529 and TM_{test} 300*529. There are various ways exist for predicting the future traffic. One way is to feed the vectors TMV in TM to prediction algorithms and predict one value of TMV at a time. This method was not correct due to the fact that the OD traffic is dependent on other ODs [26]. Thus capturing the patterns exist in previous traffic can enhance the performance of prediction of traffic matrix. We use the slide window to create a training data set, as shown in Fig 3. The number of gray color boxes is the length of the slide window with time-slots TS and OD pair. The values exist in right side of the slide window is the predicted value by prediction algorithms. We obtain $TS - SW + 1$ training records by sliding the window for the OD pair with TS time slots.

B. *Hyperparameter selection*

As LSTM network is a parameterized function, finding an optimal parameter have direct impact on minimizing the mean squared error. Initially, we divide the training data 900*529 into two parts such as 700*529 for training and 200*529 for validation. We use various experiments for FFN, RNN, LSTM, GRU and IRNN with the number of units and hidden layer, and learning rate. We train three trails of experiments for the units 100, 200, 300, 400, 500, 600 and 700 with 1 layer. All experiments are run for 200 epochs. The performance of each trained model associated with each units of FFN, RNN, LSTM, GRU and IRNN network is evaluated on the validation data, shown in Fig 4. An experiment with 600 units is performed well in FFN, RNN LSTM, GRU and IRNN network. Recurrent networks have performed well in comparison to the FFN. Moreover, LSTM has performed well in comparison to the RNN network. The performance of both GRU and IRNN is comparable to the LSTM networks. However, the computational complexity is more with LSTM in comparison to the GRU networks. The primary reason may be due to the fact that, LSTM can store and update the time dependencies across various time steps. Next, we run two trails of experiments across learning rate in the range [0.01-0.5] for FFN, RNN LSTM, GRU and IRNN network. The performance is shown in Fig 5. In most of the cases, LSTM has performed well in comparison to the RNN networks. The performance of both GRU and IRNN netwpork is comparable to the LSTM. To select an appropriate network structure for IRNN/GRU/LSTM/RNN/FFN network, we used the following network topologies

[3]http://www.geant.net/

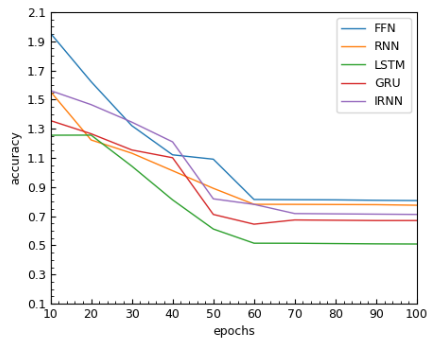

Fig. 5: MSE across hidden layers in the range [100-700]

1) FFN/RNN/LSTM/GRU/IRNN 1 layer
2) FFN/RNN/LSTM/GRU/IRNN 2 layer
3) FFN/RNN/LSTM/GRU/IRNN 3 layer
4) FFN/RNN/LSTM/GRU/IRNN 4 layer
5) FFN/RNN/LSTM/GRU/IRNN 5 layer
6) FFN/RNN/LSTM/GRU/IRNN 6 layer

Two trails of experiments are run for each network structures with number of units 500 and learning rate 0.1. All experiment are run till 100 epochs. The best performed model of each network structure over epochs is shown in Fig 5. Experiments with recurrent networks have performed well in comparison to the traditional FFN. Experiments in all network structures, LSTM has performed well in comparison to the RNN network.

C. Proposed architecture

The LSTM architecture for predicting the future traffic matrix is shown in Fig 6. In input layer, we pass traffic matrix by using the sliding window approach to hidden recurrent LSTM layers. A hidden recurrent LSTM layer contains one or more memory blocks. A memory block is a complex processing unit consists of one or more memory cell and a set of multiplicative gating units such as input and output gate. A memory block is a primary unit that houses the information across time-steps. It has a built-in self-connection called constant error carousel (CEC) value as 1. It will be triggered when it doesn't receive any value from the outside signal. The adaptive multiplicative gating units control the states of a memory block across time-steps. An entry or deny of an input flow of a cell activation to a memory cell is controlled by an input gate. The output state of a memory cell to other nodes is controlled by an output gate. An additional component is added to a memory block, called forget gate [27] instead of CEC due to the fact that the internal values of a memory cell could increase without any constraints. An LSTM network with forget gate facilitate to forget or remember its previous state values. This has been used as standard component in recent LSTM architectures. Furthermore, peephole connection is added from internal states of a memory cell to all its gates to learn the precise timing of the outputs [28]. The LSTM network is trained using the aforementioned technique, such as BPTT. A memory cell has

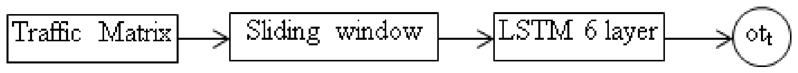

Fig. 6: Architecture of proposed LSTM architecture. Innner units and thier connections are not shown

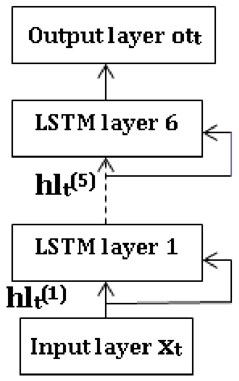

Fig. 7: Stacked LSTM

peephole connections to all of its gates that learn the precise timing of inputs with respect to the internal states. Later, many variants of LSTM introduced. LSTM network have capability in what to learn, forget across many time steps. As a result LSTM learns the long-range temporal context in sequence data modeling across many time-steps. The output of hidden recurrent layer is passed to the feed forward network layer. It use *linear* activation function for prediction with MSE for estimating the error. The staked layer LSTM, is shown in Fig 7. We use more than one hidden recurrent LSTM layer instead of increasing the memory blocks. The primary reason is that, adopting hidden recurrent LSTM layer is most suitable. This maps the data to high dimensional space and passes through non-linear recurrent layers to capture the information related to the time domain.

