Network Traffic Prediction based on Diffusion Convolutional Recurrent Neural Networks

Davide Andreoletti[1, 2], Sebastian Troia[2], Francesco Musumeci[2], Silvia Giordano[1], Guido Maier[2], and Massimo Tornatore[2]

[1]Networking Laboratory, University of Applied Sciences of Southern Switzerland, Manno, Switzerland, Email: {name.surname}@supsi.ch
[2]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy, Email: {name.surname}@polimi.it

***Abstract*—By predicting the traffic load on network links, a network operator can effectively pre-dispose resource-allocation strategies to early address, e.g., an incoming congestion event. Traffic loads on different links of a telecom is know to be subject to strong correlation, and this correlation, if properly represented, can be exploited to refine the prediction of future congestion events. Machine Learning (ML) represents nowadays the state-of-the-art methodology for discovering complex relations among data. However, ML has been traditionally applied to data represented in the Euclidean space (e.g., to images) and it may not be straightforward to effectively employ it to model graph-stuctured data (e.g., as the events that take place in telecom networks). Recently, several ML algorithms specifically designed to learn models of graph-structured data have appeared in the literature. The main novelty of these techniques relies on their ability to learn a representation of each node of the graph considering both its properties (e.g., features) and the structure of the network (e.g., the topology). In this paper, we employ a recently-proposed graph-based ML algorithm, the Diffusion Convolutional Recurrent Neural Network (DCRNN), to forecast traffic load on the links of a real backbone network. We evaluate DRCNN's ability to forecast the volume of expected traffic and to predict events of congestion, and we compare this approach to other existing approaches (as LSTM, and Fully-Connected Neural Networks). Results show that DCRN outperforms the other methods both in terms of its forecasting ability (e.g., MAPE is reduced from 210% to 43%) and in terms of the prediction of congestion events, and represent promising starting point for the application of DRCNN to other network management problems.**

***Index Terms*—traffic forecasting, graph-based machine learning, network congestion**

I. Introduction

As telecom networks become more and more complex (see, e.g., the enormous set of adjustable parameters to be managed in modern systems), is also becoming increasingly important to limit human intervention and speed up network management procedures. Novel software solutions for network automation allow to automatically configure, provision, manage and test network devices and can be used to increase the infrastructure efficiency and reduce human error and operational expenditures.

In particular, network traffic prediction plays an important role in many areas of networking, such as network management, network design, short and long-term resource allocation, traffic (re)-routing and anomaly detection. Two categories of prediction methods, based on long and short term's periods, are typically considered. Long-term traffic prediction is used to estimate future capacity requirements, and therefore enables a more effective planning decisions. Short-term traffic prediction (i.e., predictions within minutes, even seconds) is usually linked to dynamic resource allocation, and can be used to improve Quality of Service (QoS) mechanisms as well as for congestion control and optimal resource management. Several different techniques including time series models, modern data mining techniques, soft computing approaches, and neural networks have been used for network traffic analysis and prediction [1].

Within a telecom network, traffic is exchanged between nodes and crosses network links. Such links have relations among each other, i.e., due to their adjacency, their behaviour is correlated. For example, it is more likely that congestion occurs in links adjacent to a congested link than elsewhere. Due to the large amount of data that is available today in telecom networks, algorithms coming from the area of Machine Learning (ML) have been investigated to enable network intelligence [2], thanks to the ability of ML to extract useful (and sometimes "hidden") information from data. However, despite the significant amount of research in this direction, the topological relation among the links has not been traditionally leveraged by these machine learning algorithms, and, to the best of our knowledge, no existing solution is specifically designed to process graph-structured data.

In this paper we employ a recently-proposed machine learning algorithm (originally developed to do road traffic forecasting [3]) to predict the traffic load on the links of a telecom network. This algorithm is referred to as Diffusion Convolution Recurrent Neural Network (DCRNN) and, differently from traditional machine learning approaches, it can capture important topological properties of the network, which are expected to significantly influence the patterns followed by the traffic when propagating through the network.

