Path-Link Graph Neural Network for IP Network Performance Prediction

Yangzhe Kong*†, Dmitry Petrov†, Vilho Räisänen†, Alexander Ilin*
*Aalto University, Espoo, Finland {first.last}@aalto.fi
†Nokia Bell Labs, Espoo, Finland {dmitry.a.petrov, vilho.raisanen}@nokia-bell-labs.com

***Abstract*—Dynamic resource provisioning and quality assurance for the plethora of end-to-end slices running over 5G and B5G networks require advanced modeling capabilities. Graph Neural Networks (GNN) have already proven their efficiency for network performance prediction. GNN architecture matches well the structures usually met in communications networks. In this paper, the focus is on the IP transport network as one of the end-to-end 5G architecture domains. The recently published RouteNet GNN is taken as a reference and starting point for our study. RouteNet performance is verified by a new implementation in the PyTorch ML library. Next, an alternative Path-Link neural network (PLNet) architecture is proposed and evaluated. After hyper-parameter tuning for both models, the results show that PLNet and RouteNet achieve a similar accuracy level. The advantage of PLNet is in parallel architecture. It is demonstrated that its inference speed is not sensitive to the length of the network's paths.**

I. Introduction

The emerging 5th generation (5G) networks have significant architectural improvements compared with 4G. Indeed, the introduction of new engineering approaches is caused by the need to support very different categories of service requirements: ultra-low latency, extremely high data rate, low energy consumption, and low-cost [1]. Achieving this flexibility with 4G service provisioning would require significant effort. 5G and beyond 5G (B5G) networks allow different entities and stakeholders to instantiate and run virtually isolated networks with diverse business demands and priorities on the same shared infrastructure [2], [3]. These differentiated network services, called slices, possess individualized requirements typically defined by the requirements on bandwidth, packet loss ratio, delay, jitter, etc. Therefore, it becomes essential to assure service SLAs and manage traffic flows dynamically and intelligently for the coexistence of slices and optimal use of equipment [4], [5].

As pointed out in [6], unlike mostly hardware-driven advancements seen in the previous network generations, the revolution of 5G also necessitates for significant progress in software. Specifically, 5G networks provide very adaptive architecture for various types of services [7]. Network Function Virtualization (NFV), Software-Defined Networking (SDN), and Control and User Plane Separation (CUPS) are among the key enabling technologies of 5G. A hierarchy of clouds spanning from the far edge and up to core and public data centers gives the possibility to execute Virtual Network Functions (VNF) and applications in the optimal locations. However, it is still necessary to take into account the availability of computational and storage resources, transport link capacities, and traffic demands over all possible slice deployments [5].

978-3-903176-32-4

Indeed, the complexity and flexibility of 5G require new capabilities and advanced automation from end-to-end network management and orchestration. It is not sufficient that each of the network domains is orchestrated separately to guarantee end-to-end Quality of Service (QoS). Necessary interfaces and overall architecture are defined by various standardization and open-source forums, including, for example, ETSI Zero-touch network and Service Management (ZSM) [8].

Advanced modeling of end-to-end network performance and availability of domain resources act as a foundation for more efficient network operation and optimization solutions with lower computational costs. In this paper, we are focusing on the IP transport network as the concrete use-case. However, we believe that the study's findings can be applied in the broader scope for end-to-end network orchestration and slice provisioning tasks.

Despite the long history of algorithm development for KPI prediction and routing in IP networks, the research community is still looking for the tradeoff between computation efficiency and accuracy. Analytical models are efficient but cannot always achieve acceptable accuracy when network configuration becomes complex. It is also challenging to take into account non-intuitive behavior and the differences between the versions of transport-layer protocols [9]. On the contrary, detailed network simulators like ns-3 [10] or OMNeT++ [11] can produce accurate KPI predictions. However, this is achieved with high computational cost since every packet is modeled individually. The applicability of such simulators to real-time tasks is also limited due to the scalability problem when the number of packets or the size of the network increases. Hence, Machine Learning (ML) approaches, with their ability to learn from the past data and generalize [12], is a promising way forward. Consequently, there is increasing interest in utilizing ML-based methods for network modeling in traffic management tasks.

The first works that applied ML techniques for the performance evaluation of communication networks were already able to provide high prediction accuracy [13], [14]. These models assume the knowledge of high-level, manually defined features that also need periodical measurements. The majority

of the state-of-the-art approaches [15]–[17] use traditional Neural Network (NN) architectures. Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) models are known to be very efficient for computer vision [18] or natural language processing tasks [19]. However, they cannot perform that well when applied directly to the objects with a specific internal organization such as a communication network. The main reason is that fully-connected layers induce limitations on the entities and relations they are modeling [20], [21]. Therefore, once the network topology changes, the model must be re-trained or re-designed.

