Data-Driven Network Performance Prediction for B5G Networks: A Graph Neural Network Approach

Mahnoor Yaqoob
Design Engineering and Mathematics
Middlesex University
London, England
my365@live.mdx.ac.uk

Ramona Trestian
Design Engineering and Mathematics
Middlesex University
London, England
R.Trestian@mdx.ac.uk

Huan X. Nguyen
Design Engineering and Mathematics
Middlesex University
London, England
H.Nguyen@mdx.ac.uk

***Abstract*—Extreme connectivity, dynamic resource provisioning and demand of quality assurance in 5G and Beyond 5G (B5G) networks calls for advance network modeling solutions. We need functional network models that are able to produce accurate prediction of Key Performance Indicators (KPI) such as latency, overall delay, jitter or packet loss at low cost. Graph Neural Networks (GNN) have already shown great potential for network performance prediction, because of their ability to understand the network configurations. In this paper, we focus on improving the generalization capabilities of GNN in relatively complex IP transport network scenarios of future generation networks. We take RouteNet GNN as a reference model and present an alternative GNN. We train both models with relatively smaller network scenarios while for evaluation we use complex and large network configurations. After hyper-parameter tuning for RouteNet and proposed GNN, the results show that our model outperforms baseline architecture in evaluation phase. The validation losses for scenarios not seen during training phase, are significantly lower than the RouteNet.**

***Index Terms*—Deep Learning, GNN, 5G, B5G, Network Modeling**

I. Introduction

OVER the years mobile networks have been going through a steady and gradual evolution from first generation networks towards second, third, fourth, and now fifth generation networks. Ever since early 5G systems have been introduced, the research interest has been shifting towards future generation networks referred as sixth generation 6G networks [1]. With the rapid explosion of smart technologies and applications like holographic projection, virtual and augmented reality as well as mission critical applications like remote surgery, current networks and 5G networks will no longer be able to meet these expectations. The introduction of key technologies like Software Defined Networking (SDN) and Network Function Virtualization (NFV) [2] in 5G networks allowed network service providers to manipulate these individual slices without disrupting the existing services. However, this led to multi-vendor software requiring expert knowledge to control and configure network manually.

Therefore, it has become essential for Beyond 5G (B5G) networks to assure Service Level Agreement (SLA) and dynamic and autonomous traffic management. Although there has been significant work done on optimization and scheduling of radio and network resources, service-based resource allocation remains crucial for B5G networks in order to meet the ever-increasing user demand and expectation of quality of service and quality of experience.

Numerous analytical models have been proposed for network modelling over the decades. But in order to ensure model tractability in complex network these models rely on simplifying assumptions about properties of underlying architectures (e.g., traffic with poisson distribution and static routing) which sacrifice the accuracy. Conversely, packet level simulators have proven to be very accurate, but they have high computational costs making simulation modeling unfeasible for complex network scenarios of future generation networks. To this extent Deep Learning (DL) technology has a great potential to meet the expectations of B5G networks because of its capability of retrieving insightful knowledge in a data driven manner [3]. To achieve automation Machine Learning (ML) and Artificial Intelligence (AI) driven solutions are fundamental in almost every sector. The dynamic, diverse and exponentially increasing connectivity requirements in 5G and B5G networks demands incorporation of scalable ML models to accomplish network automation. Anomaly detection, resource level prediction, proactive policy generation and mitigation are some of the applications envisioned when it comes to applying ML based solutions for networks.

Moreover, the traffic explosion gives rise to many problems but at the same time it provides a larger data base for learning and training ML algorithms that will accelerate the network performance of B5G networks. Hence, leveraging ML algorithms [1], [3], such as Graph Neural Networks (GNN) for complex network structures, where the topologies are in form of complex graphs can significantly improve the performance of networks and help predict future failures as well as optimize the performance of network like latency, throughput, bandwidth allocation and resource allocation.

