JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Modelling SDN using Graph Neural Networks

[1]Sanjana, [2]Dr. Nitin Kumar
Department of CSE
Ganga Institute of Technology & Management, Haryana, India

Abstract : **Building self-driving Software-Defined Networks requires network modeling, particularly to identify the best routing protocols that satisfy the objectives stated by administrators. However, the requirements for providing precise estimates of pertinent performance indicators like latency and jitter are not addressed by present modeling methodologies. In this study, we provide a unique Graph Neural Network (GNN) model capable of comprehending the intricate link between topology, routing, and input traffic to generate precise estimates of the mean delay and jitter for each source/destination pair. Because GNN is designed to learn and model data that is organized as graphs, our model can generalize over any topologies, routing protocols, and varying traffic intensity.**
Additionally, we demonstrate the model's potential for network operation through the presentation of a number of use cases that demonstrate its successful application in the optimisation of delay and jitter for each source-destination pair as well as its generalization abilities by reasoning in topologies and routing schemes that were not encountered during training.

IndexTerms - Software Designed Network,Graph Neural Network,Neural Network

1.INTRODUCTION

A centralized management of the network is made possible by software defined networks, which enable the separation of the control plane from the forwarding plane. Instead of allowing routers to define their own forwarding strategies based on routing algorithms that lack a context of the entire network, software defined networks allow a centralized authority to decide the forwarding strategy and enforce it over the network's devices.

The key to achieving effective network operation in future self-driving networks will be efficient network models in this case. The goal of this dissertation is to create models that can assist in predicting crucial performance metrics like delay, jitter, and loss even over training topologies that aren't yet known. The primary objective of network optimisation is to operate networks as effectively as possible. Using the SDN paradigm, network optimisation is accomplished by adding the following two elements to the SDN controller: An optimisation method and a network model, respectively. The optimisation method that utilizes the network model and the network policy is typically configured by the network administrator to provide the configuration that achieves the goals.
In this classic and well-known design, the model is in charge of forecasting the network's performance (e.g., per-link utilization) and topology (e.g., routing) for a certain configuration. The optimisation algorithm is then tasked with looking through the configurations to find one that satisfies the network administrator's objectives. Traffic engineering is an illustration of this, where the objective is to create a routing configuration that maintains the per-link utilization below the per-link capacity. Effective optimisation solutions use expert knowledge to minimize the configuration's normally extremely high dimensionality.
We can only optimize what we can model, which is one of network optimization's core properties.

For instance, we want a model that can comprehend how jitter links to other network properties in order to optimize the jitter of the packets crossing the network. Many precise network models have been created in the past for fixed networks, especially when applying queuing theory. However, these models simplify things by assuming several unrealistic characteristics of real-world networks, such as traffic generation with a Poisson distribution and probabilistic routing. Additionally, they struggle to solve networking issues including estimate of end-to-end performance measurements and multi-hop routing (also known as multi-point to multi-point queueing). Due to the fact that they are insufficient for big networks with actual routing topologies, delay, jitter, and losses continue to be important performance measures for which no workable model is available.

In-Depth Learning. The networking community has been interested in attempting to employ these cutting-edge approaches to create a new breed of models that are specifically centered on complicated network behavior and/or measurements.

In this situation, pertinent study is being done in this new subject. Researchers are utilizing neural networks to simulate computer networks and to optimize networks, often in conjunction with cutting-edge Deep Reinforcement Learning techniques.

These ideas frequently make use of well-known neural network (NN) designs, such as fully-connected neural networks, convolutional neural networks (extensively used for image processing), recurrent neural networks (used for text processing), or variational auto-encoders. These kinds of NN are not intended to learn data organized as graphs, despite the fact that computer networks are primarily represented as graphs. The models developed as a result have severe limitations, such as poor accuracy and a lack of topological or routing configuration generalization.

This is one of the key reasons why ML-based network optimisation strategies have, as of this writing, fallen short of their goals and outperformed conventional methods by a wide margin.

2. NETWORK ARCHITECTURE

The control plane can use network modeling to fully utilize SDN's capability for performing fine-grained management. This makes it possible to assess the performance of what-if scenarios without having to change the data plane's current state. Several network management applications, including optimisation, planning, and quick failure recovery, can be profitable.