Recently, several papers [20], [22]–[24] proposed GNN models for network optimization and traffic management. The authors show that the GNN architecture is appropriate for learning the underlying relationships over graph-like structures and, therefore, can generate accurate KPI predictions for network topologies and routing schemes not present in the training data-set. Importantly, this approach can be effectively used for large topologies with thousands of links and communication nodes and as a part of deep reinforcement learning algorithms [25].

For this study, the version [26] of the RouteNet model [22] is used as a starting point. Firstly, we re-implement RouteNet with PyTorch library [27]. It is trained and cross-checked against the original TensorFlow implementation. Then, we construct a new Path-Link GNN (PLNet) with a simpler architecture. Our modeling assumption is that the communication network can be presented as a bipartite graph. The results show that the sequence of links in a path has little impact on the data flow performance. Additionally, we tune the hyper-parameters of both RouteNet and PLNet. After that, it is demonstrated that PLNet can archive similar path delay prediction accuracy as RouteNet. The significant fact is that PLNet inference speed is not dependent on the number of links in the path. Furthermore, we consider several ways to implement PLNet and simplify its design to optimize for execution speed at the price of a bit lower accuracy.

The rest of the paper is organized as follows. In the next Section, the studied scenario and data-sets are introduced. In Section III, we describe how GNNs can be applied in this context. RouteNet architecture is explained, and it is shown how it can be made more efficient in PLNet. Section IV is devoted to the performance comparison of RouteNet and PLNet and the analysis of the results. Finally, we draw the conclusions in Section V.

II. Experimental setup

For training and evaluation of the models, we use publicly available Knowledge-Defined Networking (KDN) training data-sets [28] generated with packet-level simulator OMNeT++.

The considered IP network has the form of a graph without disconnected nodes. Each node in the modeled is a router. Depending on the destination address, it can forward IP packets through one of the interfaces. The number of interfaces is defined by the number of one-hop links from the node to its neighbors. Each node can create packet flows to any other node in the network with specified traffic demands in Kbps. There are no particular assumptions about the implementation of the routing protocol. The routing table is given for each node at the beginning of the simulation and can be defined randomly, manually, or based on some algorithm that takes node, link, and route KPIs into account. In the simplest case, the routes are calculated based on the Shortest Path First (SPF) Dijkstra's algorithm.

TABLE I
Statistics of KDN training data-sets

Data-set	# nodes	# links	Avg. path length	Max path length
NSFNET	14	42	2.14	4
GBN	17	52	2.70	6
GEANT2	24	74	2.92	7
Germany50	50	176	4.05	12

In KDN training data-sets data-sets, it is assumed that each node of the network can send packets to any other one. We agree that in 5G deployments, the traffic can be more directive, i.e., from the RAN towards the Core. However, the introduction of edge computing will make it more distributed. Therefore, we believe it is still a valid scenario and a good starting point both for the training and validation of the models.

We agree that in 5G deployments, the traffic can be more directive, i.e., from the RAN towards the Core. On the other hand, the introduction of edge computing will make it more distributed.

The KDN data-sets have been generated for IP networks with four real-world topologies: the 14-node NSFNET, the 17-node GBN, the 24-node GEANT2, and the 50-node Germany50 (see Table I for details). For each data-set, the capacity of each link is set to a fixed value, and the traffic demand d_{ij} between each source-destination pair (i, j) is randomly simulated as

$$d_{ij} = \frac{zt}{N-1}\,,\ i \neq j,$$

where z is drawn from a uniform distribution in the range $[0.1, 1]$, t is a parameter that controls the overall traffic intensity in the network simulation, and N is the number of nodes in the network. Each topology data-set includes simulation results from around 100 different routing schemes with varying traffic intensities, giving a total of 125,000 independent results. This could eliminate the dependency on any particular implementation of the routing algorithm or policy.

For each end-to-end path, the simulator produces multiple performance indicators, such as the average packet jitter and delay, the absolute number of transmitted and dropped packets.

In this paper, we concentrate on the prediction of the logarithm of the path average packet delay. Extending the models for predicting other performance indicators is straightforward.

III. Models for network performance prediction

Formally, each simulation is characterized by the following components: a set of nodes, a set of links between the nodes and a routing scheme which is represented by a set of paths (see Fig. 1 for a simple example). Each path consists of one or several links in a specific order. The links can have known characteristics such as the link capacity. The paths are assumed to have some known properties as well, such as the traffic demand.

A. RouteNet

Given a configuration of the IP network, the RouteNet predicts path KPIs by building a computational graph that contains T iterations. Each iteration t consists of two steps: 1) update of the path states p_i^t and 2) update of the link states l_j^t (see Fig. 2).