Network modeling enables evaluation of the subsequent performance of what-if scenarios without necessarily changing the state of data plane. Moreover, network modeling is profitable for network control and management applications of B5G such as planning, efficient recovery in case of failures and optimization. Cutting edge solutions in network modeling of end-to-end network performance can act as basis for more network automation and optimization solutions for B5G with

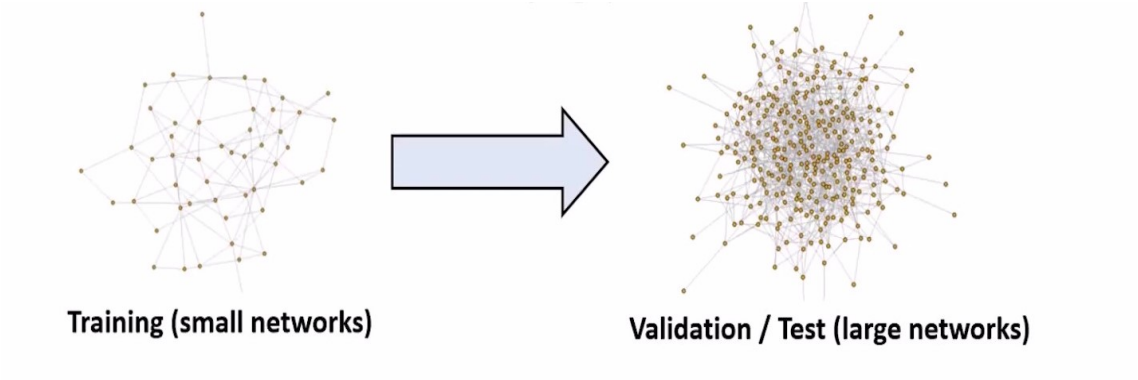


Fig. 1: Network topologies in training vs validation sets

minimum computational costs.

In this paper we mainly focus on analyzing the generalization capabilities of GNN for accurate network performance indicator predictions in case of complex network scenarios where topologies are changing constantly. GNN perform well in case of fixed topologies. However, in case of dynamically changing topologies and complex network scenarios the ability of GNN to generalize suffers significantly. Thus, we need to develop scalable GNNs that are able to predict accurate KPIs in case of unseen scenarios. As a starting point we implement recently proposed RouteNet model using TensorFlow and Keras python libraries. Furthermore, we present an extended version of RouteNet to improve the accuracy. We evaluate both models and found that proposed algorithm performs better on validation sets in comparison to RouteNet. Although we focus on IP network scenario only in this study, we believe that the study's findings on generalization capabilities of GNN can be applied to broader scope for end-to-end network orchestration and network slicing.

Rest of the paper is organized as follows: in section II related works and recent efforts of network modelling using GNNs are discussed. In section III we describe the working of RouteNet Algorithm for training network data and we propose updated version of RouteNet. In section IV we describe the experimental setup, dataset and state the results while section V concludes the paper.

II. Related Works

Owing to its recent success in many domains, deep learning is gaining popularity amongst researchers for optimization and predictions of network performance KPIs as well as for autonomous network management. Many researchers have shifted their attention to exploring the uses of DL for wireless network, for instance authors in, [4] have systematically reviewed popular DL approaches to address wireless network optimization problems. Most commonly used ML approaches identified are Deep Neural Networks (DNN), GNN, Graph Convolutional Networks (GCN) and Deep Reinforcement learning (DRL) [5].

In this paper we merely focus on applications of GNN for network modeling and optimization of performance indicators. GNN is a type of artificial neural network architecture which is specifically designed for data structured as graphs. In a graph structured data, there are set of graphs with set of nodes and edges, where a feature vector is associated to each graph. The ability of GNN to retain basic topological relations between nodes, also known as isomorphism, makes it a perfect candidate to be used for various topologies. Most of GNN architectures are special cases of Message Passing Neural Network (MPNN) [6]. MPNN are based on the assumption that nodes, edges and the whole graphs can be encoded using feature vectors also referred as embeddings. In forward propagation of MPNN there are three functions: *message*, *update* and *readout*. Node and link embeddings are inputs to message function, while the information vector or message to the neighboring nodes in a graph is the output. All the messages from neighboring nodes in the graph are collected by update function and the embeddings are constantly updated. Message passing can be repeated several times in neural network before readout function can take all the resulting embedding of nodes and edges to produce final output.

