POLITECNICO DI MILANO

BONSAI RESEARCH GROUP

SUPSI

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

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

[1] R. Babiarz, *et al.,* "Internet traffic midterm forecasting: a pragmatic approach using statistical analysis tools", 2006, Lecture Notes on Computer Science

Introduction (2)

- 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

[2] https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/
[3] A. Azzouni and et al, “Neutm: A neural network-based framework for traffic matrix prediction in sdn,” CoRR, vol. abs/1710.06799, 2017
[4] 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.
[5] X. Cao and et al, “Interactive temporal recurrent convolution network for traffic prediction in data centers,” IEEE Access, vol. 6, pp. 5276– 5289, 2018.

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

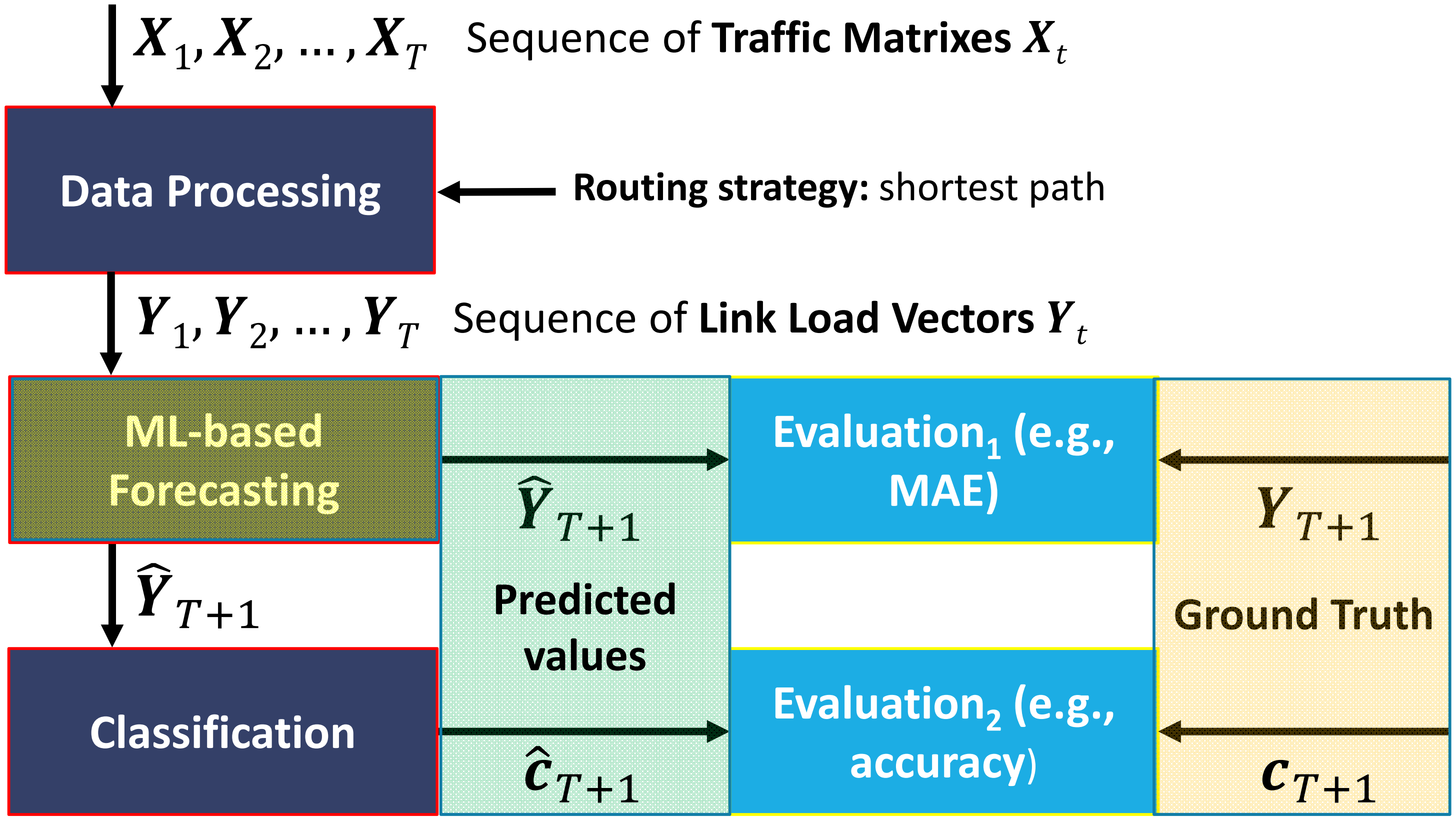
- Given:
  - A telecom network with **N** nodes and **M** unidirectional links
  - **T** traffic matrices (NXN)
    - The t^{th} traffic matrix represents the volume of aggregated traffic exchanged between each pair of network nodes during the t^{th} time slot
    - t ranges from 1 to T (i.e., number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

