Deep Learning-based Traffic Prediction for Network Optimization

Sebastian Troia, Rodolfo Alvizu, Youduo Zhou, Guido Maier, Achille Pattavina
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan 20133, Italy
e-mail: sebastian.troia@polimi.it

ABSTRACT
In recent years, researchers realized that the analysis of traffic datasets can reveal valuable information for the management of mobile and metro-core networks. That is getting more and more true with the increase in the use of social media and Internet applications on mobile devices. In this work, we focus on deep learning methods to make prediction of traffic matrices that allow us to proactively optimize the resource allocations of optical backbone networks. Recurrent Neural Networks (RNNs) are designed for sequence prediction problems and they achieved great results in the past years in tasks like speech recognition, handwriting recognition and prediction of time series data. We investigated a particular type of RNN, the Gated Recurrent Units (GRU), able to achieve great accuracy (<7.4 of mean absolute error). Then, we used the predictions to dynamically and proactively allocate the resources of an optical network. Comparing numerical results of static vs dynamic allocation based on predictions, we can estimate a saving of 66.3% of the available capacity in the network, managing unexpected traffic peaks.
Keywords: Deep Learning, Machine Learning, Internet Traffic Prediction, Network Optimization.

1. INTRODUCTION

As telecommunications networks become increasingly important, understanding how to correctly predict network behaviour plays a vital role in the management and provisioning of mobile and fixed network services. As a consequence, traffic prediction has become very important for network providers [1]. Having an accurate traffic predictor tool is essential for most network management tasks, such as resource allocation, short-time traffic scheduling or re-routing, long-term capacity planning, network design and network anomaly detection.
A proactive prediction-based approach allows network providers to optimize network resource allocation, potentially improving the quality of service. Recently, some relatively reliable predictive methods in the field of temporal data prediction have been proposed. Zhuo *et al.* [2] have developed a prediction model based on the analysis of the autocorrelation coefficients of the time series in order to improve the accuracy of the prediction. After considering the autocorrelation features, they have implemented a Long Short-Term Memory Recurrent Neural Network (LSTM RNN) as prediction algorithm. Azzouni *et al.* [3] presented a framework for network traffic matrix prediction based on LSTM RNNs. They have validated the framework on real-world data from GEANT network showing a very low mean squared error. In this work, we show an effective use of deep learning as a tool to perform an intelligent network optimization. We have implemented a Gated Recurrent Unit Recurrent Neural Network (GRU RNN) on real-world data from Abilene[1] network. Our model proposes an Evaluation Automatic Module (EAM), which has the task of automating the learning process and generalizing the prediction model with the best possible performance. After that, we used the predictions as input for a network optimization tool called Net2Plan [4] to optimize the Routing and Spectrum assignment in the underlying optical network. The results of this work demonstrate the importance of traffic prediction for a network operator in the efficient management of its network. The paper is organized as follows: Section 2 describes the dataset we used for our experiments; Section 3 introduces the RNN and the GRU that represent the foundation of this work. Section 4 presents the proposed system to make predictions and network optimization, in which we explain the EAM. Section 5 shows the results of our experiments and finally Section 6 provides the conclusions.

2. DATASET AND PROCESSING

In this work, we propose an end-to-end deep learning architecture for short-term traffic matrix prediction. A deep learning architecture is trained and evaluated by using a dataset of traffic matrices of Abilene[1] network. Abilene is a backbone network created by the Internet2 Community[2] that enables the development of advanced Internet applications. Data traffic matrices are taken in 5 minute steps starting on March 1st 2004 at 00:00 and ending on September 10th 2004. The traffic matrices were processed to obtain a dataset describing each traffic matrix as a vector, where each element represents the traffic value of a specified couple of nodes. The result of this data manipulation provides a dataset with a number of rows equal to the number of starting matrices, that is, 48096; and a number of columns equal to the square of the number of nodes, or 144. Thanks to this number of samples available, it is possible to create a reliable and effective prediction algorithm. The dataset was divided into 3 sets using the following division: training 60%, validation 10% and testing 30%. The first, served to train the prediction

[1] http://sndlib.zib.de/home.action
[2] https://www.internet2.edu/

model in order to obtain the set of weights W of the Artificial Neural Network (ANN) that minimizes the error between the predicted value and the true value of the traffic. The validation set, served to generalize the model, that is to solve the problem of overfitting. The latter is a frequent problem in machine learning algorithms, and occurs when the selected algorithm learns to fit just the training data set, but starts performing badly with the incoming (new) data. Finally, the testing set is left to test the algorithm with data that the prediction model has never seen.