GNNs have been gaining popularity in the area of wireless networks because of their ability to learn from various network topologies and to make generalizations in unseen scenarios. In this context Rkhami et al. [7] used GCN and DRL to tackle Virtual Network Embedding (VNE) problem. They have formalized VNE as Markov Decision Process and in order to extract the features GCNs are used and different placement strategies have been introduced for placement of virtual networks and virtual links. However, the network topology used this case includes 24 nodes and 37 links only. Wang et al. [8] proposed a network digital twin model that is based on GNN for efficient management of network slicing and to predict the E2E latency of individual network slices. They have evaluated the generalization capabilities of proposed GNN solution by minimizing the Log-Cosh loss and results were within 95 % confidence interval. Maximum number of nodes in the network topology were 50 with 276 links. Moreover, Zhang et al. [9] proposed a topology aware deep learning framework for network flow optimization for a multihop wireless network. Using message passing network approach which is a form

of GCN they have iteratively update the node and edges embeddings. They used dynamic topologies with 10, 30 and 50 nodes to efficiently prune the critical links using GNN, thus maximizing the minimum commodity flow in the network.

Rusek et al. [10], [11] proposed a novel GNN architecture RouteNet to accurately predict the network performance indicators such as jitter, delay and packet loss. They used SDN scenario where each node is modelled as routers in IP network. RouteNet is gaining popularity because of its ability to accurately predict the network statistics even in scenarios that were not seen while training. Similar to previous studies in [12] authors proposed a path-link GNN (PL-NET) to predict the overall delay for IP network. They modelled IP network as a bipartite graph since according to the authors, the sequence of links in the paths of graph have very little impact on performance of data flow. They compared the accuracy of the proposed approach with baseline RouteNet Model and results were similar to baseline architecture. Moreover, in [13] GNN and LSTM approach was used to propose an intent based solution for automatic network orchestration.

All the above-mentioned studies use network topologies with smaller networks during training and for evaluation similar network scenarios are tested. However, future generation networks will not have static topologies and network scenarios are expected to be far more complex. Therefore, in this paper we unveil the potential of GNNs to generalize to more complex networks shown in Fig. 1, not seen during the training process. We use network topologies with nodes ranging from 55 to 300 for evaluation phase and to the best of our knowledge such distributions have not been used before in previous studies. The aim is to analyze the generalization capabilities of GNN for scenarios that are far more complex then those scene during training phase. It is of utmost importance for B5G networks to have scalable and robust ML solutions that can cope with ever changing network requirements. Therefore, it is crucial for network models to accurately predict KPIs and understand the network configurations.

III. MODELS FOR NETWORK PERFORMANCE PREDICTION

An IP network scenario in B5G can be represented by a graph G_S with set of V_S physical nodes and E_S set of links connecting the nodes and set of paths P_S. Each path has one or more specific ordered links and they are characterized by routing schemes. Each link has features like bandwidth and each path has features such as packets transmitted, traffic demand etc. An example network topology with four nodes and two paths can be seen in Fig. 2. The input feature vectors of links for each state i can be denoted by x_{li} and for each path state it can be denoted by x_{pi} while current state (hidden states in GNN) of links and paths can be denoted by h_{li} and h_{pi} respectively.

A. RouteNet Model

Routenet model [10] is designed to predict the KPIs from IP network configurations. In order to predict over all delay, jitter, and packet loss, RouteNet model builds computational graphs having T message passing iterations in GNN. Every iteration has two steps and messages are updated in each step:

1) Updates of path states denoted by p_i^t where t is message passing iteration and i is state index.
2) Update of link states denoted by l_i^t where t is message passing iteration and i is state index.
3) Message from path updates and link updates denoted by $m_{i\to j}^t$ where t is message passing iteration and $i \to j$ indicates the messages from path i to link j, which are aggregated to update the link states.

Gated Recurrent Units (GRU) are used for inferring these path and link states. In order to update the path states GRU relies on inputs of current states of the links of the path (Fig. 3 top half). For updates of link states GRU takes sum of the messages from all paths of current link (Fig. 3 bottom half). These message passing iterations are carried out for T repetitions before final readout unit can predict the output of path KPIs (delay, jitter and packet loss). RouteNet takes the order of links in a particular path into consideration, in order to process the updates of path states.