Compute the load on each link

Forecast the next load on links

Classify as congestion if traffic is above a threshold

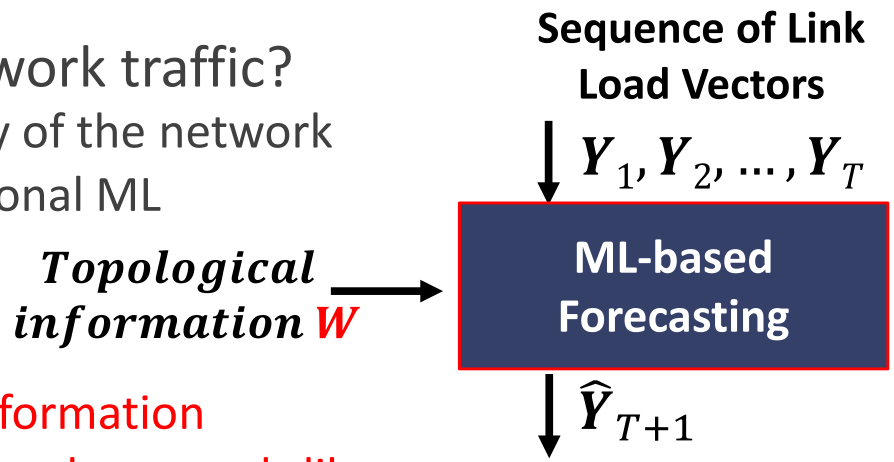


Proposed Approach (2)

- Regression Problem: $\widehat{\boldsymbol{Y}}_{T+1} = \mathcal{F}(\boldsymbol{Y}_1, \boldsymbol{Y}_2, \dots, \boldsymbol{Y}_T)$
  - 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: $\boldsymbol{Y}_t \in \mathbb{R}^{MX1}$ (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



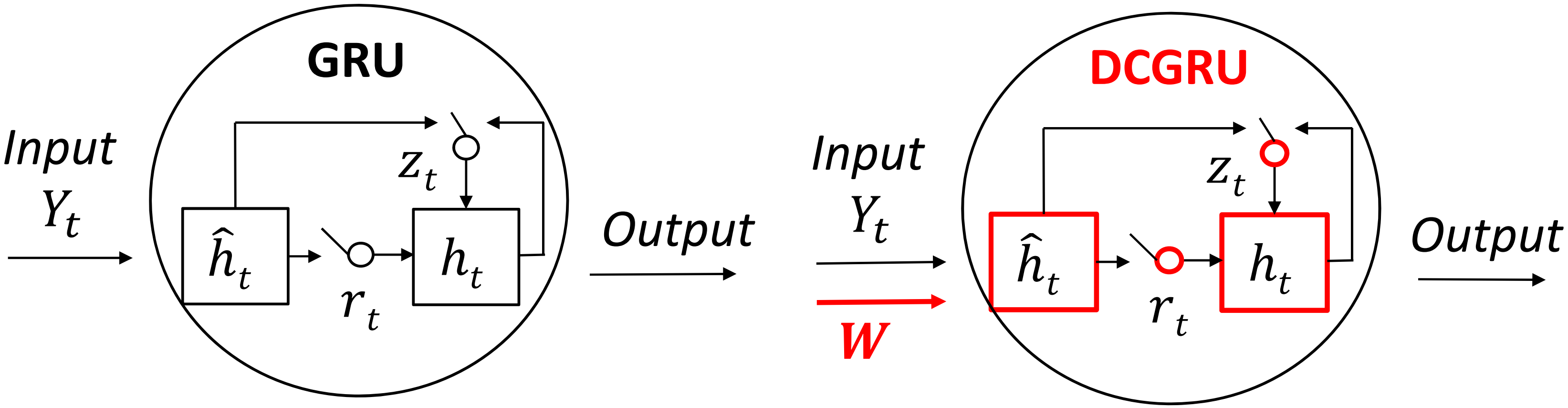
Proposed Approach (3)