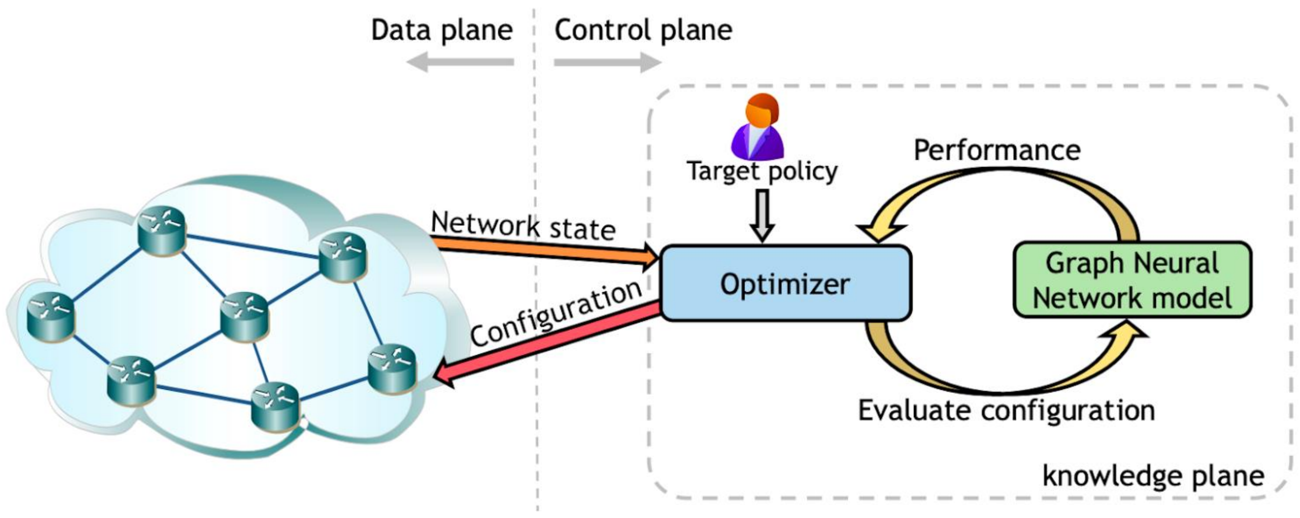

Figure 1:Architecture for Network Optimization

Similar to this, there is an optimizer in the knowledge plane whose actions are constrained by a set goal policy. A declarative language like NEMO may be used to express this policy, which is in accordance with intent-based networking, before being converted into a (multi-objective) network optimisation problem. By using it to conduct optimisation algorithms (like hill-climbing) that iteratively examine the performance of potential solutions in search of the ideal configuration, an accurate network model may thus play a critical part in the optimisation procedure. The training step is purposefully excluded from the scope of this architecture.

The network model must satisfy two key criteria in order to be effective in scenarios like the one described above: (i) delivering accurate results; and (ii) having a low computing cost to enable network optimizers to work in short time scales. The ability to simulate what-if scenarios including various routing protocols, topological modifications, and variations in the traffic matrix is also crucial for optimizers. In order to do this, we depend on Graph Neural Network (GNN) models' capacity to function and generalize well in graph-based contexts.

The Message-Passing Neural Network utilized in the chemistry field served as the basis for our GNN-based model, RouteNet, which can propagate any routing scheme throughout a network topology and abstract relevant information about the current network state. A schematic illustration of the model may be seen in Fig. 1. To put it more specifically, RouteNet receives as inputs (i) a specified topology (ii) a source-destination routing scheme (i.e., relationships between end-to-end pathways and connections) (iii) and (iv) a traffic matrix (i.e., the bandwidth between each pair of nodes in the network), and then creates (v) performance indicators based on the status of the network today.

3. NETWORK MODELING WITH GNN

The process of exchanging data or messages between nodes in a graph-based structure is known as message passing in neural networks. The ability to propagate and aggregate information across the graph is a fundamental function of graph neural networks (GNNs).

Message computation and aggregation are two crucial steps in the message passing process for GNNs. Every node in the graph receives messages from the nodes that are close to it, processes those messages, and then creates a brand-new representation or message. In order to create an updated representation for each node, these computed messages are then combined.