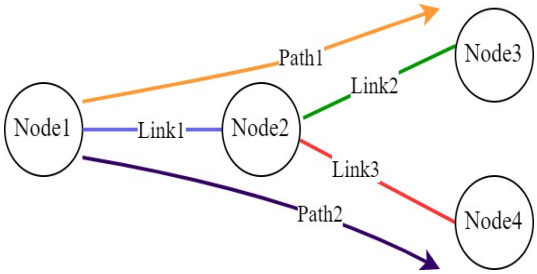


Fig. 2: An example of a network topology with two paths.

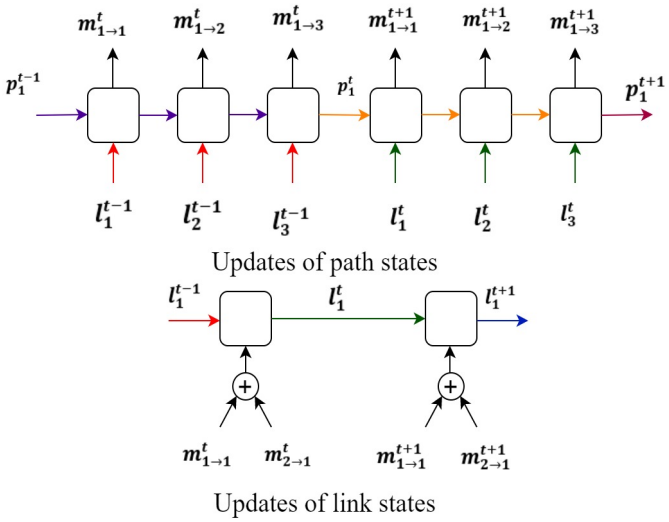


Fig. 3: Representation of Computational graph for network topology in RouteNet.

B. Updates to the RouteNet

Since RouteNet is not designed to model networks with considerably larger graphs than those seen during training process, the accuracy to generalize suffers significantly. Hence, we apply some updates to the existing algorithm to improve the performance and decrease the loss function values. Firstly, we apply some tuning to hyperparameters used in baseline RouteNet. We started with increasing the number of hidden units to 64 and 128 as well as we have used 12 message passing iterations and 12 readout units. Adam optimizer [14] is used with the initial learning rate of 0.001.

In the baseline only traffic matrix is used as a feature to predict the per-path delays, conversely, we used traffic and packets as input features. We have also added batch normalization layers and for activation function we have used RELU6, which is a modification of traditional rectified linear units, where we limit the activation to 6. RELU6 is chosen instead of RELU to solve the exploding gradient problem. We have witnessed the Mean Absolute Percentage Error (MAPE) reducing significantly after these updates to the baseline. Since MAPE is computed using (1) (the reason for evaluation loss score being really high) where n is the number of samples while y'_i is the predicted delay and y_i is the true delay value.

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y'_i - y_i}{y_i} \right| \tag{1}$$

For evaluation in our implementation, we use the Mean Squared Error (MSE) score shown in (2), to computed over the delay predictions.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - y'_i)^2 \tag{2}$$

The details of the updated model are listed in Algorithm 1. Lines 1 to 3 represent initialization of feature vectors for path and link states. The loop from lines 5 to 12 represent the message passing between the encoded hidden state among links and paths. Likewise, line 13 to 16 are update functions that encode the new updates into hidden states of paths and links. Lastly, line 17 is a readout function which predicts the over all per path delays.

Algorithm 1 Updates to Message Passing RouteNet

Require: packets and traffic load for paths, capacity for links
Ensure: overall path delays ▷ Initializing features

1: **for each** path **do** $[x_{pi}, 0, 0, .., 0]$;
2: **end for**
3: **for each** link **do** $[x_{li}, 0, 0, .., 0]$;
4: **end for**
5: **for** $t = 0$ to $T - 1$ **do** ▷ Message Passing Function with RELU6 and BatchNormalization
6: **for each** path **do**
7: **for each** link **do**
8: $h_p^t \leftarrow GRU_t(h_p, h_l)$
9: $m_{p,l}^{t+1} \leftarrow h_p^t$
10: **end for**
11: $h_p^{t+1} \leftarrow h_p^t$
12: **end for**
13: **for each** link **do** ▷ Update Function
14: $h_l^{t+1} \leftarrow U_t(h_l^t, \sum_{p:k \in p} m_{p,k}^{t+1})$
15: **end for**
16: **end for** ▷ Readout Function
17: $y'_p \leftarrow F_p(h_p)$

