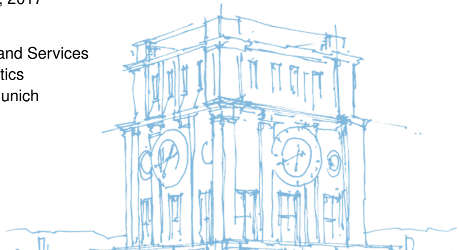
Performance Evaluation of Network Topologies using Graph-Based Deep Learning

Fabien Geyer

fgeyer@net.in.tum.de

Thursday 30th November, 2017

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich



Motivation

Performance evaluation for network topologies

Evaluating the performance of network protocols requires understanding:

1. The details of the protocols used
2. The network topology and the routing used for the flows
3. The interdependence between flows at the different bottlenecks

Drawbacks of current machine learning approaches

- Those three points are encoded as high-level and protocol-specific features: RTT, packet loss, . . .
- The network topology is always indirectly taken as input

Question: **Is there a more general approach to applying ML for the evaluation of network topologies?**

Motivation

Proposed approach

Key concepts

- A network topology is abstracted as a graph
- Flows are special nodes in the graph
- Use a neural network which can be trained on graphs

Advantages

- No need to engineer input features
- Any topology can be evaluated

Outline

General approach

From network topology and flows, to paths and queues, and to graphs

- Computers and switches/routers are abstracted as queues
- Queues are vertex, connected according to the different physical links
- Each flow is represented by a vertex
- The path of a flow is represented by edges connected to the traversed queues

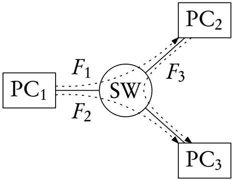


Figure 1: Example network topology

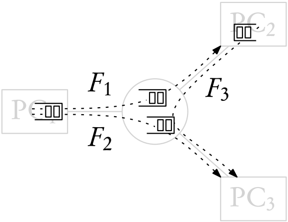


Figure 2: Associated queuing network

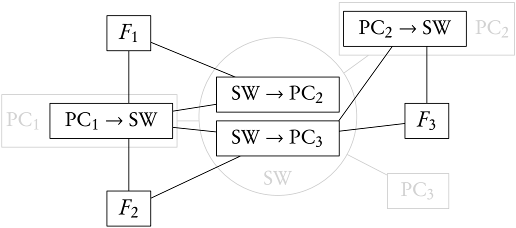


Figure 3: Graph representation of the example network topology

Graph Neural Networks

Key concepts of Graph Neural Networks [1, 3]

- Each node i has a hidden state vector $\mathbf{h}_i \in \mathbb{R}^k$
- The hidden state depends on the neighboring nodes

$$\mathbf{h}_i = f\left(\mathbf{h}_{n_1}, \dots, \mathbf{h}_{n_N}\right)$$

- Use fixed-point evaluation to compute $\mathbf{h}_i$

$$\mathbf{h}_i^{(t)} = f\left(\mathbf{h}_{n_1}^{(t-1)}, \dots, \mathbf{h}_{n_N}\right)^{(t-1)}$$

Key concepts of Gated Graph Neural Networks [2]

- Store hidden state in memory unit (GRU)
- Unroll the loop for a fixed number of iterations
- f as sum, such that using the graph adjacency matrix $\mathbf{A}$:

$$\begin{bmatrix} \mathbf{h}_1^{(t)} \\ \vdots \\ \mathbf{h}_N^{(t)} \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{h}_1^{(t-1)} \\ \vdots \\ \mathbf{h}_N^{(t-1)} \end{bmatrix}$$

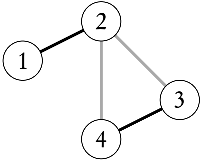


Figure 4: Example graph

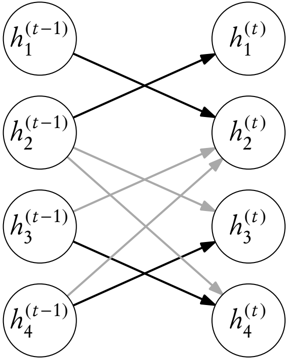


Figure 5: Propagation of hidden state

Graph Neural Networks

Features

Nodes have input features

- Encode node type as one-hot vector: $[0, 1]$ for queues, $[1, 0]$ for flows
- Encoding of other parameters also possible: packet size, protocol type, . . .
- Encoded in initial value of hidden vector $\mathbf{h}_k^{(0)}$

Graph Neural Networks

Final neural network architecture

- Memory units are used to compute dependencies over multiple loop iterations
- After the iterations, use feed-forward neural network for computing the output vector
- Standard gradient-based techniques can be used for training

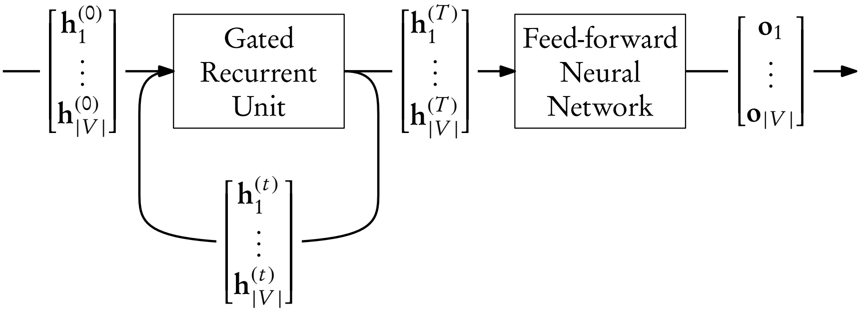


Figure 6: Overall architecture of the neural network

Numerical results

Studied use-cases

Evaluation of TCP flows

- Prediction of average bandwidth
- Graph extended for taking TCP ACKs into account

