Network Traffic Prediction based on Diffusion Convolutional Recurrent Neural Networks

DAVIDE ANDREOLETTI1,2, SEBASTIAN TROIA2, FRANCESCO MUSUMECI2, SILVIA GIORDANO1, GUIDO MAIER2, AND MASSIMO TORNATORE2

(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
Outline

• Introduction
• Objectives
• Problem Formulation
• Proposed Approach
• Network Dataset
• Experiments
• Conclusions
Introduction (1)

• Traffic matrix datasets can reveal valuable information for the management of mobile and metro-core networks

• Predicting network behaviour plays a vital role in the management and provisioning of mobile and fixed network services

• Traffic prediction represents an important service for network providers [1]
  • Resource allocation
  • Short-term traffic scheduling
  • Long-term capacity planning
  • Network design and network anomaly detection

High interest in Machine Learning, applications are growing rapidly [2]

Many works focus on network traffic prediction exploiting artificial and recurrent neural networks such as:

- [3] proposes a framework for network Traffic Matrix (TM) prediction based on Recurrent Neural Networks (RNN) equipped with the Long Short-Term Memory (LSTM) units
- [4] proposes a convolutional and a recurrent module to extract both spatial and temporal information from the traffic flows
- [5] treats network matrices as images and use the Convolutional Neural Networks (CNN) to find the correlations among traffic exchanged between different pairs of nodes

None of the existing methods explicitly considers the topological information of the network as feature to perform traffic prediction

Objectives

In this work, we show:

• Use of deep learning as a tool to perform intelligent traffic engineering
  • Network traffic prediction based on network data and topology information
  • Detect congestion events

• Validation with real backbone network traffic traces
  • We exploit open-source datasets to validate the proposed approach

• Comparisons with state-of-the-art deep learning methods
  • LSTM, CNN, Fully-Connected neural network
Problem Formulation

Given:
- A telecom network with \( N \) nodes and \( M \) unidirectional links
- \( T \) traffic matrices (NXN)
  - The \( t^{th} \) traffic matrix represents the volume of aggregated traffic exchanged between each pair of network nodes during the \( t^{th} \) time slot
  - \( t \) ranges from 1 to \( T \) (i.e., number of considered time slots)
- A fixed routing policy: shortest path

Goals:
- **Forecast** the volume of traffic on all the network links at time slot \( t+1 \)
- Perform a binary **classification**: congested/not congested link
Proposed Approach (1)

Data Processing

Sequence of Traffic Matrixes $X_t$

Sequence of Link Load Vectors $Y_t$

Routing strategy: shortest path

ML-based Forecasting

Predicted values $\hat{Y}_{T+1}$

Classification

Evaluated values $\hat{C}_{T+1}$

Evaluation 1 (e.g., MAE) $Y_{T+1}$

Evaluation 2 (e.g., accuracy) $C_{T+1}$

Routing strategy: shortest path

Compute the load on each link

Forecast the next load on links

Classify as congestion if traffic is above a threshold
Proposed Approach (2)

- **Regression Problem:** \( \hat{Y}_{T+1} = \mathcal{F}(Y_1, Y_2, \ldots, Y_T) \)
  - Traditional ML can learn a function \( \mathcal{F} \) to map historical values to the future ones
  - ML requires datapoints to be defined in Euclidean Spaces: \( Y_t \in \mathbb{R}^{MX1} \) (e.g., audio, video, financial data)

- **Is the Euclidean representation suitable for network traffic?**
  - Traffic propagation is highly-influenced by the topology of the network
  - Topological information are simply discarded by traditional ML

- **Our proposal:**
  - Represent network data as a graph to exploit spatial information
  - Employ a ML-based predictor specifically-designed to work on graph-like data
Proposed Approach (3)

- Starting from a graph $G = (V, E, W)$, with $V$ set of nodes, $E$ set of edges and $W$ the adjacency matrix.
- We represent $G$ by its attributes that are $Y_t$ and $W$.
- Nodes’ feature vector $Y_t \in \mathbb{R}^{M \times 1}$ encodes the volume of traffic on each link at time $t$.
  - $M$ is the number of links.
- Topology’s feature matrix: $W \in \mathbb{R}^{N \times N}$ encodes the relations among the nodes (e.g., adjacency matrix of the graph).
  - $w_{ij}$ entry is 1 if i-th and j-th link are connected, 0 otherwise.
Proposed Approach (4)

- We deploy a **Diffusion Convolution Recurrent Neural Network (DCRNN)** [6] to perform network traffic prediction.
- The DCRNN is composed by recurrent layers equipped with the **DCGRU** units allowing to exploit both spatial and temporal dependency of traffic.
- The DCGRU unit, based on the GRU, takes into account the topology information through the diffusion convolutional operation.