The states are inferred using Gated Recurrent Units (GRU) [29]. The GRU that updates the path states gets the current states of the links of the path as an input (top of Fig. 2). The GRU that updates the link states uses the sum of the messages from all the paths that contain the current link (bottom of Fig. 2). The messages to the link-state GRU are taken as intermediate hidden states of the path-state GRU. After the final T-th iteration, there is a readout Neural Network (NN) from the final path states to predict the path KPIs. The readout NN is implemented as a multi-layer perceptron (MLP).

RouteNet takes into account the order of the links in the paths. This is due to processing the links in the same order that they appear in the path when updating the path states p_i^t (top of Fig. 2). As stated by the authors of [22], the order of the links in end-to-end paths matters because the potential probability of packet loss and delays introduces sequential dependence between the link states. Thus, in each path, the relation between path delays and link delays may not be additive.

A potential for improvement of the RouteNet architecture is in removing the long sequence of path state updates. When a path contains many links, a computational graph with many intermediate layers is needed. This can slow down the training and inference procedures.

In this work, we propose GNN architecture, which ignores the order of the links in the paths. The IP network is presented as a bipartite graph that contains path and link nodes. The edge between a link node and a path node means that the link belongs to the path. An example of such a graph for the network from Fig. 1 is shown in Fig. 3. Our results

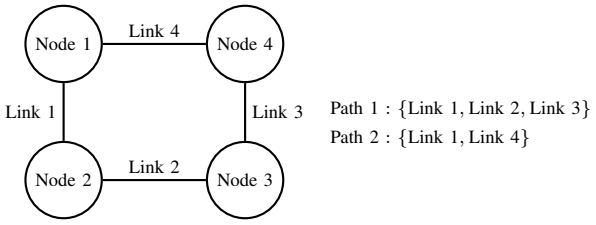


Fig. 1. An example of a network topology with two paths.

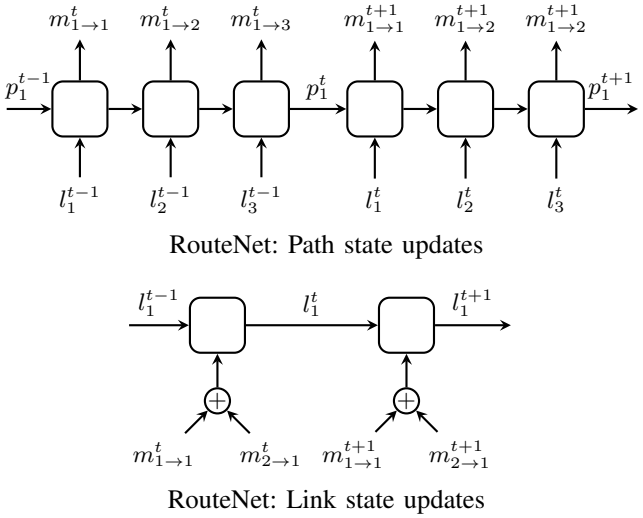


Fig. 2. Illustration of the RouteNet computational graph for the network topology from Fig. 1. Above: Updates of the states of path 1. Below: Updates of the states of link 1. Two graph iterations t and $t+1$ are shown. The square blocks represent GRU units. $m_{i\to j}^t$ denote messages from path i to link j which are aggregated to update the link states.

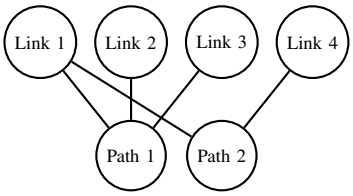


Fig. 3. A bipartite graph with path and link nodes which corresponds to the topology from Fig. 1. This representation of the IP network ignores the order of the links in paths.

demonstrate that the model can capture the inter-dependencies between the link properties without taking into account the order in which the links appear in the path.

B. PLNet

We construct PLNet to predict the IP network KPIs by applying the message passing methodology [21], [30] applied to the bipartite path-link graph, such as the one shown in Fig. 3. In each iteration of our GNN model, we compute the messages to all the nodes in the path-link graph by using message networks called MsgNet.

MsgNets take as inputs the current states of the source and destination nodes. There are several architectural choices avaliable, such as GRU or MLP. We will show below that using GRU instead of MLP as MsgNet can help to avoid exploding gradient problem and thus achieve better performance. Hence, the states of the nodes are updated using GRUs, similar to the RouteNet.