During the message computation stage, a neural network operation is applied to the incoming messages as well as the node's own features. This process usually entails a transformation function that combines the node's features with the incoming messages, like a neural network layer or a graph convolutional operation. This step serves the purpose of updating the node's representation in light of the data received from its neighbors.Following the computation of the messages, the computed messages are combined or aggregated to produce an updated representation for the node in the aggregation step. In order to determine the relative importance of various messages, this aggregation process may include various operations like adding, subtracting, or averaging.In GNNs, message passing is typically carried out iteratively over a number of layers, enabling nodes to gather and spread information throughout the graph. Based on the data gathered from their neighbors, nodes can improve their representations during each iteration. With the aid of this iterative process, GNNs can detect intricate dependencies and connections within the graph structure.Message passing allows information to flow and communication between nodes in a graph, which is a critical function of GNNs. It enables GNNs to take advantage of the innate graph structure and capture significant relationships, making them suitable for a range of graph-based tasks, including node classification, link prediction, and graph-level predictions.

In chemical and material science contexts where we have graph data, the Message Passing Neural Network is a deep learning architecture that is intended for implementation.

4. EVALUATION OF THE ACCURACY OF THE GNN MODEL

In this part, we assess RouteNet's precision in estimating the mean delays and jitter for each source and destination for various network topologies and routing strategies.

4.1 Setup for simulation

We used OMNeT++ (version 4.6) to develop a bespoke packet-level simulator with queues in order to provide a ground truth to train and assess the GNN model. The bandwidth capacity of the relevant egress connections is connected to the latency and jitter models in each queue in this simulator. We calculate the average end-to-end latency and jitter for every pair of nodes during a 16k-unit time period for each simulation. For each S-D pair in the network, we model the traffic matrix (TM) as follows: TM(Si,Dj) = U(0.1, 1) TI/(N 1) i, j nodes,i, j

Where N is the number of network nodes, TI is a parameter to adjust the total traffic intensity in the network scenario, and U(0.1, 1) is a uniform distribution in the range [0.1, 1].

We employed the 14-node and 21-link NSF network design to train and assess the model.

4.2 Training and Evaluation

We used TensorFlow to construct the RouteNet delay and jitter models. All of the training and assessment datasets utilised in this study, together with the source code, are accessible online.

Using our packet-level simulator, we created a collection of 260,000 training samples from the NSF network, which we used to train both models (jitter and delay). Even though this dataset only comprises samples from a single topology, it has a wide range of traffic matrices with various levels of traffic intensity, as well as almost 100 distinct routing strategies. Thirty thousand samples are used in the examination.

In our studies, we use a size of 32 for the hidden states on the route (hp) and a size of 16 for the hidden states on the link (hl).

The bandwidth that each source-destination path transports (extracted from the traffic matrix) determines the initial path characteristics (xp). We don't add initial link features (xl) in this situation. Be aware that greater sizes for the concealed states may be required for larger networks. Additionally, we do T = 8 repetitions for each forward propagation. The readout neural network's dropout rate is equal to 0.5, which indicates that we randomly deactivate half of its neurons after each training step. This enables us to sample the findings probabilistically and determine the degree of confidence in the estimations.