Proposed Approach (5)

- The propagation of traffic within the telecom network can be modeled as a diffusion process over a graph $G$.
- The diffusion process is characterized by a random walk on $G$ with:
  - Restart probability $a=[0, 1]$.
  - State transition matrix $D_0^{-1} W$.
    - $D_0$ is the out-degree diagonal matrix of $G$.
    - $W$ adjacency matrix of $G$.

Mathematically, this process can be expressed as a $K$-steps convolution between a graph signal $Y \in \mathbb{R}^{M \times 1}$ and a filter $f_\theta$ [6]:

$$ Y \ast f_\theta = \sum_{k=0}^{K-1} \left( \theta_{k,1} (D_0^{-1} W)^k + \theta_{k,2} (D_0^{-1} W^T)^k \right) Y $$

- $\theta \in \mathbb{R}^{K \times 2}$, parameters of the filter.

IDEA: Use the Diffusion Convolutional Operator as building blocks inside the Gated Recurrent Units (GRU) networks to learn parameters $\theta$ by means of backpropagation.

Network Dataset

- **Abilene Network**: a high performance backbone network created by the internet2 community [7]
- **Network characteristics:**
  - 12 nodes
  - 15 link
- **Dataset** is composed by Traffic Matrices:
  - 5 minutes of granularity
  - 6 month of time horizon
  - 48096 traffic matrices

[7] internet2 community. URL: [https://www.internet2.edu](https://www.internet2.edu)
Experiments – Data processing

- The traffic matrices were processed to obtain a dataset describing each traffic matrix as a vector of link loads.
- Assuming the shortest path routing policy, we obtained a dataset of aggregated traffic on links.
- Data processing steps:
  - Cleaning of raw data (fill with zeros missing traffic data in the corresponding time slot)
  - Aggregation of 5-minutes in 1-hour data
  - Setup of input sequences for training
  - Division in training set (70%), validation set (20%), test set (10%)
Experiments – Setup

• We consider a DCRNN architecture composed by:
  • 2 hidden layers with 4 DCGRU units each
  • The first layer acts as encoder (for validation) and the second as the decoder (for testing)

• Baseline deep learning methods:
  • The LSTM-based network: 5 recurrent layers with 20 LSTM units each
  • The CNN-based network: 1 layer that implements the convolution using 32 kernels of size 2
  • The CNN-LSTM-based network: 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: 3 layers of 30, 20 and 10 units that apply a sigmoid operation to their input

• Training is performed minimizing the Mean Absolute Error (MAE)
  \[ MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \]
Experiments – Results (1)

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

• DCRNN significantly outperforms the baselines in MAPE, MAE and RMSE
• MAPE drops from almost 210% obtained with the LSTM-based architecture to 43% by using the DCRNN
• 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
• Time needed to train the DCRNN (i.e., 512sec) is one order of magnitude higher than the LSTM-based architecture
Experiments – Results (2)

- **True Positive** = 1,97
- **True Negative** = 94,74
- **False Positive** = 0,93
- **False Negative** = 2,40
- **Accuracy** = 96,67
- **Recall** = 45,01

**Accuracy**: is a ratio of correctly predicted congestions to the total events

**Recall**: is the ratio of correctly predicted congestions to the all actual congestion events
Conclusions

• Use of deep learning as a tool to perform network traffic prediction
  • Detect congestions events with 96.67% of accuracy
• Implementation of a Diffusion Convolution Recurrent Neural Network (DCRNN)
  • Reduction of the baseline MAPE from 210% to 43%
• Validation with real backbone network traffic traces
  • We exploit Abilene open-source datasets to validate the proposed approach
• Comparisons with state-of-the-art deep learning methods
  • LSTM, CNN, CNN-LSTM, Fully connected
Thank you
Experiments – Results (3)

\[ \alpha = 3 \]