Finally, we apply a readout network that predicts the path KPIs from the final path states. The readout network is implemented as MLP. There is no limitation on number of KPI predictions PLNet can make for each path. One can train PLNet to predict multiple KPIs by simply manipulating the

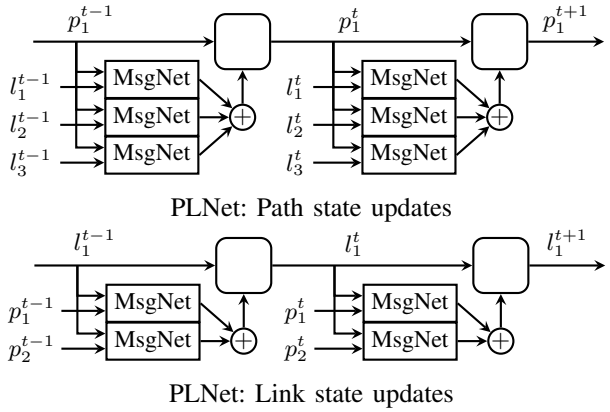


Fig. 4. Illustration of the PLNet computational graph for the network topology from Fig. 1. Above: Updates of the states of path 1. Below: Updates of the states of link 1. Two graph iterations t and $t+1$ are shown. The square blocks represent GRU units.

dimensionality of the outputs as long as any data-set with such KPIs is available.

The computational graph of PLNet is illustrated in Fig. 4. One can see that the links and the paths are handled in the model in the same way. The detailed listing of PLNet algorithm can be found in the Appendix.

We evaluate four versions of the PLNet architecture. The default versions that we denote PLNet-MLP and PLNet-GRU use two distinct message-processing MsgNets of MLP and GRU type, respectively. One of the MsgNets computes messages to the paths and another one to the links. The simplified versions of PLNet that we denote PLNet-MLP-1 and PLNet-GRU-1 use the only one MsgNet of MLP/GRU type for all message computations. Exclusion of the second MsgNet speeds up the training and inference procedures because the computations of the messages are further parallelized.

IV. Comparison of the models

A. Comparison of prediction accuracy

Firstly, we test the performance of all five models, RouteNet, PLNet-GRU, PLNet-MLP, PLNet-GRU-1, and PLNet-MLP-1, on the GBN data-set. It is split into training, validation, and test parts in proportion 80–10–10%. The number of message-passing iterations is set to $T = 8$ for all models. As the loss function, we use the mean squared error (MSE) between the true log-delays and the predicted ones for all paths in the network. The neural network is regularized with weight decay procedure and dropout layers after the first two layers of the readout MLP for all the five models.

The Adam optimizer [31] is used. The models are trained for 100 epochs. The learning rate is reduced by half after the 15-th and 45-th epoch. The batch size is 32. The exponential moving average is applied to the weights with the decay parameter 0.999 to compute the weights of the final model. We clip the norm of the gradient with the upper threshold of 0.5.

We tune the hyperparameters of the models by doing a random search within a limited number of pre-defined options. It is identified that their selection can have a significant impact on the performance. For example, it was found out that there is no need to use weight decay at all. The optimal hyperparameter values are summarized in Table II, and the learning curves of the models with the best values of the hyperparameters are shown in Figure 5.

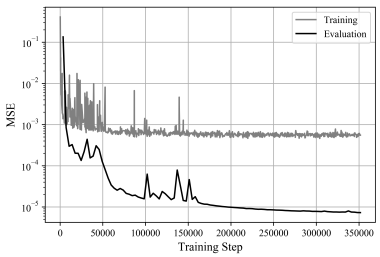
Next, the models are trained with optimal hyperparameters for multiple times. Their performance on the test set of GBN data-set with 95% confidence intervals are shown in Table III. PLNet models are slightly better than the RouteNet, and PLNet models with GRU MsgNet have better and more stable performance than PLNet models with MLP MsgNet. PLNet-GRU-1 and PLNet-MLP-1 have slightly higher losses than PLNet-GRU and PLNet-MLP, respectively, due to simpler architecture. The excellent performance of the models can also be observed from Figure 6 and 7 where we plot the target log delays against the predicted ones and the CDF of relative error. From Figure 6, it can be noticed that the models can produce reasonable confidence intervals using the dropout technique.

Regarding the comparison between PLNet models and routeNet, we can say that the simpler assumption that the order of the links in the paths doesn't matter is appropriate for end-to-end path KPI prediction. This is mainly because the relation between path delay and link delays will be additive if the packet flows in a given network topology become stable. Another reason is that the utilization of MsgNet to compute the messages from links to paths instead of feeding the link hidden states directly to the path update MsgNet increases the expressive power of the model.

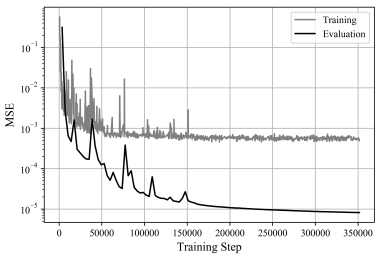
Besides, using GRUs instead of MLPs as MsgNets can greatly solve the exploding gradient problem and thereby help stabilize the training. This can be shown by the fact that the confidence intervals of PLNet-GRU and PLNet-GRU-1 are more than 5 times smaller than those of PLNet-MLP and PLNet-MLP-1. Although we've observed that the best performance of PLNet-MLP models and PLNet-GRU models are very close, PLNet-GRU models have a better guarantee that the models on average perform well. Consequently, GRU is a more favorable choice for MsgNet.