IV. Results and Discussions

A. Experimental Setup

For training and evaluation, we use open-source Knowledge defined networking dataset, which is produced by packet level simulator OMNET++ [15]. The dataset contains simulation results of delay, jitter and packet loss for different network topologies containing 20, 25, 30, 35, 40, 45 and 50 nodes for training set and validation and test sets include samples of networks considerably larger, containing nodes ranging from 50 to 300. All these networks with varying topologies have been generated artificially using the Power-Law Out-Degree algorithm [16] where the parameter have been extrapolated from real-world topologies acquired from repository of Internet Topology Zoo [17].

The IP network scenario considered in dataset has a form of undirected graph with nodes, links, adjacency matrix, routing configuration, queue statistics and port status.

Each node in the dataset represents routers and links joining the routers with queueing policies configured for each node in a SDN scenario. Nodes can send packets to any other node. The routing table is also provided for each node, routes are calculated based on Shortest Path First (SPF) algorithm. For routing configuration 100 different routing schemes have been defined for each topology and each sample also contains multiple simulation runs providing ample data to train and eliminating any dependencies on routing algorithm. The statistics for Knowledge defined dataset are presented in Table I. The data set is divided into three different sets for training, validation and test. The training set consists of 120000 samples with varying topologies ranging from 25 nodes to 50 nodes, where each sample is composed of input files of network topology, traffic matrix, routing scheme and simulation results and one such sample has results of 25 simulation runs. Similarly, validation set has 3120 samples of similar configuration but network sizes are larger in comparison to training sets, that is nodes ranging from 50 to 300. Moreover, test set has 1560 samples of similar configuration and network sizes also follow same distribution as validation sets.

TABLE I: Statistics for training and validation datasets

Dataset	# Nodes in topology	# Max links	# samples
Training set	25-50	280	120000
Validation set	50-300	1464	3120
Test set	50-300	1168	1560

TABLE II: Optimal hyperparameters for training and evaluation

Hypperparameters	RouteNet	Updated Algorithm
Learning rate	0.001	0.005
Hidden units for path state	32	64
Hidden units for link state	16	64
Message passing iterations	8	12
Number of readout units	8	12

B. Comparison of prediction accuracy

We first evaluate RouteNet in Tensorflow and NetworkX using Keras API, and use it as a baseline for comparison. We use mini batches of size 100 and the training with one of such batches forms a step. After every 4000 steps we save a checkpoint of the model. Evaluation is performed of validation and test sets. When model stops improving the training process is stopped; in our case after 400,000 steps. We first use MAPE loss function to see the percentage error. As shown in Fig. 4 the baseline RouteNet has an error percentage of more than 300%. After performing some hyperparameter tuning the validation loss has dropped to 100%, which is quite an improvement. As seen in Fig. 4 training RouteNet with more hidden units and readout units gives better performance on validation sets but after a while the loss function stabilizes and it stops improving. We also found that model with 32 hidden units and 64 hidden units have almost similar results as shown in Fig. 4a and 4b, so after a while training a bigger model does not necessarily improve the performance.

In order to further improve the validation loss scores we use, MSE loss function and include batch normalization layers to the proposed Updated GNN. We use learning rate 0.005 and RELU6 activation function. In addition to adding normalization layers, we also include packets information as an input feature vector along with traffic matrix to predict per path delays. All the hyperparameters are carefully chosen from relevant literature [8], [9], [12] and the optimal hyperparameters used for training and evaluation are listed in Table II. As seen in Fig. 5a and 5b the training loss is significantly improved after adding more input features. The validation loss is still quite high as expected, this is because validation set includes network topologies that are unseen during training. Moreover, larger network sizes mean smaller paths, which is in contrast to smaller networks. However, we have witnessed significant improvement in validation loss by including more input features to the model and by increasing the size of the model. We have also seen that training and validation losses are lower in case of 32 hidden units for updated version of RouteNet as shown in Fig. 5.

It is worth mentioning that there is a clear tradeoff between cost and prediction accuracy. In our efforts to improve the validation losses and achieving better accuracy we increase the hidden state and size of the GNN which in turn resulted in increased compute time and compute resources. As a future work we intend to combine analytical solution approaches with our algorithm in order to predict the KPIs so that we can achieve better results while minimizing the computational cost.