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>TN</th>
<th>FP</th>
<th>FN</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCRNN</td>
<td>1.97</td>
<td>94.70</td>
<td>0.93</td>
<td>2.40</td>
<td>96.67</td>
<td>67.93</td>
<td>45.01</td>
<td>54.14</td>
</tr>
<tr>
<td>LSTM</td>
<td>1.14</td>
<td>93.64</td>
<td>1.92</td>
<td>3.03</td>
<td>95.05</td>
<td>42.37</td>
<td>31.80</td>
<td>36.33</td>
</tr>
<tr>
<td>CNN</td>
<td>1.58</td>
<td>93.15</td>
<td>2.40</td>
<td>2.85</td>
<td>94.74</td>
<td>41.86</td>
<td>35.67</td>
<td>37.85</td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>1.36</td>
<td>93.57</td>
<td>1.98</td>
<td>3.08</td>
<td>94.93</td>
<td>40.71</td>
<td>30.70</td>
<td>34.93</td>
</tr>
<tr>
<td>Fully-Connected</td>
<td>1.15</td>
<td>93.44</td>
<td>2.11</td>
<td>0.029</td>
<td>94.91</td>
<td>41.31</td>
<td>32.94</td>
<td>36.45</td>
</tr>
</tbody>
</table>

- DCRNN outperforms the baselines for all the considered metrics
- The precision is increased of up to 25% with respect the best baseline (i.e., the LSTM-based architecture)
Introduction: towards a data-driven and flexible network management

- Internet traffic is steadily increasing\(^2\)
  - Network operators need to address this issue while limiting Capex and Opex

- Two main approaches:
  - Architectural solutions, e.g., Software-Defined Networking (SDN) to increase flexibility of network management
  - ML-based analytic solutions, e.g., reliable traffic estimators to early address network events (e.g., congestion)

---

Deep Learning for Traffic Prediction

• We need to learn patterns from a sequence to perform the forecast $\hat{Y}_{T+1} = \mathcal{F}(Y_1, Y_2, ..., Y_T)$: let us use Recurrent Neural Networks (RNN)

• RNNs equipped with a Gated Recurrent Unit (GRU) can capture dependencies on different time scales

• GRUs perform specific matrix multiplications

• By replacing the matrix multiplications with the diffusion convolutional operation, the RNN unit becomes the Diffusion Convolutional Gated Recurrent Unit (DCGRU)

$$F \otimes f_\theta = \sum_{k=0}^{k-1} \left( \theta_{k,1} (D_0^{-1} W)^k + \theta_{k,2} (D_0^{-1} W^T)^k \right) F$$

Introduction (2)

- High interest in Machine Learning, applications are growing rapidly\(^2\)

Proposed Approach (3)

- We model the spatial dependency by relating traffic flow to a diffusion process [6]

- Given a graph $G = (V, E)$ the diffusion process is characterized by a random walk on $G$ with:
  - Restart probability $a = [0, 1]$
  - State transition matrix $D_0^{-1}W$
    - $D_0$ is the out-degree diagonal matrix of $G$
    - $W$ adjacency matrix of $G$
Proposed Approach (3)

• Representing data as a graph $G = (V, E)$
  • Nodes’ Feature Matrix: $F \in \mathbb{R}^{N \times P}$
    • i-th row of $F$ represents the feature vector of node $v_i \in V$, $\forall i$ ($P$ number of features)
  • Topology’s Feature Matrix: $W \in \mathbb{R}^{N \times N}$
    • $W$ is a weighted matrix encoding the relations among the nodes (e.g., adjacency matrix of the graph)

• We give a new representation of the graph $G$
  • Nodes’ feature vector $Y_t \in \mathbb{R}^{M \times 1}$
    • $v_i \in V$ represents the i-th network link
    • $Y_t$ encodes the volume of traffic on each link at time $t$
  • Topology’s Feature Matrix $W$, whose ij-th entry is 1 if i-th and j-th link are connected, 0 otherwise
Proposed Approach (3)

- Given a graph $G = (V, E)$, with $V$ set of nodes and $E$ set of edges
- We represent $G$ in a different way, considering: $V$ the number of links $M$ and $E$ the adjacency matrix $W$
- Nodes’ feature vector $Y_t, \in \mathbb{R}^{M \times 1}$, encodes the volume of traffic on each link at time $t$
- Topology’s feature matrix: $W \in \mathbb{R}^{N \times N}$
  - $W$ is a weighted matrix encoding the relations among the nodes (e.g., adjacency matrix of the graph)
  - $w_{ij}$ entry is 1 if i-th and j-th link are connected, 0 otherwise

\[ Y_{t-1}, Y_{t-2}, \ldots, Y_t \quad \xrightarrow{W} \quad \text{ML-based Forecasting} \quad \xrightarrow{} \hat{Y}_{t+1} \]