- Starting from a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{W})$, with *V* set of nodes, *E* set of edges and *W* the adjacency matrix
- We represent $\boldsymbol{G}$ by its attributes that are $\boldsymbol{Y}_t$ and $\boldsymbol{W}$
- Nodes' feature vector $\boldsymbol{Y}_t \in \mathbb{R}^{Mx1}$ encodes the volume of traffic on each link at time t
  - M is the number of links
- Topology's feature matrix: $\boldsymbol{W} \in \mathbb{R}^{NxN}$ 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 reccurrent 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

Candidate activation: $\hat{h}_t$
Update gate: z_t
Activation: h_t
Reset gate: r_t



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

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} $\boldsymbol{W}$
    - D_0 is the out-degree diagonal matrix of G
    - W adjacency matrix of G

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

- Mathematically, this process can be expressed as a K-steps convolution between a graph signal $Y \in \mathbb{R}^{Mx1}$ and a filter f_θ [6]:
  - $Y \circledast f_\theta = \sum_{k=0}^{K-1}\left(\theta_{k,1}(D_0^{-1}\ \boldsymbol{W})^k + \theta_{k,2}(D_0^{-1}\ \boldsymbol{W}^T)^k\right)Y$
  - $\theta \in \mathbb{R}^{Kx2}$, parameters of the filter

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

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

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-mintutes 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 perfomed minimizing the Mean Absolute Error (MAE)
  - $MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$

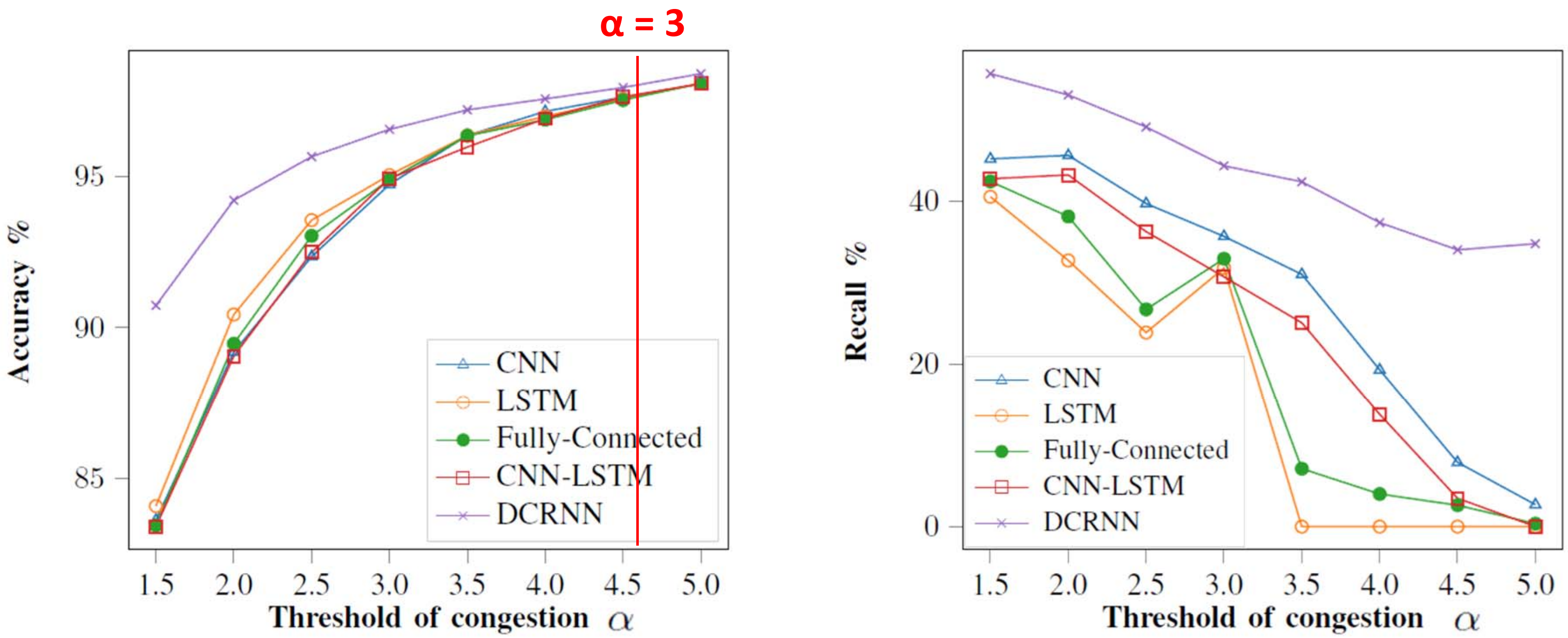
Experiments – Results (1)