Our objective is to predict to next load on a link of a telecom network, given the sequence of the past observations of link loads. The problem is modeled as a regression where the objective is to minimize the error between the predicted and the actual next load on the links. In the literature, this specific problem has already been addressed by using ML methods. However, to the best of our knowledge, this is the first time that a ML algorithm able to capture the topological relations of the links of telecom network is employed to perform this task. Specifically, we train a DCRNN using real data gathered from a backbone network (i.e., Abilene) and compare this approach

with several baselines traffic-prediction algorithms (e.g., the LSTM network [4]). A comparison of both the effectiveness of the regression (e.g., measured in terms of mean absolute error) and the ability to detect congestions events (which are defined following a threshold-based criterion) is carried out. Results show a remarkable improvement of the DCRNN with respect to all the baseline methods on both the aspects, and encourage its application also for other network management tasks.

The rest of the paper is structured as follows. In Section II we review some works related to the use of machine learning as a tool for network traffic prediction, as well as the several machine learning algorithms specifically designed to work on graph-structured data. Section III briefly reviews the concepts needed to understand the proposed methodology, such as the recurrent neural networks and the diffusion convolutional operator. In Section IV, we present the problem statement and describe the employed methodology. Description of the simulation settings and presentation of results is given in Section V. Finally, Section VI concludes the paper.

II. Related Work

An accurate prediction of network traffic is of utmost importance for network operators, as it enables an efficient management of resources and load balancing. Given the importance of the topic, the related literature is abundant. We focus here on several related works evaluating ML-based methods for network traffic prediction.

The authors of [5] propose a framework for network Traffic Matrix (TM) prediction based on Recurrent Neural Networks equipped with the Long Short-Term Memory units, i.e., RNN LSTM. TM prediction is defined as the problem of estimating future network traffic matrix from the previous ones. Similar approaches can be found in [6], [7]. [6] proposes an end-to-end deep learning architecture consisting of a convolutional and a recurrent module that, combined, can extract both spatial and temporal information from the traffic flows. [7] proposes a model of neural network which can be used to combine LSTM with Deep Neural Networks (DNN). An autocorrelation coefficient is added to the model to improve the accuracy of predictions. The main novelty of [7] is to include autocorrelation of the time series in the input of the ML algorithm, which leads to superior performance with respect to existing methods. The combination of a special type of LSTM unit, i.e., the Gated Recurrent Units (GRU) and the Convolutional Neural Network (CNN) in the 2D domain (CNN-2D) has been proposed for the task of network traffic prediction in datacenters in [8]. The underlying idea of the work in [8] is to treat network matrices as images and use the CNN2D to find the correlations among traffic exchanged between different pairs of nodes. Note that, in literature, the prediction of traffic exchanged among network nodes is more common than the prediction of the load on network links. However, examples of application of ML to this specific task can be found, e.g., in [9] where Support Vector Machines are employed to perform the regression.

To our knowledge, none of the existing methods of traffic prediction explicitely considers the topological information of the network. Arguably, this is due to the fact that ML solutions specifically designed to process data that do not belong to the Euclidean domain [10], and in particular those with a graph-based structure, have appeared in the literature only recently [10]. At a high-level, these methods are based on filtering operations designed to be suitable for graphs. These filters are used within machine learning algorithms and their parameters are learned to make them able to capture hidden patterns of the relations among the nodes of the graph. For example, [11], [12] propose a generalization of the CNN that is suitable to process graphs to perform, for example, classification of the nodes. The authors of [3] propose the diffusion convolution operator and build a machine learning algorithms based on this. This algorithm is then used to perform traffic forecasting on road traffic in [3], [13]. Here, we use the same methodology to forecast the load on the links of a telecommunication network.

III. Background

In this Section, we briefly review background concepts to understand the DCRNN, as well as benchmarks algorithms that we compare with the proposed approach.