The results reveal the potential of GNN to generalize in different network scenarios, which are relatively more complex than those seen while training process. In future it is possible to even apply transfer learning by fine tuning readout function only with new network configurations while reusing computationally expensive message passing function. We need functional network models that are able to produce accurate prediction of KPI such as latency, overall delay, jitter or packet loss at low cost. Because network optimization solutions solely rely on performance metrics generated by network models, thus, in order to optimize performance metrics accurate prediction of network performance indicators is essential. Flexible models that can accurately predict performance and understand the relation between network configuration and KPI are essential for optimization solution that are a key to enhance performance for B5G service classes like enhanced Ultra Reliable an Low Latency Communications (URLLC) where latency constraints are very strict. However, the functional models to understand the relation between performance indicators and network state metrics are still lacking and can be a future want. This study is a modest attempt towards achieving the goal of unveiling the potential of GNN to retain the information between network metrics and performance measure to predict network KPIs.

V. Conclusions

In this paper we have presented an updated version of RouteNet algorithm that can predict IP networks' performance in terms of overall path delays. We have shown potential of GNN to learn from network topologies and to retain information about traffic matrix and different routing schemes. We begin with implementation of RouteNet in tensorflow and later we have updated the baseline model. We have also performed tuning of hyperparameters. We have trained and evaluated both models, the results have shown that both models perform well on training sets while performance degrades with increasing the network size. However, the updated algorithm performs significantly better on validation sets in comparison with baseline RouteNet. As a future work we intend to incorporate more feature in the proposed model, as well as use techniques such as data augmentation to further improve the generalization capabilities of GNN.

References

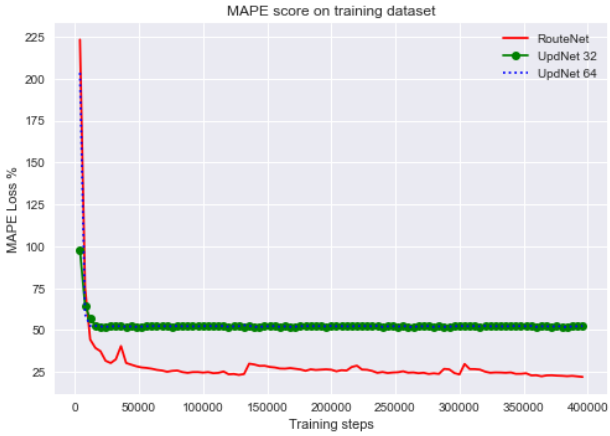
[1] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, GK. Karagiannidis and P. Fan,"6G Wireless Networks: Vision, Requirements, Architecture, and Key Technologies", in *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 28-41, Sept. 2019.

[2] S. Wijethilaka and M. Liyanage, "Survey on Network Slicing for Internet of Things Realization in 5G Networks," in *IEEE Communications Surveys Tutorials*, vol. 23, no. 2, pp. 957-994, June. 2021.

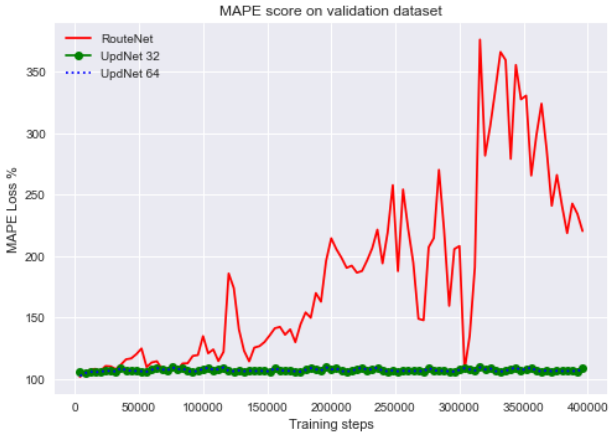
[3] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao and K. Wu, "Artificial-Intelligence-Enabled Intelligent 6G Networks," in *IEEE Network*, vol. 34, no. 6, pp. 272-280, Dec. 2020.