3. RECURRENT NEURAL NETWORKS

This section describes the Recurrent Neural Network (RNN) used for network traffic prediction and introduces a special type of RNN: the Gated Recurrent Units (GRU) networks. RNN is a learning method in the fields of deep learning that gained a lot of attention in recent years. It is different from the traditional Feedforward Neural Network (FNN) because it introduces a recurring structure for implementing a memory mechanism, which until now was absent in the FNNs. Through this structure, neurons keep track of past information and use it to influence the output at the current moment, making it suitable for predicting time series data. Unfortunately, the RNN is affected by a problem, called gradient explode or gradient vanish [5][6], that prevents complete learning of the time series. Due to this issue, we investigated the Gated Recurrent Units Recurrent Neural Networks (GRU RNNs) that are an evolution of the Long Short Term Memory (LSTM) RNNs proposed by Hochreiter in [7].

3.1 Gated Recurrent Units (GRU)

GRU was proposed by Cho et al. [8] to make each recurrent unit (or neuron) capable of adaptively capturing dependencies on different time scales. Similar to the LSTM unit, the GRU has gating units that modulate the flow of information inside the neuron, as shown in *Figure 1*, thus implementing a memory mechanism. The output of the unit (or activation) h_t^j, where t is time (or epoch) and j is the j^{th} unit (or neuron), is a linear interpolation between the previous activation h_{t-1}^j and the candidate activation $\hat{h}_t^j$:

$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j,$$

z_t^j is an update gate that decides how much the unit updates its activation, or content. The update gate is computed by:

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j$$

where W_z is the weighted matrix of the input and U_z is the weighted matrix of the previous time step. The candidate activation $\hat{h}_t^j$ is computed in a way similar to the traditional recurrent unit,

$$\hat{h}_t^j = tanh\big(Wx_t + U(r_t \circ h_{t-1})\big)^j,$$

where r_t^j is a set of reset gates and $\circ$ is an element-wise multiplication. When off (r_t^j close to 0), the reset gate effectively makes the unit act as if it is reading the first symbol of an input sequence, allowing it to forget the previously computed state. The reset gate r_t^j is computed similarly to the update gate:

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j$$

The most prominent feature shared between these units is the additive component of their update from t to $t + 1$, which is lacking in the traditional recurrent unit. The traditional recurrent unit always replaces the activation, or the content of a unit with a new value computed from the current input and the previous hidden state. This additive nature has one big advantage: it is easy for each unit to remember the existence of a specific feature in the input stream for a long series of steps. Any important feature, decided by the update gate of the GRU, will not be overwritten but be maintained as it is.

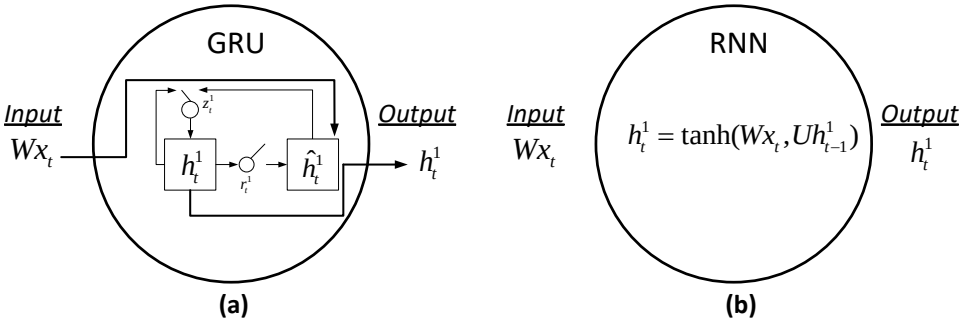

Figure 1. (a) GRU neuron example with j=1; (b) RNN neuron example with j=1.

4. SYSTEM MODEL

We input the processed dataset into the proposed prediction system. As shown in *Figure 2*, our method mainly incorporates two section: the first part is the classical ANN based on GRUs, and the second part is the proposed Evaluation Automatic Module (EAM). We use ANN to train the prediction model, and use the EAM to evaluate the model at each iteration of learning. The evaluation module has the task of automating the learning model without human intervention. In particular, at each iteration (or epoch[3]) of the ANN, EAM receives the result of the prediction $\hat{y}_T$ and the respective target y_T, and evaluates performance based on the prediction error of the training set E_T, and the one with the validation set E_V (see *Figure 2.c*). At each iteration, it stores the error in a database and compares it with the one of the previous iteration. When $E_V(e) < E_T(e)$, EAM let the ANN continue the training; otherwise, it stores the ANN at the epoch $e_1 = e$, and continue the training for other K iterations. In the end, after the K iterations, if it finds an epoch e_2 where $E_V(e_2) > E_T(e_2)$ and $E_V(e_2) < E_V(e_1)$, then stores the model and stops the training; otherwise, it considers the previous model at the epoch e_1 as the prediction model for the incoming traffic.