A. Convolutional Neural Networks

Convolution is widely employed in signal processing to perform filtering operations. The convolution between two signals x and w is defined as:

$$(x * w)(t) = \sum_{\tau=0}^{\mathrm{T}} x(t) \cdot w(t-\tau) \tag{1}$$

where w is generally referred to as *kernel* of *filter* and T is its support. In general, the kernel w is hand-crafted by expert designers in such a way that the convolution captures some desidered properties of the signal. A Convolutional Neural Network (CNN) is a machine learning module that is trained to learn the parameters of a number of filters (whose support, i.e., their length, is fixed).

CNN networks can be formed by stacking together multiple CNN layers. In general, these architectures are characterized by a Dropout layer on top of each CNN. Although CNNs are more commonly used in the 2D domain (e.g., to perform image recognition), it is not rare to see their employment also in the 1D domain, e.g., for time-series forecasting. The support of the kernel tunes the level of temporal dynamic that the filters can capture. Namely, filters with long support can extract longer temporal dynamic with respect to shorter ones.

B. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have been designed with the specific purpose to overcome the limitations of feedforward neural networks in modeling sequences. RNN networks are composed of units (i.e., neurons) capable of keeping track of past observations. This allows RNNs to model the input data based on both current and previously

seen observations. Among the proposed RNNs architectures, the Long Short-Term Memory (LSTM) proved particularly effective in modeling long-range temporal dependendies of input data.

More formally, given an input vector $\mathbf{x}(t)$ and the current observation (say $\mathbf{x}(t+1)$), the LSTM unit recursively performs the following operations:

$$\mathbf{i}(t) = \sigma(\mathbf{W}_i[\mathbf{x}(t), \mathbf{h}(t-1)] + \mathbf{b}_i) \tag{2}$$

$$\tilde{\mathbf{c}}(t) = tanh(\mathbf{W}_c[\mathbf{x}(t), \mathbf{h}(t-1)] + \mathbf{b}_c) \tag{3}$$

$$\mathbf{f}(t) = \sigma(\mathbf{W}_f[\mathbf{x}(t), \mathbf{h}(t-1)] + \mathbf{b}_f) \tag{4}$$

$$\mathbf{c}(t) = \mathbf{f}(t) \odot \mathbf{c}(t-1) + \mathbf{i}(t) \odot \tilde{\mathbf{c}}(t) \tag{5}$$

$$\mathbf{o}(t) = \sigma(\mathbf{W}_o[\mathbf{x}(t), \mathbf{h}(t-1)] + \mathbf{b}_o) \tag{6}$$

$$\mathbf{h}(t) = \mathbf{o}(t) \odot tanh(\mathbf{C}(t)) \tag{7}$$

where $\odot$ is the element-wise matrix multiplication. $\mathbf{W}_i, \mathbf{W}_c, \mathbf{W}_f, \mathbf{W}_o$ and $\mathbf{b}_i, \mathbf{b}_c, \mathbf{b}_f, \mathbf{b}_o$ are learnable kernels and biases, respectively, whereas $\mathbf{i}$, $\tilde{\mathbf{c}}$, $\mathbf{f}$, $\mathbf{c}$, $\mathbf{o}$ are referred to as *input*, *input modulation*, *forget*, *cell* and *output* gates and jointly perform operations to make the LSTM able select the information to remember and to forget from the input data. Finally, the *hidden state* $\mathbf{h}$ encodes what the LSTM unit retains about past observations and, along with $\mathbf{x}$, is successively used as input data.

Many variants of LSTM units have been proposed in the literature, and the above formulation only refers to its most common implementation. For example, the RNN Gated Recurrent Units (GRU) is a widely-used and simplified version of the LSTM, which is used for example in the recently-proposed Diffusion Convolutional Recurrent Neural Network (DCRNN).

C. Diffusion Convolutional Recurrent Neural Network

Machine Learning algorithms have been originally thought to learn models of data defined on Euclidean domains and their application to other types of data, such as graphs, is not straightforward [10]. Specifically, a graph is defined as the pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. If the graph is characterized by attributes (e.g., properties of nodes and edges), $\mathcal{G}$ can be alternatively described as $(\mathbf{X} \in \mathbb{R}^{NXP}, \mathbf{W} \in \mathbb{R}^{NXN})$, where N is the number of nodes and P the number of their attributes (i.e., features). $\mathbf{X}$ is the *feature matrix* and $\mathbf{W}$ is a weighted matrix that encodes the relations among the nodes, e.g., the adjacency matrix of the graph.