	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**

- 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



(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

- **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)

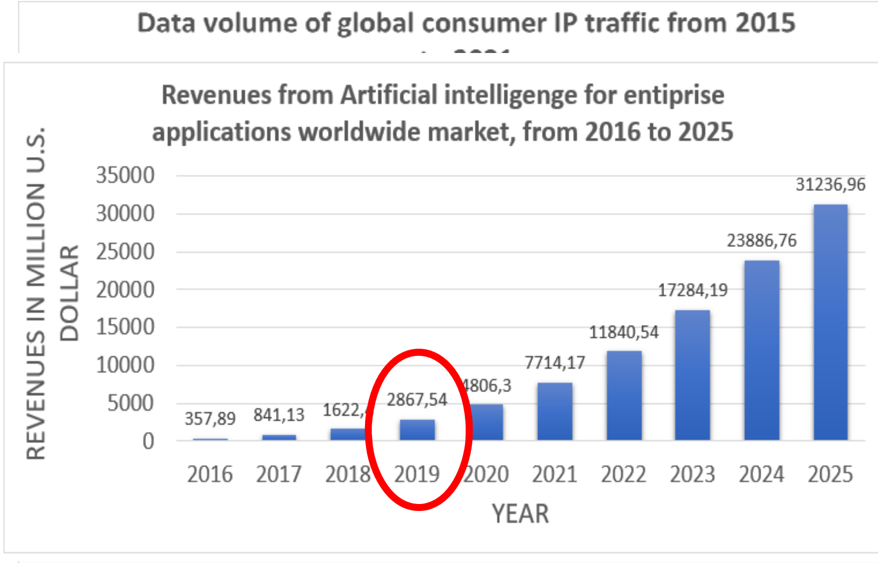
$\alpha = 3$

	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

- 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)



Global view of the network Flexible traffic steering

Software-Defined Networking **+**

Easy access to network statistics (e.g., load on links)

Classify network events

Machine Learning

Forecast network events

= Intelligent network management

[2] https://www.statista.com/statistics/267202/global-data-volume-of-consumer-ip-traffic/

Deep Learning for Traffic Prediction

- We need to learn patterns from a sequence to perform the forecast $\widehat{\boldsymbol{Y}}_{T+1} = \mathcal{F}(\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_T)$: 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)