Traffic models used

- Infinite flows
- Finite ON/OFF flows:
  - ON: Sending files with random size following an exponential distribution
  - OFF: Idle for a random duration following an exponential distribution
- Mix of finite and infinite flows

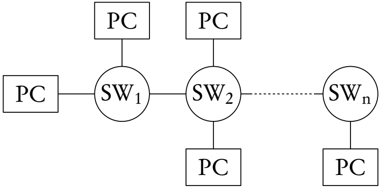


Figure 7: Evaluated network topology

Property	Mean	Min.	Max.
Number of flows	16.38	2	30
Number of queues	35.54	4	66
Number of edges	250.47	20	596

Table 1: Parameters of the dataset with the largest network topologies.

Numerical results

Evaluated neural network architectures

- RNN: Architecture similar to the original Graph Neural Network [1, 3] without memory unit
- GG-GRU-NN: Original GGNN architecture from [2]
- GG-LSTM-NN: Proposed extension of [2] with LSTM unit *(Long Short-Term Memory)*

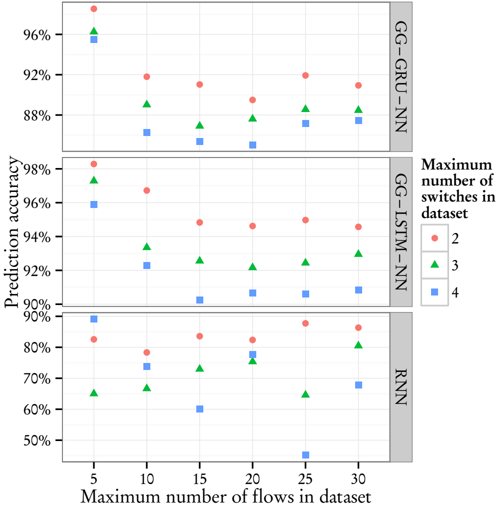
Metrics for evaluation

- Bandwidth discretized in order to do classification instead of regression
- Accuracy = 1 if correctly classified, 0 otherwise

Implementation done using Tensorflow

Numerical results

Accuracy with infinite flows



Numerical results

Accuracy with finite flows

Finite ON/OFF flows

- ON: Sending files with random size following an exponential distribution
- OFF: Idle for a random duration following an exponential distribution

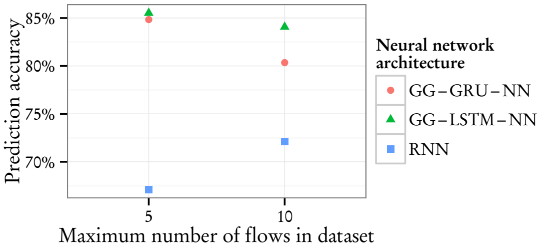


Figure 8: With only finite flows

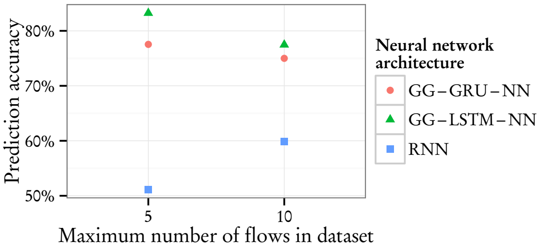
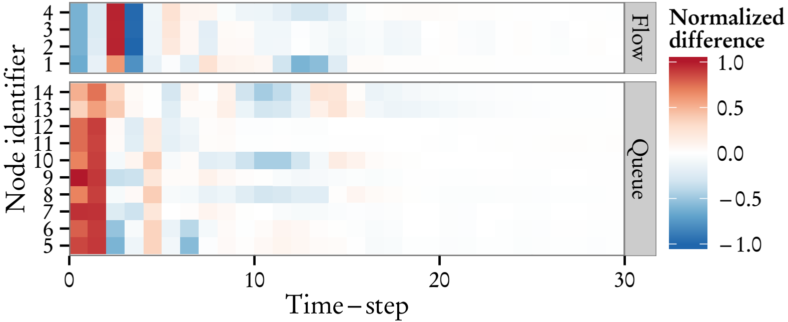


Figure 9: With finite and infinite flows

Numerical results

Interpretation of the learned weights

- Visualization of $\mathbf{h}_i^{(t)} = f\left(\mathbf{h}_{n_1}^{(t-1)}, \dots, \mathbf{h}_{n_N}\right)^{(t-1)}$ as t increases
- How does the hidden representation of a node evolve at different time-steps?



Conclusion and future work

Summary

- **New approach for applying machine learning to network topologies and flows**
- No need for high-level engineered features, only the **graph representation of the topology**
- **Generic model** which can be applied to various protocols
- Application to **prediction of TCP bandwidth** with promising accuracy

More to come

- Study of other protocols and metrics
- Application to network engineering and network optimization

Bibliography

[1] M. Gori, G. Monfardini, and F. Scarselli.
A New Model for Learning in Graph Domains.
In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*, volume 2 of *IJCNN'05*, pages 729–734. IEEE, Aug. 2005.

[2] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel.
Gated Graph Sequence Neural Networks.
In *Proceedings of the 4th International Conference on Learning Representations*, ICLR'2016, Apr. 2016.

[3] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini.
The Graph Neural Network Model.
20(1):61–80, Jan. 2009.