Traditional ML algorithms (e.g., LSTM or CNN) can easily process $\mathbf{X}$, but fall short in including the information encoded in $\mathbf{W}$. Recent approaches proposed in the literature aim to enrich the feature matrix with this relational information. In this Section, we briefly review one of the most prominent solutions of this kind, which is based on the idea that the relation between two nodes can be represented as a *diffusion* process. Specifically, the probability that a random walk of K steps that starts at the first node and ends at the second can be computed knowing the state transition matrix $\mathbf{D_0}^{-1} \cdot \mathbf{W}$ (with $\mathbf{D_0}$ being the out-degree diagonal matrix of the graph).

Intuitively, the diffusion process gives important clues on the influence that each node excercises on all the others. This contextual knowledge may be used to improve the representation of the nodes within the feature space (i.e., $\mathbf{X}$) through the application of filtering performed using appropriate convolutional operations.

The K-steps diffusion convolution between a graph signal $X \in \mathbb{R}^{NXP}$ and a filter $f_{\boldsymbol{\theta}}$ is referred to as $*_{\mathcal{G}}$ and defined as:

$$\mathbf{X} *_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1} (\mathbf{D_0}^{-1} \mathbf{W})^k + \theta_{k,2} (\mathbf{D_0}^{-1} \mathbf{W}^\intercal)^k \right) \cdot \mathbf{X} \tag{8}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{KX2}$ are the parameters of the filter, $\mathbf{D_0}^{-1} \cdot \mathbf{W}$ is the state transition matrix of the diffusion process and $\mathbf{D_0}^{-1} \cdot \mathbf{W}^\intercal$ is its transpose.

The diffusion convolutional operator can be used as building block of a Diffusion Convolutional Layer of Neural Network and $\boldsymbol{\theta}$ learnt using common training approaches (e.g., back-propagation). Specifically, this layer can be trained to map the feature matrix $\mathbf{X} \in \mathbb{R}^{NXP}$ to an output $\mathbf{H} \in \mathbb{R}^{NXQ}$ as follows:

$$\mathbf{H}_{:,q} = \sigma \left(\sum_{p=1}^{P} \mathbf{X}_{:,p} *_{\mathcal{G}} f_{\boldsymbol{\Theta}_{q,p,:,:}} \right) \quad , \forall q \in \{1, ..., Q\} \tag{9}$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{QXPXKX2}$ is the tensor of the trainable parameters. By replacing the matrix multiplications described in Section III-B with the diffusion convolutional operation, the RNN unit becomes the Diffusion Convolutional Gated Recurrent Unit (DCGRU) [3]. For the sake of precision, the authors of [3] present a modified version of the RNN GRU mentioned in Section III-B and formally described by the following equations:

$$\mathbf{r}(t) = \sigma(\boldsymbol{\Theta}_r *_{\mathcal{G}} [\mathbf{X}(t), \mathbf{H}(t-1)] + \mathbf{b}_r) \tag{10}$$

$$\mathbf{C}(t) = \tanh(\boldsymbol{\Theta}_C *_{\mathcal{G}} [\mathbf{X}(t), (\mathbf{r}(t) \odot \mathbf{H}(t-1))] + \mathbf{b}_c) \tag{11}$$

$$\mathbf{u}(t) = \sigma(\boldsymbol{\Theta}_u *_{\mathcal{G}} [\mathbf{X}(t), \mathbf{H}(t-1)] + \mathbf{b}_u) \tag{12}$$

$$\mathbf{H}(t) = \mathbf{u}(t) \odot \mathbf{H}(t-1) + (1 - \mathbf{u}(t)) \odot \mathbf{C}(t) \tag{13}$$

where $\odot$ is the element-wise tensor multiplication. $\mathbf{r}$, $\mathbf{u}$ and $\mathbf{C}$ are referred to as *reset*, *update* and *cell* gates respectively

and perform similar operations of the gates described in Section III-B. $\mathbf{\Theta}_r, \mathbf{\Theta}_u, \mathbf{\Theta}_C$ are the parameters of the kernels learnt during the training (along with their relative biases $\mathbf{b}_r$, $\mathbf{b}_u$, $\mathbf{b}_C$). $\mathbf{X}(t)$ (resp., $\mathbf{H}(t)$) is the input (resp., the output) of the model at time t.