B. Generalization capabilities

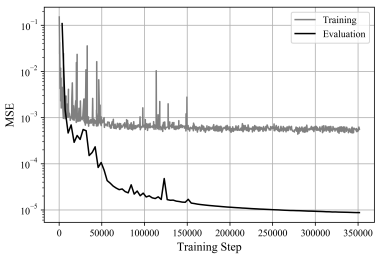
We evaluate the models trained with GNB data-set on NSFNET, GEANT2, and Germany50 topologies to test their generalization capabilities. As it can be seen from Table IV and in Figure 7, the models are able to generalize well to NSFNET and GEANT2 networks, which are similar to GBN. However, all the models fail to generalize to Germany50 that differs a lot from GBN. Our observation is that models that are over-fitted to a particular topology may generalize worse to other topologies and have unstable performance if the topologies are too different from each other. A piece of evidence is that the generalization results are in the approximately opposite order than the test-set results, i.e., PLNet-MLP-1, RouteNet, PLNet-MLP, PLNet-GRU-1, PLNet-GRU,



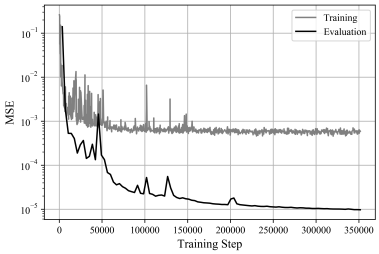
(a) PLNet-GRU



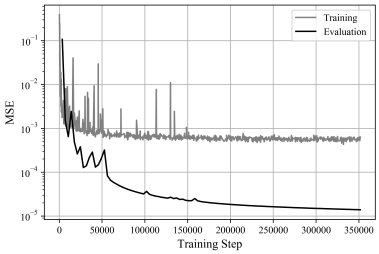
(b) PLNet-MLP



(c) PLNet-GRU-1



(d) PLNet-MLP-1



(e) RouteNet

Fig. 5. Learning curves of PLNet models and RouteNet.

TABLE II
OPTIMAL HYPERPARAMETERS FOR GBN DATA-SET

Hyperparameters	PLNet-GRU	PLNet-MLP	PLNet-GRU-1	PLNet-MLP-1	RouteNet
Dropout rate	0.1	0.1	0.1	0.1	0.1
Learning rate	0.001	0.001	0.001	0.001	0.001
Weight decay	0	0	0	0	0
Dimensionality of path state GRU	64	64	64	64	64
Dimensionality of link state GRU	64	64	64	64	64
Number of neurons in MLP MsgNet	N/A	128	N/A	128	N/A
Output dimension of MLP MsgNet	N/A	32	N/A	32	N/A

TABLE III
MSE LOSS ON THE GBN TEST DATA-SET

	PLNet-GRU	PLNet-MLP	PLNet-GRU-1	PLNet-MLP-1	RouteNet
Loss, $\times 10^{-5}$	1.46 ± 0.05	2.05 ± 0.38	1.67 ± 0.09	2.52 ± 0.48	2.78 ± 0.19

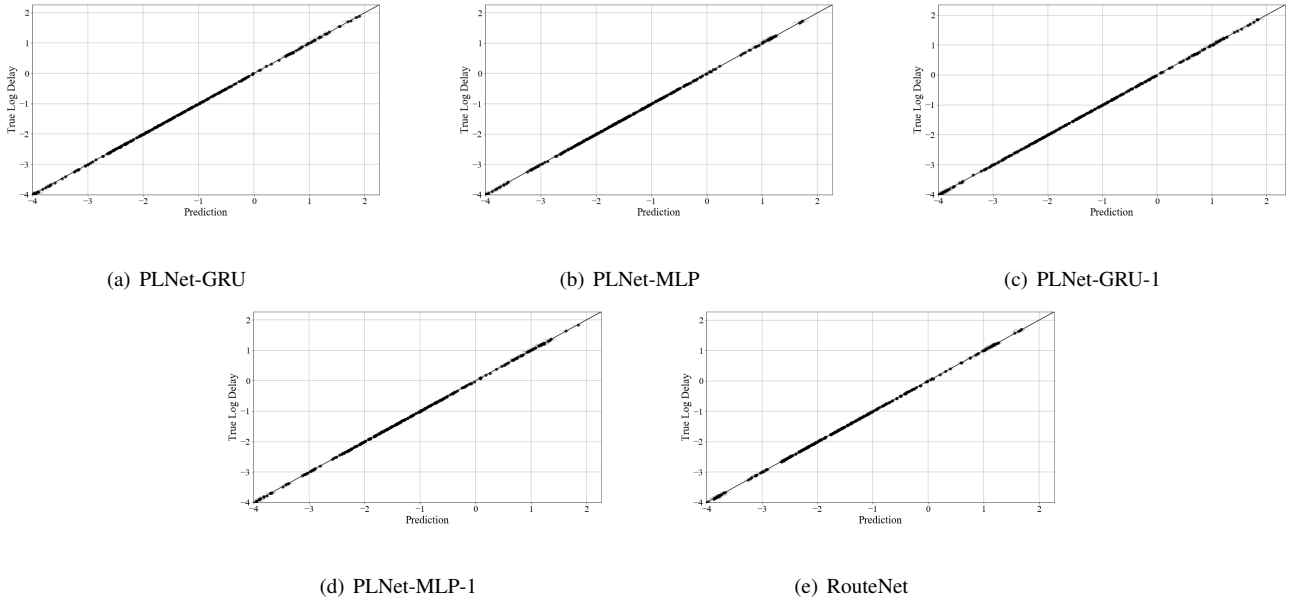
and PLNet-GRU, PLNet-GRU-1, PLNet-MLP, PLNet-MLP-1, RouteNet, respectively. Note that for each model the figures illustrate only one trained model from the total of 10 training runs. Hance, it's more appropriate to refer to Table IV for more accurate comparison as it contains the means of all the runs and the confidence intervals of the performance metrics.