V. Evaluation Results

Based on the aforementioned hyperparamter approach, we select the 5 layer with 500 units and learning rate 0.1 for all networks such as IRNN/GRU/LSTM/RNN to evaluate the performance on the testing data set. We train 3 models for each FFN/RNN/LSTM/GRU and IRNN using the training data of size 900*529. The performance of the trained model is evaluated on the test data of size 300*529. The detailed result is reported in Table I.

TABLE I: Prediction performance

Algorithm	MSE
FFN	0.091
RNN	0.067
LSTM	0.042
GRU	0.051
IRNN	0.059

VI. Conclusion

This paper has discussed data preprocessing and feeding techniques for RNN and it evaluated the effectiveness of various RNN architectures for network traffic matrix prediction using GÉANT backbone networks. In real time network data sets, LSTM has performed well in comparison to the other RNN, FFN and classical methods. However, the performance of the both GRU and IRNN techniques is comparable to LSTM. Moreover, the GRU network has took less computational cost in comparison to the LSTM networks. Overall, the proposed method has achieved the best performance by accurately predicting the traffic matrix.

With concerning on computational cost, we are not able to train more complex architecture. The reported results can be further improved by using high computation cost architecture. The complex network architectures can be trained by using advanced hardware and following distributed approach in training that we are incompetent to try.

The discussed RNN models are all same except the computational unit in the recurrent hidden layer. RNN are fundamentally complex networks that consist of many interactions between the computational units in the recurrent hidden layers to carry out a certain task from one layer to the other. Despite the fact that the effectiveness of various RNN has analyzed on traffic matrix prediction data sets under different experiments, the information about the internal procedure of the operation in the network is partly demonstrated. This can be explored by transforming the state of network to linearized dynamics and compute, analyze the structure of Eigen values and Eigen vectors from them across many time steps [29]. This analysis provides which Eigen vector actually carrying out application specific information and it will be orthogonal to the invariance of the network. So overall the context of Eigen vector associated with the linearized weight matrix enables to understand the overall provenance of the networks performance in each time steps.

References

[1] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale ip traffic matrices from link loads," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1. ACM, 2003, pp. 206–217.

[2] A. Soule, A. Nucci, R. Cruz, E. Leonardi, and N. Taft, "How to identify and estimate the largest traffic matrix elements in a dynamic environment," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, no. 1. ACM, 2004, pp. 73–84.

[3] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 161–174, 2002.

[4] P. Tune and M. Roughan, "Internet traffic matrices: A primer," *Recent Advances in Networking*, vol. 1, pp. 1–56, 2013.

[5] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot, "Traffic matrices: balancing measurements, inference and modeling," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1. ACM, 2005, pp. 362–373.

[6] M. Polverini, A. Iacovazzi, A. Cianfrani, A. Baiocchi, and M. Listanti, "Traffic matrix estimation enhanced by sdns nodes in real network topology," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*. IEEE, 2015, pp. 300–305.

[7] W. E. Leland, W. Willinger, M. S. Taqqu, and D. V. Wilson, "On the self-similar nature of ethernet traffic," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 202–213, 1995.

[8] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Internet traffic forecasting using neural networks," in *Neural Networks, 2006. IJCNN'06. International Joint Conference on*. IEEE, 2006, pp. 2635–2642.

[9] H. Feng and Y. Shu, "Study on network traffic prediction techniques," in *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, vol. 2. IEEE, 2005, pp. 1041–1044.

[10] J. Dai and J. Li, "Vbr mpeg video traffic dynamic prediction based on the modeling and forecast of time series," in *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*. IEEE, 2009, pp. 1752–1757.

[11] V. Dharmadhikari and J. Gavade, "An nn approach for mpeg video traffic prediction," in *Software Technology and Engineering (ICSTE), 2010 2nd International Conference on*, vol. 1. IEEE, 2010, pp. V1–57.

[12] A. Abdennour, "Evaluation of neural network architectures for mpeg-4 video traffic prediction," *IEEE transactions on broadcasting*, vol. 52, no. 2, pp. 184–192, 2006.

[13] M. Barabas, G. Boanea, A. B. Rus, V. Dobrota, and J. Domingo-Pascual, "Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition," in *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 95–102.

[14] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

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

[16] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[17] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:1504.00941*, 2015.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[19] Y. Wen and G. Zhu, "Prediction for non-gaussian self-similar traffic with neural network," in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 1. IEEE, 2006, pp. 4224–4228.

[20] ——, "Prediction for non-gaussian self-similar traffic with neural network," in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 1. IEEE, 2006, pp. 4224–4228.

[21] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[25] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, 2006.

[26] R. Moazzezi, "Change-based population coding," Ph.D. dissertation, UCL (University College London), 2011.

[27] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[28] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.