IV. The proposed forecasting approach

In this work, we employ a deep learning approach to perform the network traffic forecasting task. Specifically, our objective is to predict the load on network links given historical records of traffic loads.

A. Problem Statement

We consider a telecom backbone network composed of a set of nodes and a set of links connecting them. The traffic exchanged among the nodes is assumed to be routed according to the shortest path. The resulting traffic load measured on the network at time *t* can be represented as a matrix $\mathbf{X}(t) \in \mathbb{R}_{\geq 0}^{MX1}$, where M is the number of links of the network.

Given the sequence of traffic loads measured during the previous T time slots, it is possible to forecast the load at time $t+1$, $\forall$ links $l \in \{1, ..., M\}$, i.e., $\mathbf{X}^{(t+1)}$, using well-known machine learning techniques (e.g., a LSTM network). However, whilst the topological properties of the network play a significant role in the diffusion of the traffic (e.g., because they constrain the traffic to flow only on existing paths), they are not easy to consider using the common approaches.

Instead, we employ an existing deep-learning architecture described in Section IV-B and based on the DCGRU described in Section III-C that is specifically designed to take advantage from the topological properties of graphs. To exploit this additional information, we represent the traffic crossing the network as a directed graph $\mathcal{G}$ that can be described by the matrix $\mathbf{X}(t) \in \mathbb{R}_{\geq 0}^{MX1}$ (which encodes the attributes of the M nodes, i.e., the load for each link of the telecommunication network) and by its adjacency matrix $\mathbf{W}$, where $w_{ij} = 1$ iff l_i and l_j are connected, and 0 otherwise, which encodes the relation between the nodes. The forecasting problem is formulated as follows:

$$\mathbf{X}^{(t+1)} = \mathcal{F}\big(\mathbf{W}, \mathbf{X}^{(t-T)}, ..., \mathbf{X}(t)\big) \tag{14}$$

where $\mathcal{F}$ is the estimator that we learn by employing the architecture described in the following.

B. DCRNN for Network Traffic Prediction

The deep-learning architecture proposed in [3] belongs to the family of *Sequence-to-Sequence* deep-learning architectures [14], which are characterized by an *encoder* and a *decoder*. The former learns a map between the input (which can be a sequence of unknown length) and a fixed-sized encoding vector. The latter learns how to map the encoding vector to the output sequence. Encoder and decoder perform symmetrical operations and are composed by the same (arbitrary) number of layers. In the architecture described in [3] and employed in our link load forecasting task, each layer is composed of $\mathcal{U}$ DCGRU units described in Section III-C.

V. Experiments

A. Experiment Setup

The objective of this work is to evaluate the ability of different deep learning architectures to forecast the traffic load on the links of a backbone network. In the following Sections, we provide details about the considered network and about the baseline methods that we use for comparison.

1) Dataset Preprocessing: We consider the backbone Abilene network, of which several information are public[1]. For example, its topology (characterized by 12 nodes and 30 unidirectional links connecting them) and some statistics of a trace of real traffic crossing it is available. Specifically, we know the volume of traffic, aggregated over slots of 5 minutes, that is exchanged between each pair of nodes starting from March 1st 2004 to September 10th 2004.

Assuming that traffic is routed considering the shortest path between two nodes, we can compute the traffic load on the links at each time slot. From this data, we have derived another dataset that gives information about the traffic load on each network link aggregated over slots of 1 hour, which results in 4000 vectors with 30 components. These data are arranged in cronological order and grouped together in sequences of 10 vectors, which are used as input of the ML algorithm. The output (i.e., the next value of the sequence to predict) is a single vector obtained by applying a shift to the corresponding sequence. Then, starting from the topology of the Abilene network, we have obtained a $30X30$ adjacency matrix representing the graph whose nodes are univocally associated with the network links and the edges encode the relation among them (i.e., an edge exists iff the corresponding links are connected in Abilene).