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

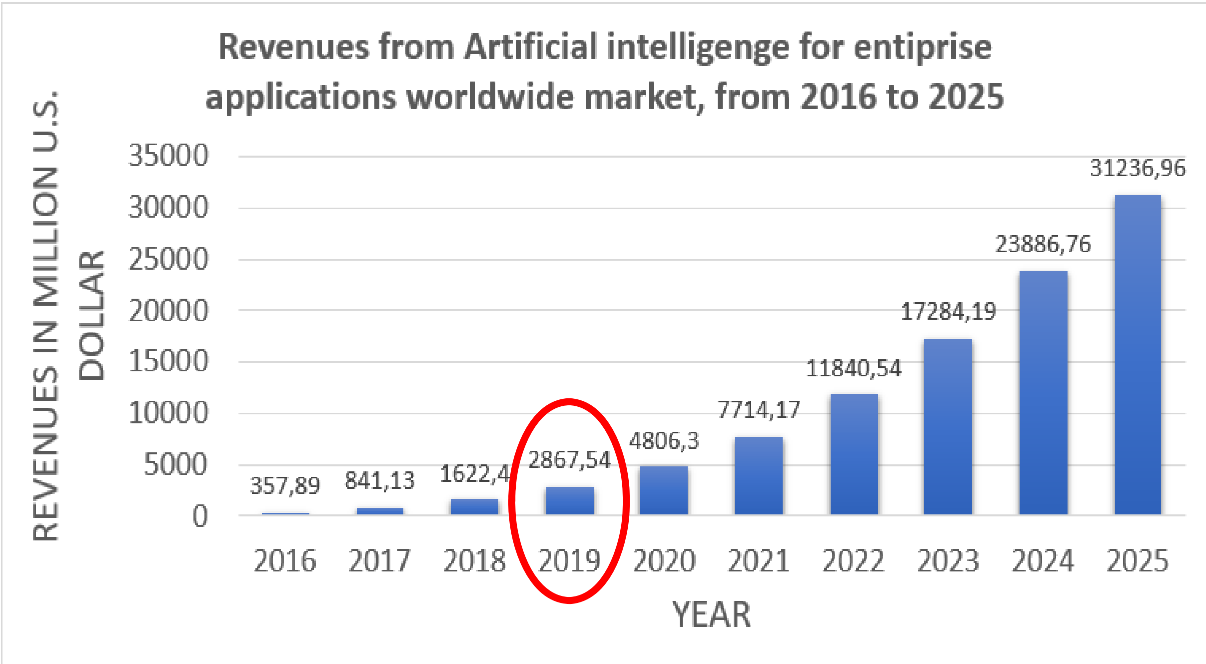
Topological Information

Graph's Nodes Information

[4] K. Cho, et al.,"On the properties of neural machine translation: Encoder-decoder approaches", arXiv preprint arXiv:1409.1259, 2014.

Introduction (2)

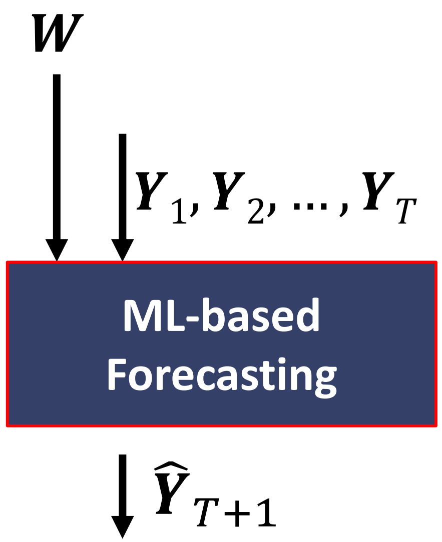
- High interest in Machine Learning, applications are growing rapidly[2]



[2] https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/

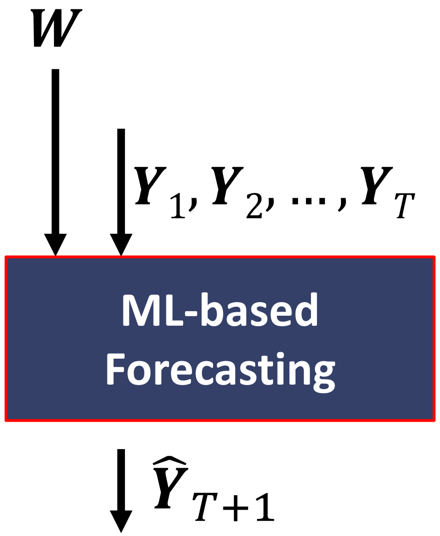
Proposed Approach (3)

- We model the spatial dependency by relating traffic flow to a diffusion process [6]
- Given a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ 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 $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$
  - Nodes' Feature Matrix: $\boldsymbol{F} \in \mathbb{R}^{NxP}$
    - i-th row of $\boldsymbol{F}$ represents the feature vector of node $v_i \in V, \forall i$ (P number of features)
  - Topology's Feature Matrix: $\boldsymbol{W} \in \mathbb{R}^{NxN}$
    - $\boldsymbol{W}$ 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 $\boldsymbol{G}$:
  - Nodes' feature vector $\boldsymbol{Y}_t \in \mathbb{R}^{Mx1}$
    - $v_i \in V$ represents the i-th network link
    - $\boldsymbol{Y}_t$ encodes the volume of traffic on each link at time t
  - Topology's Feature Matrix $\boldsymbol{W}$, whose ij-th entry is 1 if i-th and j-th link are connected, 0 otherwise

Proposed Approach (3)

- Given a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, with *V* set of nodes and *E* set of edges
- We represent $\boldsymbol{G}$ in a different way, considering: *V* the number of links M and *E* the adjacency matrix W
- Nodes' feature vector $\boldsymbol{Y}_t$, $\in \mathbb{R}^{Mx1}$, encodes the volume of traffic on each link at time t
- Topology's feature matrix: $\boldsymbol{W} \in \mathbb{R}^{NxN}$
  - $\boldsymbol{W}$ 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