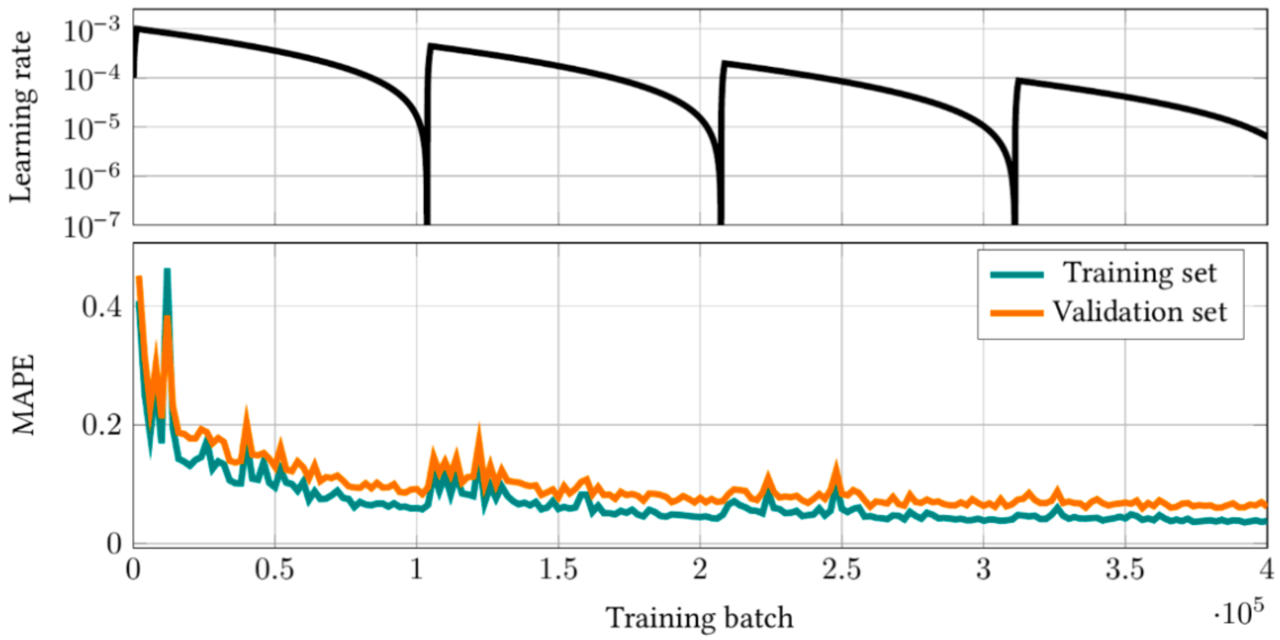

Figure 2:Learning Rate and MAPE

The mean squared error (MSE) between RouteNet's prediction and the actual data, as well as the L2 regularisation loss (= 0.1), were both minimised during training. Using an Adam optimizer with a 0.001 starting learning rate, the loss function is reduced. After around 60,000 training steps, this rate drops to 0.0003.

Over 300,000 batches of 32 randomly chosen samples from the training set, we conducted the training. This took around 96 hours on our testbed using a GPU Nvidia Tesla K40 XL (27 samples per second). The loss experienced during training is seen in Figure 2. Here, we see that the loss is rapidly declining while the training is steady.

5.CONCLUSION

With 750 000 training samples, we were able to achieve a MAPE of 1.66% on the evaluation dataset. On 750 000 instances of training data, we trained the model. There was a 4-hour training time limit because the model was primarily trained on Google Colab. In order to run the model more than once, we had to save checkpoints.

In comparison to monotonically-decreasing schedulers, we discovered that Pytorch's Cyclic Learning Rate Scheduler enabled our models to achieve lower error values. It makes intuitive sense that this scheduler avoids local minima by "jumping" to unexplored areas of the parameter space with a high k learning rate.

This becomes very apparent around batch 100000: following a gradual decline, the learning rate suddenly jumps back to around 5 104. The training loss then rises for approximately 50 000 batches before falling. It is important to note that the loss over the validation dataset follows the loss over the training dataset with a reasonable difference and demonstrates no evidence of overtraining. The graph between the anticipated delay and the model-predicted delay can be seen in Figure 3. The predicted values are represented in the graph by dots. The majority, as we can see.