2) Deep Learning Architectures and Training Methodology: We consider a DCRNN architecture composed of two layers with 4 DCGRU units each. The first layer acts as encoder and the second as the decoder. We compare the DCRNN with the following baseline: a LSTM-based network, a CNN-based network, a CNN-LSTM-based network and a Fully-Connected Neural Network. The analysis of the hyper-parameters, which we omit in this paper, led to select architectures with the following characteristics:

- The LSTM-based network is composed of 5 recurrent layers with 20 LSTM units each
- The CNN-based network is composed of 1 layer that implements the convolution using 32 kernels of size 2
- The CNN-LSTM-based network is composed of 1 recurrent layer of 20 LSTM units stacked on top of a CNN layer (with 16 kernels of size 2)
- The Fully-Connected Neural Network is composed of 3 layers of 30, 20 and 10 units that apply a sigmoid operation to their input

The sequences described in the previous Section are taken in cronological order and divided such that 70% is used for training, 20% for validation, and the remaining 10% for

[1] http://sndlib.zib.de/home.action

TABLE I
COMPARISON OF THE DEEP LEARNING ARCHITECTURES CONSIDERING THEIR ABILITY TO PERFORM THE FORECAST OF THE NEXT TRAFFIC LOAD

	MAPE	MAE (Mbit/s)	RMSE (Mbit/s)	Convergence Epoch	Convergence Time (sec)
DCRNN	**43.2%**	**92.5**	**497.1**	225	525.1
LSTM	210.34%	142.43	525.21	**87**	19.83
CNN	234.75%	121.32	506.55	252	9.82
CNN-LSTM	248.16%	127.18	512.91	240	5.76
Fully-Connected	220.75%	138.24	522.65	201	**3.14**

testing. The training of the ML architectures is performed to minimize the Mean Absolute Error (MAE) between predictions and ground-truth and stops when no improvement on the validation set is noticed for at least 50 training epochs. The traning is performed using the Adam optimizer [15] with initial learning rate set to 0.01. In the following Section, we describe the results derived by averaging the results obtained in 100 simulations.

B. Experiment Results

The first set of experiments evaluates the employed method considering the *Mean Absolute Percentage Error* (MAPE), the *Mean Absolute Error* (MAE) and the *Root Mean Squared Error* (RMSE), as well as metrics related to the speed of convergence, i.e., number of epochs and time at which training is interrupted due to an early stopping event.

The results of the evaluation are summarized in Table I, where it is possible to notice how the DCRNN method significantly outperforms the baselines in MAPE, MAE and RMSE. In particular, the MAPE drops from $\sim 210\%$ obtained with the LSTM-based architecture to $\sim 43\%$ by using the DCRNN. We notice also an improvement with respect to the best MAE and RMSE (both obtained with the CNN-based architecture) which decrease from ~ 121 to ~ 92 Mbit/s and from ~ 506 to ~ 497 Mbit/s, respectively.

The improvement of the MAE of ~ 30 Mbit/s with respect to the best baseline is significant considering an average traffic on links of ~ 301 Mbit/s and that 50% of the measured loads are below 180 Mbit/s (see Table II, where we show several values of the percentile of the load on links). The improvement of the RMSE is the least impressive. This result can be explained saying that the DCRNN performs in general a better prediction of the next link loads (as indicated by the remarkable decrease of the MAPE), but it hardly predicts sudden high peaks (i.e., burst events). We do not consider this a limitation of the model, since the prediction of this type of event is essentialy not possible. As for the convergence speed, the time needed to train the DCRNN (i.e., $\sim 512sec$) is one order of magnitude higher than the LSTM-based architecture, which presents the most time-consuming training process among the baselines (i.e., $\sim 19sec$). We underline that the forecasting process introduces a negligible delay for all the considered models.