The models that fit best one topology may have problems to generalize to very different scenarios. Thus, none of these models are ready-to-use when network topology changes considerably. It is always better to be careful about hyper-parameter tuning and to take into account the concrete field of application for the model.

C. Comparison of inference speed

We evaluate how long it takes for the models to build the prediction in a simple topology that is easy to scale. It consists of one path with the varying numbers of links from 1 to 50. The comparison is made on Nvidia 2080Ti GPU with 11GB of memory. In Fig. 8, one can observe that the execution time of RouteNet grows linearly with the maximum number of links in paths, while the execution time of PLNet models remains constant.

In a large network, the PLNet models get faster because the computation of messages in RouteNet cannot be parallelized over links in the paths. If path updates of RouteNet are counted sequentially, the effective number of NN layers gets



(a) PLNet-GRU (b) PLNet-MLP (c) PLNet-GRU-1

(d) PLNet-MLP-1 (e) RouteNet

Fig. 6. True log delays against the predicted ones on the GBN test set. 30 samples from the GBN test set are randomly selected for each figure.

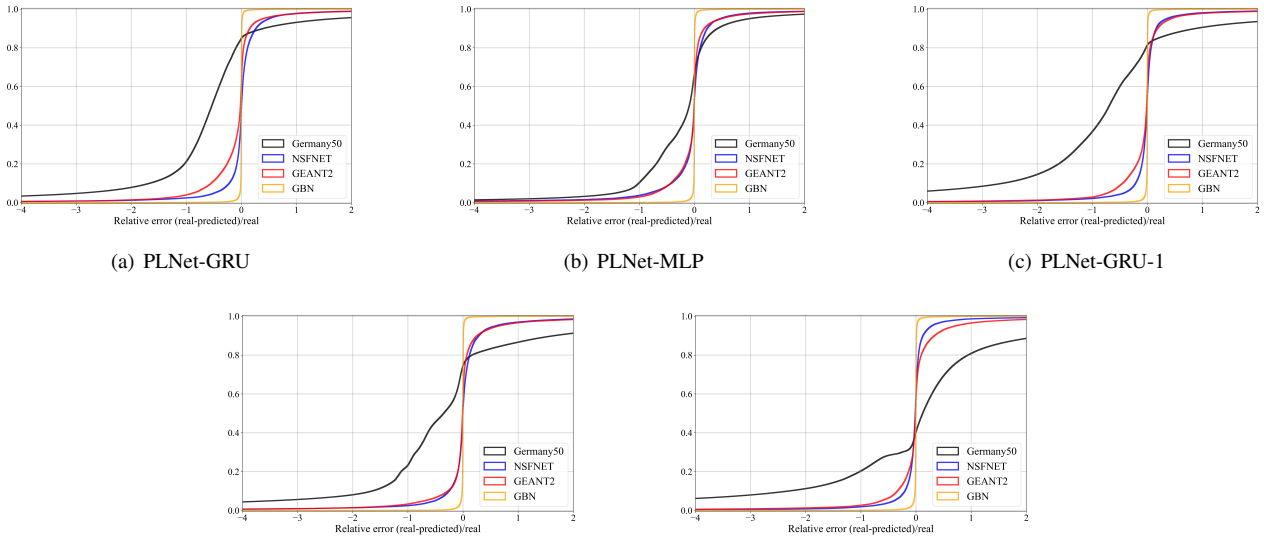
TABLE IV
PERFORMANCE OF MODELS ON UNSEEN TOPOLOGIES

Data-sets	Metrics	PLNet-GRU	PLNet-MLP	PLNet-GRU-1	PLNet-MLP-1	RouteNet
NSFNET	Loss, $\times 10^{-2}$	4.10 ± 2.78	1.76 ± 1.51	2.51±0.85	2.90±0.63	1.31 ± 0.67
	ρ	0.978 ± 0.028	0.990 ± 0.009	0.986±0.005	0.985±0.003	0.993 ± 0.003
	R^2	0.948 ± 0.038	0.977 ± 0.022	0.971±0.010	0.966±0.006	0.985 ± 0.007
GEANT2	Loss, $\times 10^{-2}$	5.92 ± 1.12	2.92 ± 1.19	4.46±1.17	1.38±0.38	2.52 ± 0.56
	ρ	0.982 ± 0.003	0.990 ± 0.004	0.986±0.004	0.995±0.001	0.988 ± 0.002
	R^2	0.949 ± 0.011	0.978 ± 0.009	0.963±0.011	0.990±0.003	0.962 ± 0.05
Germany50	Loss	0.71 ± 0.152	1.67 ± 1.27	1.04±0.43	0.58±0.27	0.78 ± 0.14
	ρ	0.768 ± 0.067	0.683 ± 0.127	0.708±0.093	0.845±0.061	0.828 ± 0.036
	R^2	-0.054 ± 0.212	-0.231 ± 0.696	-0.198±0.304	0.500±0.184	-0.318 ± 0.352

larger when the network topology grows. However, message networks in PLNet models have a fixed number of fully-connected layers for message computations and the computations of the messages can also be done in parallel. Thus, PLNet is less sensitive to the number of links in the paths. No matter how the network size increases, the new design stays less sensitive to the scale of the network. Specifically, as we see in Figure 8, PLNet-GRU-1, and PLNet-MLP-1 are already faster than RouteNet even when the path length is smaller than 5, which is a typical case in practice, whereas PLNet-MLP and PLNet-GRU are only faster than RouteNet after the path length grows larger than 25. Hence, one needs to consider the use case of the model in order to have a better trade-off between accuracy and efficiency. That is to say, if accuracy is the most important, then PLNet-GRU may be preferred, and if efficiency is more crucial than accuracy, PLNet-GRU-1 and RouteNet are both good models, depending on the size of the network topology under consideration.