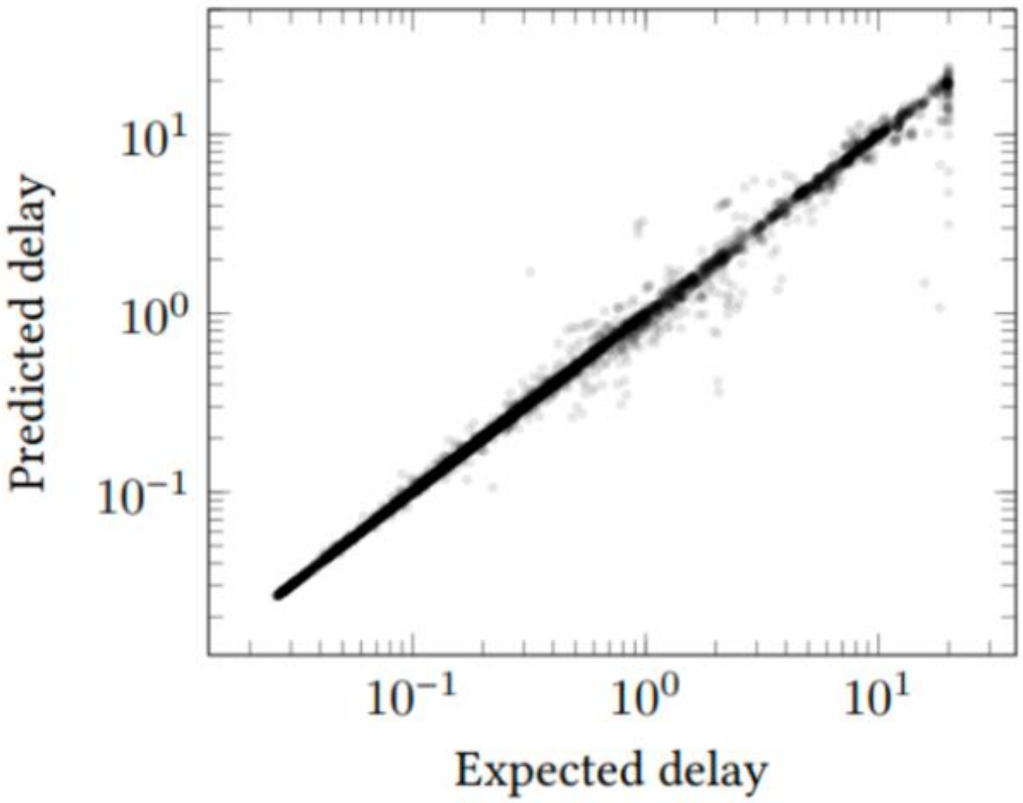

Figure 3:Expected Delay vs Predicted Delay

References

[1] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, “Software-Defined Networking: State of the Art and Research Challenges,” arXiv e-prints, p. arXiv:1406.0124, May 2014.

[2] M. Zukerman, “Introduction to Queueing Theory and Stochastic Teletraffic Models,” arXiv e-prints, p. arXiv:1307.2968, July 2013.

[3] A. Varga, OMNeT++, pp. 35–59. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[4] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” CoRR, vol. abs/1609.02907, 2016.

[5] T. N. Kipf and M. Welling, “Variational Graph Auto-Encoders,” arXiv e-prints, p. arXiv:1611.07308, Nov. 2016.

[6] X. Li and Y. Cheng, “Understanding the Message Passing in Graph Neural Networks via Power Iteration Clustering,” arXiv e-prints, p. arXiv:2006.00144, May 2020.

[7] K. Rusek, J. Suarez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, ´ 38
“RouteNet: Leveraging Graph Neural Networks for network modeling and optimization in SDN,” arXiv e-prints, p. arXiv:1910.01508, Oct. 2019.

[8] S. Xiao, D. He, and Z. Gong, “Deep-q: Traffic-driven qos inference using deep generative network,” Proceedings of the 2018 Workshop on Network Meets AI & ML, 2018.

[9] A. Mestres, E. Alarcon, Y. Ji, and A. Cabellos-Aparicio, “Understanding ´ the modeling of computer network delays using neural networks,” CoRR, vol. abs/1807.08652, 2018.

[10] F. Geyer and G. Carle, “Learning and generating distributed routing protocols using graph-based deep learning,” in Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, Big-DAMA ’18, (New York, NY, USA), p. 40–45, Association for Computing Machinery, 2018.

[11] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph Neural Networks: A Review of Methods and Applications," arXiv e prints, p. arXiv:1812.08434, Dec. 2018.

[12] "Knowledge defined networking datasets." https://knowledgedefinednetworking.org/. Accessed: 2021-02-27.

[13] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in Proceedings of the 34th International
Conference on Machine Learning (D. Precup and Y. W. Teh, eds.), vol. 70 of Proceedings of Machine Learning Research, pp. 1263–1272, PMLR, 06–11 Aug 2017.

[14] "PyTorch geometric MessagePassing." https://pytorch-geometric.readthedocs.io/. Accessed: 2021-06-24.

[15] K. Rusek, J. Suarez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, ´ "Unveiling the potential of graph neural networks for network modeling and opti mization in sdn," in Proceedings of the 2019 ACM Symposium on SDN Research, SOSR '19, (New York, NY, USA), p. 140–151, Association for Computing Ma chinery, 2019.

[16] J. Zhang and L. Meng, "Gresnet: Graph residual network for reviving deep gnns from suspended animation," CoRR, vol. abs/1909.05729, 2019.

JETIR