A straightforward application of a reliable estimator of traffic load is the early detection of congestion events. In the second set of experiements, we assess the ability of our approach to perform this task and we compare it with the baselines. We assume that a congestion occurs on a link if the traffic load is above a threshold that is directly proportional (of a factor α) to the average amount of traffic observed on that link. In this way, we perform a fair comparison that takes into consideration the different patterns of link load that Abilene presents. The evaluation is done considering the following metrics: percentage of *false positives*, *false negatives*, *true positives* and *true negatives*, from which we derive *precision*, *accuracy*, *recall* and *F-score*.

TABLE II
PERCENTILE OF THE LOAD MEASURED ON LINKS FOR THE TEST SET

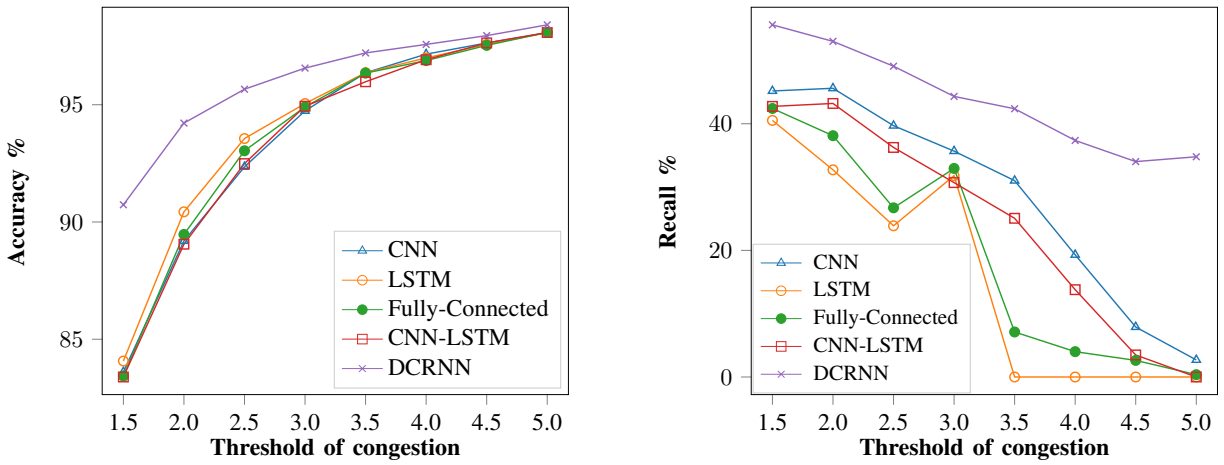
	25%	**50%**	**75%**	**100%**
Traffic On Links (Mbits/s)	59.67	180.33	389.41	5929.52

In Table III we show the results obtained with $\alpha = 3$ (i.e., a link is congested when the volume of traffic is above 3 times the average load). The DCRNN outperforms the baselines for all the considered metrics. In particular, the precision (i.e., the percentage of congestion predictions that are actually congestion events) is increased of up to 25% with respect the the best baseline (i.e., the LSTM-based architecture).

As far as the congestion prediction task is concerned, the recall is the percentage of congestion events that are correctly predicted, whereas the accuracy is the percentage of correct predictions (being they referred to congestion or normal loads). Hence, they both give essential indications to a network operator that takes decision based on the likelihood that congestion will (or will not) occur.

We depict the accuracy and the recall in Fig. 1(a) and Fig. 1(b), respectively, as a function of the threshold congestion expressed by $\alpha \in [1.5, ..., 5]$. We notice that the DCRNN always outperforms the baselines also in this task. Increasing α maens to limit the congestion events only to traffic volumes that are significantly higher than the average link load. This has two opposite effects on the accuracy and on the recall. In fact, the accuracy increases as a consequence of the increased number of non-congestion events, which positively affects the number of correct classifications. Conversely, the recall shows a general decrease with increasing α. This can be explained considering that the models hardly predict very high and sudden peaks, as already discussed in relation to the RMSE.