V. CONCLUSION

In this study, we build a new GNN model that can be used to predict IP networks' performance. We begin with the analysis and a new implementation of RouteNet architecture. Checking and tuning of model parameters and assumptions was performed. An alternative PLNet architecture is considered. Then, we train and evaluate both designs together. The results have shown that all models have good performance on the test-set within the same topology, while PLNet models have slightly better accuracy. However, we find out that the over-tuned models to a particular topology may have worse results on new unseen networks. Hence, it is advisable to be careful about the selection of the training data-set. Besides, we observe faster execution time of prediction and better scalability on PLNet models than RouteNet when the network's path length is getting longer. Because of these properties, PLNet models can enrich the choices of deep learning-based network modeling tools.



(a) PLNet-GRU (b) PLNet-MLP (c) PLNet-GRU-1

(d) PLNet-MLP-1 (e) RouteNet

Fig. 7. Comparison of Cumulative Distribution Function of relative error on different topologies.

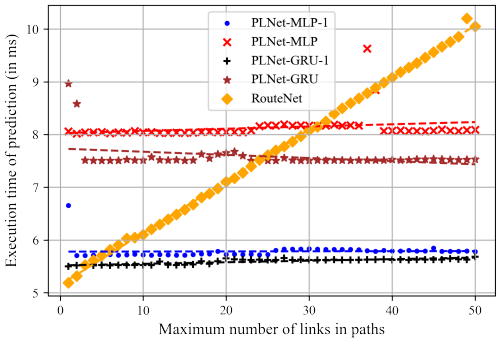


Fig. 8. Inference speed comparison of PLNet-GRU, PLNet-GRU-1, PLNet-MLP, PLNet-MLP-1, and RouteNet: dots - experimental points, lines - linear approximation.

There are several future directions for the continuation of this study. Firstly, the implementation of a similar simulation scenario in a fully open-source ns-3 simulator gives a possibility to verify the paper's results in a new simulation environment. Secondly, RouteNet and PLNet models have good potential for reinforcement learning. Thirdly, we believe the availability of data-sets on real 5G networks would be much richer in the future, making verification of the algorithms on real test-beds possible. Finally, although we consider more generic scenarios than 5G in this paper, it is still a good starting point for going further into more specific 5G scenarios. That is to say, extending the comparison and application of the models on more extensive networks and in the context of 5G scenarios like end-to-end slicing are also promising research topics.

REFERENCES

[1] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on selected areas in communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[2] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[3] V. Räisänen, "A framework for capability provisioning in b5g," in *6G Summit, Levi*. IEEE, 2020.