[4] Y. Cheng, B. Yin and S. Zhang, "Deep Learning for Wireless Networking: The Next Frontier," in *IEEE Wireless Communications*, vol. 28, no. 6, pp. 176-183, Dec. 2021.

[5] H. Wang, Y. Wu, G. Min, J. Xu and P. Tang, " Data-driven Dynamic Resource Scheduling for Network Slicing: A Deep Reinforcement Learning Approach" in *Information Sciences*, 2019.

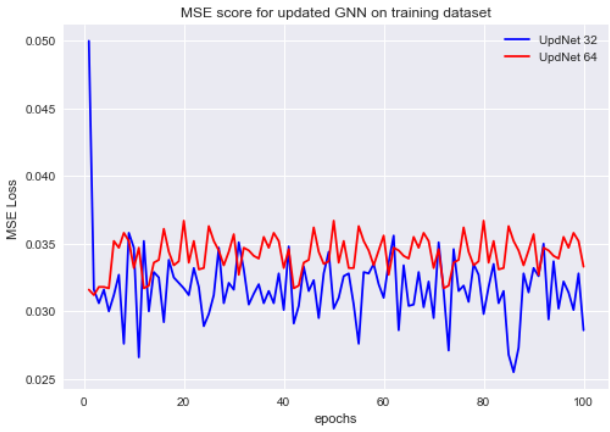


(a) Training loss

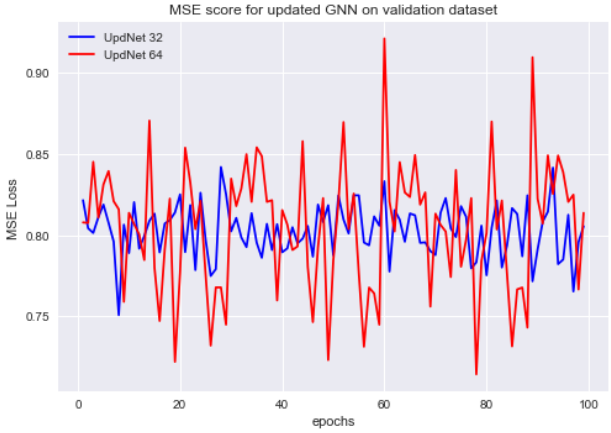


(b) Validation loss

Fig. 4: Comparison of learning curves of RouteNet and Updated GNN with MAPE



(a) Training loss



(b) Validation loss

Fig. 5: Learning curves of Updated RouteNet with MSE loss

[6] J. Gilmer, SS. Schoenholz, PF. Riley, O. Vinyals, and GE. Dahl. "Neural message passing for quantum chemistry." In *International conference on machine learning*, pp. 1263-1272. PMLR, 2017.

[7] A. Rkhami, T. A. Quang Pham, Y. Hadjadj-Aoul, A. Outtagarts and G. Rubino, "On the Use of Graph Neural Networks for Virtual Network Embedding," *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1-6, 2020.

[8] H. Wang, Y. Wu, G. Min and W. Miao, "A Graph Neural Network-Based Digital Twin for Network Slicing Management," in *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1367-1376, Feb. 2022.

[9] S. Zhang, B. Yin and Y. Cheng. "Topology aware deep learning for wireless network optimization." *arXiv preprint arXiv*:1912.08336, 2019.

[10] 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*, pp. 140-151. 2019.

[11] A. Badia-Sampera, J. Suárez-Varela, P. Almasan, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio. "Towards more realistic network models based on Graph Neural Networks." *In Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies*, pp. 14-16. 2019.

[12] Y. Kong, D. Petrov, V. Räisänen, and A. Ilin. "Path-Link Graph Neural Network for IP Network Performance Prediction." *In 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 170-177. IEEE, 2021.

[13] TA. Khan, K. Abbas, JJD. Rivera, A. Muhammad, and WC. Song. "Applying RouteNet and LSTM to Achieve Network Automation: An Intent-based Networking Approach." *In 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 254-257. IEEE, 2021.

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

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

[16] C. R. Palmer and J. G. Steffan, "Generating network topologies that obey power laws," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 1, 2000, pp. 434–438.

[17] S. Knight, H. Nguyen et al., "The internet topology zoo," *IEEE JSAC*, vol. 29, no. 9, pp. 1765 –1775, 2011.