$\alpha = 5$ represents the hardest conditions to detect a congestion event. In this scenario, in fact, the DCRNN reaches a recall of $\sim 34\%$, which means that 56% of the actual congestions are not detected. Notice, however, that this result is still 32% higher than the recall obtained by the best baseline



(a) *Accuracy of the effectiveness to detect a congestion event*

(b) *Recall of the effectiveness to detect a congestion event*

Fig. 1. Comparison of all the methods considering the ability to detect a congestion event in terms of Accuracy and Recall

TABLE III
COMPARISON OF THE DEEP LEARNING ARCHITECTURES CONSIDERING THEIR ABILITY TO DETECT A CONGESTION EVENT WHEN THRESHOLD FACTOR $\alpha = 3$

	TP	TN	FP	FN	Accuracy	Precision	Recall	F-score
DCRNN	**1,97**	**94,70**	**0,93**	**2,40**	**96,67**	**67,93**	**45,01**	**54.14**
LSTM	1,14	93,64	1,92	3,03	95,05	42,37	31,80	36,33
CNN	1,58	93,15	2,40	2,85	94,74	41.86	35,67	37,85
CNN-LSTM	1,36	93,57	1,98	3,08	94,93	40,71	30.70	34,93
Fully-Connected	1,15	93,44	2.11	0.029	94,91	41,31	32,94	36,45

(i.e., the CNN-based architecture).

VI. CONCLUSIONS

In this work, we employ an existing graph-based machine learning algorithm (i.e., the DCRNN) to forecast the next traffic load on the links of the backbone telecom network Abilene. The main novelty of this appraoch is the ability to learn a representation of the telecom network that considers both the features (i.e., the load on the links) and the topological relations among them (i.e., if the links are connected or not). The DCRNN is compared to the baselines (e.g., LSTM and CNN) considering the effectiveness of the forecasting and the ability to detect congestion events. For example, a reduction of the MAPE from 210% to 43% is observed. These promising results suggest that the forecasting of events within a telecom network may significantly benefit from using ML approaches explicitely-designed to capture, along with the properties of the events themselves, also the structure of the network.

VII. ACKNOWLEDGEMENTS

The work leading to these results has been supported by the European Community under grant agreement no. 761727 Metro-Haul project and by the EU FP7 ERANET program under grant CHIST-ERA-2016 UPRISE-IOT.

REFERENCES

[1] M. Joshi and T. H. Hadi, "A review of network traffic analysis and prediction techniques," *CoRR*, vol. abs/1507.05722, 2015. [Online]. Available: http://arxiv.org/abs/1507.05722

[2] R. Alvizu, S. Troia, G. Maier, and A. Pattavina, "Matheuristic with machine-learning-based prediction for software-defined mobile metro-core networks," *Journal of Optical Communications and Networking*, vol. 9, no. 9, pp. D19–D30, 2017.

[3] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," 2018.

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

[5] A. Azzouni and et al, "Neutm: A neural network-based framework for traffic matrix prediction in sdn," *CoRR*, vol. abs/1710.06799, 2017.

[6] Y. Liu and et al, "Short-term traffic flow prediction with conv-lstm," in *Wireless Communications and Signal Processing (WCSP), 2017*. IEEE, 2017, pp. 1–6.

[7] Q. Zhuo and et al, "Long short-term memory neural network for network traffic prediction," in *ISKE*. IEEE, 2017, pp. 1–6.

[8] X. Cao and et al, "Interactive temporal recurrent convolution network for traffic prediction in data centers," *IEEE Access*, vol. 6, pp. 5276–5289, 2018.

[9] P. Bermolen and D. Rossi, "Support vector regression for link load prediction," *Computer Networks*, vol. 53, no. 2, pp. 191–201, 2009.

[10] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[12] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

[13] X. Wang, C. Chen, Y. Min, J. He, B. Yang, and Y. Zhang, "Efficient metropolitan traffic prediction based on graph recurrent neural network," *arXiv preprint arXiv:1811.00740*, 2018.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[15] D. P. Kingma and et al, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.