[4] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[5] Nokia, "Machine learning in 5g networks for enhanced mobile broadband and industry solutions," *White Paper*, 2019.

[6] J. Moysen and L. Giupponi, "From 4g to 5g: Self-organized network management meets machine learning," *Computer Communications*, vol. 129, pp. 248–268, 2018.

[7] H. Holma, A. Toskala, and T. Nakamura, *5G Technology: 3GPP New Radio*. John Wiley & Sons, 2020.

[8] ETSI GS ZSM, "Zero-touch network and Service Management (ZSM); Reference Architecture," Tech. Rep., 2019.

[9] P. Velho, L. Schnorr, H. Casanova, and A. Legrand, "Flow-level network models: have we reached the limits?" 2011.

[10] ns-3 simulator home page. [Online]. Available: https://www.nsnam.org/

[11] Omnet++ simulator home page. [Online]. Available: https://omnetpp.org/

[12] T. S. Buda, H. Assem, L. Xu, D. Raz, U. Margolin, E. Rosensweig, D. R. Lopez, M.-I. Corici, M. Smirnov, R. Mullins *et al.*, "Can machine learning aid in delivering new use cases and scenarios in 5g?" in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 1279–1284.

[13] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction," *IEEE/ACM Transactions on Networking*, vol. 18, no. 4, pp. 1026–1039, 2010.

[14] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar, "Answering what-if deployment and configuration questions with wise,"

in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, 2008, pp. 99–110.

[15] X. Chen, J. Guo, Z. Zhu, R. Proietti, A. Castro, and S. J. B. Yoo, "Deep-rmsa: A deep-reinforcement-learning routing, modulation and spectrum assignment agent for elastic optical networks," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, 2018, pp. 1–3.

[16] S. Xiao, D. He, and Z. Gong, "Deep-q: Traffic-driven qos inference using deep generative network," in *Proceedings of the 2018 Workshop on Network Meets AI & ML*, 2018, pp. 67–73.

[17] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, 2017, pp. 185–191.

[18] R. K. Sinha, R. Pandey, and R. Pattnaik, "Deep learning for computer vision tasks: a review," *arXiv preprint arXiv:1804.03928*, 2018.

[19] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational intelligence magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[20] J. Suárez-Varela, S. Carol-Bosch, K. Rusek, P. Almasan, M. Arias, P. Barlet-Ros, and A. Cabellos-Aparicio, "Challenging the generalization capabilities of graph neural networks for network modeling," in *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, 2019, pp. 114–115.

[21] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[22] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of graph neural networks for network modeling and optimization in sdn," in *Proceedings of the 2019 ACM Symposium on SDN Research*, 2019, pp. 140–151.

[23] P. Almasan, J. Suárez-Varela, A. Badia-Sampera, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: An optical network routing use case," *arXiv preprint arXiv:1910.07421*, 2019.

[24] F. Geyer, "Deepcomnet: Performance evaluation of network topologies using graph-based deep learning," *Performance Evaluation*, vol. 130, pp. 1–16, 2019.

[25] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[26] Routenet demo github repository. [Online]. Available: https://github.com/knowledgedefinednetworking/demo-routenet

[27] A. Paszke, S. Gross *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8026–8037. [Online]. Available: http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[28] Knowledge-defined networking training datasets. [Online]. Available: http://knowledgedefinednetworking.org/

[29] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. [Online]. Available: http://dx.doi.org/10.3115/v1/D14-1179

[30] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," 2017.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

APPENDIX

Algorithm 1 : PLNet model steps

Input: paths traffic demand $\{d_i\}$, links capacity $\{\mathbf{x}_j\}$
Output: delays for all paths $\{y_i\}$

// Initialize path and link states
for each path i **do**
  $p_i^0 \leftarrow [d_i, 0 \ldots, 0]$
end for
for each link j **do**
  $l_j^0 \leftarrow [\mathbf{x}_i, 0 \ldots, 0]$
end for
for $t = 1$ to T **do**
  // Message passing from links to paths
  for each path i **do**
    $m_i^t \leftarrow 0$
    for each link j in path i **do**
      $m_i^t \leftarrow m_i^t + \text{PathMsgNet}(p_i^{t-1}, l_j^{t-1})$
    end for
    $p_i^t \leftarrow \text{PathUpdate}(p_i^{t-1}, m_i^t)$
  end for
  // Message passing from paths to links
  for each link j **do**
    $\mu_j^t \leftarrow 0$
    for each path i to which link j belongs **do**
      $\mu_j^t \leftarrow \mu_j^t + \text{LinkMsgNet}(l_j^{t-1}, p_i^{t-1})$
    end for
    $l_j^t \leftarrow \text{LinkUpdate}(l_j^{t-1}, \mu_j^t)$
  end for
end for
// Readout function
for each path i **do**
  $y_i \leftarrow \text{MLP}(p_i^T)$
end for