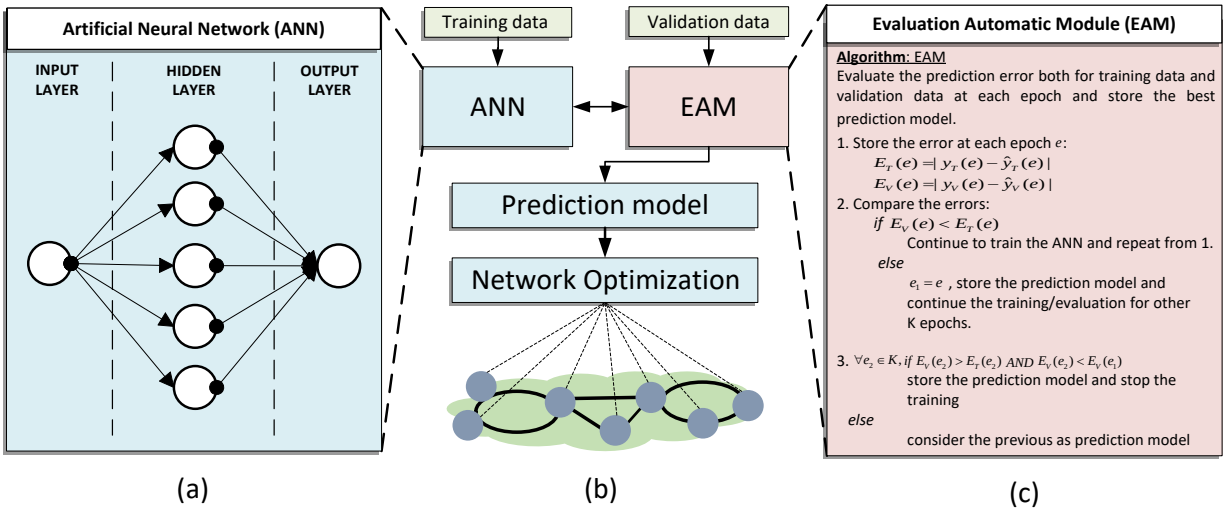

Figure 2. (a) General diagram of the ANN based on GRUs; (b) General diagram of the proposed system; (c) Pseudocode of the EAM algorithm proposed in this work.

The choice of parameter K is arbitrary. In this work we have chosen $K = 50$ because the errors E_T and E_V decrease as a negative exponential. For other applications or for other datasets, it could happen that errors decrease more slowly and in a very variable way; in such cases the parameter K needs to be increased. When the prediction model has been obtained, it will be used to predict the traffic matrix of the next hour, thus making a prediction hour by hour. The result of this prediction will be used as input for an optical network optimization algorithm based on an heuristic, solving the Routing, Spectrum, Modulation Assignment (RSMA) problem with regenerator placement, in a fixed grid optical WDM networks. The performance metric used is proportional to the amount of waisted capacity on each established lightpath. The network optimization was done with the open source planning tool Net2Plan [4].

5. EXPERIMENTS

In order to verify the accuracy of the prediction model, we used data from a backbone network called Abilene, presented in Section 2. After processing the initial dataset, we obtained the traffic matrices in vector form and we put them as input of the system proposed in Section 4, which has trained and generalized a prediction model in a fully automatic way. We trained the model to perform a one-hour prediction given the previous six hours. After that, the predicted traffic matrix is used to solve the RSMA (explained in the previous section) which provides the optimal routing and resource allocation for the next hour. To evaluate the performance of our system we computed the Mean Absolute Error (MAE) between testing-set matrices and predicted matrices over one day interval: $\boldsymbol{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\boldsymbol{Y_i} - \widehat{\boldsymbol{Y_i}}|$, where $\boldsymbol{Y_i}$ is a vector of $\boldsymbol{n}$ observed values and $\widehat{\boldsymbol{Y_i}}$ is the vector of the predictions. As the name suggests, the mean absolute error is an average of the absolute errors $|\boldsymbol{Y_i} - \widehat{\boldsymbol{Y_i}}|$ and it is a measure of prediction accuracy. MAE is a widely-used measure of prediction accuracy and a small value of it means that the predictor has high performance. The prediction model that consists in an ANN with one hidden layer composed

[3] From now on, iteration and epoch will be used as synonyms.

by 5 GRU nodes and one output layer with one node, presents a MAE of 7.4. This promising result is given by the memory structure implemented in each neuron of the ANN.

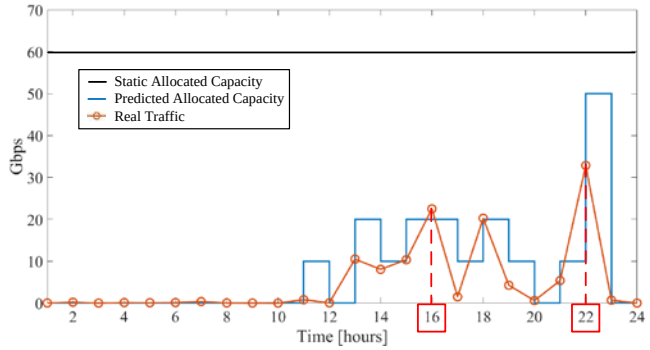

Figure 3. Capacity allocated to a link of Abilene net. Over all hours of a specific day in the testing set.

	Dynamic Allocation	**Static Allocation**
Capacity saving	66.3%	0%
Over-provisioning	68.9%	89.4%
Traffic Lost	3.0%	0%

Table 1. Total performances of the system over the testing set

We compared the resource allocation of our proposed method with a static planning of the network. The static planning was done by calculating the maximum bandwidth of each link in the traffic matrix, over the training set. In order to cope with unexpected traffic peaks, in both approaches we allocate 30% of extra capacity. *Figure 3* shows the allocation of capacity between two nodes of the Abilene network. At 16.00 we found a prediction error, that is, the system predicted a traffic value lower than the real one. This led the system to allocate less resources than the required to satisfy the bandwidth request; generating traffic loss. On the other hand, the system managed to predict a traffic peak at 22.00, allocating more capacity than needed; increasing the *over-provisioning* in the network. Over-provisioning is the extra capacity that has been allocated compared to the capacity sufficient to satisfy the request. The performance of our model over the whole testing set is summarized in *Table 1*. Using dynamic allocation we obtained a capacity saving of 66.3% compared to the static case. In addition, we achieved an over-provisioning saving of 20.5% when comparing with the static resource allocation. As expected, the unpredictability of the network traffic lead to situations in which the traffic prediction is lower than the real one. Even though we considered a 30% of extra capacity, our results reported 3% of *traffic lost* over the testing set.

6. CONCLUSIONS

This paper analyses the use of deep learning based traffic prediction to optimize proactively and dynamically an optical network. The proposed EAM shows how the automation of the training and generalization of a deep learning algorithm is effective for obtaining a reliable and high performance prediction model. Based on the numerical results, the system has obtained very promising performances both from the point of view of the prediction and from that of the allocation of resources. Furthermore, the model is part of one of the first attempts to optimize the operator's network with machine learning.

ACKNOWLEDGEMENTS

The work leading to these results has been supported by the European Community under grant agreement no. 761727 Metro-Haul project

REFERENCES

[1] R. Babiarz, *et al.*, “Internet traffic midterm forecasting: a pragmatic approach using statistical analysis tools”, 2006, Lecture Notes on Computer Science, 3976, 111–121
[2] Q. Zhuo, *et al.*, "Long short-term memory neural network for network traffic prediction," 2017 12th International Conf. on Intelligent Systems and Knowledge Engineering (ISKE), Nanjing, 2017, pp. 1-6.
[3] A. Azzouni, *et al.*, “NeuTM: A Neural Network-based Framework for Traffic Matrix Prediction in SDN”, arXiv preprint arXiv:1710.06799
[4] P. Pavon-Marino, *et al.*, “Net2plan: an open source network planning tool for bridging the gap between academia and industry”, IEEE Network, vol. 29, no 5, p. 90-96, October/November 2015.
[5] Y. Bengio, *et al.*, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 2002, 5(2):157-166.
[6] S. Hochreiter, *et al.*, “The vanishing gradient problem during learning recurrent neural nets and problem solutions”, World Scientific Publishing Co. Inc. 1998.
[7] S. Hochreiter, *et al.*, “J. Long short-term memory”, Neural Computation, 1997, 9(8):1735.
[8] K. Cho, *et al.*,“On the properties of neural machine translation: Encoder-decoder approaches”, arXiv preprint arXiv:1409.1259, 2014.