Exploring the Limitations of Current Graph Neural Networks for Network Modeling

Martin Happ*†, Jia Lei Du*, Matthias Herlich*, Christian Maier*, Peter Dorfinger* and José Suárez-Varela‡
*Intelligent Connectivity, Salzburg Research, Austria
†IDA Lab, University of Salzburg, Austria
Email: {martin.happ, jia.du, matthias.herlich, christian.maier, peter.dorfinger}@salzburgresearch.at
‡Barcelona Neural Networking Center, Universitat Politècnica de Catalunya, Spain
Email: jose.suarez-varela@upc.edu

Abstract—**Graph neural networks (GNN) have recently been proposed as a technique for accurate and cost-efficient network modeling. As an example, the GNN-based model RouteNet has shown potential for network performance evaluation, being the first-of-its-kind Machine-Learning-based model with generalization capabilities to other networks and configurations unseen during training.**

In this paper we assess the generalization limits of RouteNet, by analyzing how different network parameters affect the accuracy of this model. To this end, we systematically evaluate the accuracy of RouteNet under modifications of properties of the network and the traffic, such as the topology size, link capacities, the packet size distribution, and the network congestion level. We determine that, while this GNN model is robust to changes in the structure of its input graph, the quality of the estimates degrades considerably, when the distributions of the predicted values of the evaluation data differ from the training (e.g., end-to-end delays). As a result, we argue that to achieve practical GNN-based solutions for network modeling, new methods are needed that can, for example, cope with traffic loads and network sizes that are significantly different than those seen during training.

I. Introduction

Nowadays, more and more applications are emerging with strict Quality-of-Service (QoS) requirements, such as bounded latency, jitter, or high reliability. In this context, predicting key performance indicators at a fine-grained granularity enables to satisfy many relevant QoS-aware networking use cases. For example, what-if analysis and automatic network optimization can be achieved by predicting the performance of alternative configurations (e.g., new routing schemes) without actually applying them in the real network infrastructure, thus avoiding possible misconfigurations that may break the correct operation of the network [1]. To this end, it is essential to have accurate network models that can predict performance (e.g., delay, jitter, loss) for a network scenario and configuration (Fig. 1).

To predict the performance in networks, network engineers and researchers can rely on packet-level network simulators, for example, OMNeT++[1] [2] and NS-3[2] [3]. However, simulations are typically time-consuming because each packet in the network is simulated; and do not scale to large real-world scenarios. This is especially important in use cases

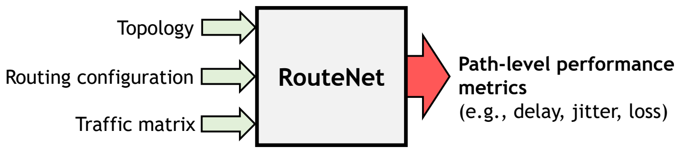

Fig. 1: Black-box representation of RouteNet [1].

where: (1) the simulator is used for online optimization (e.g. to dynamically change routes in Software-Defined Networks), or (2) many simulations are needed to explore a wide range of configurations. Alternatively, analytical models such as queuing theory [4] or network calculus [5] offer more lightweight and scalable executions. However, these models do not perform well when predicting fine-grained performance metrics in real-world scenarios with complex configurations and realistic traffic patterns [6].

In this context, during the last few years the networking community is experimenting with Machine Learning (ML) techniques as effective tools for achieving accurate network models with limited execution cost. This new generation of ML-based network models is often referred to as Digital Twin Networks [7]. In this paper, we take as a reference an improved variant of RouteNet [1], a model based on Graph Neural Networks that represents the state-of-the-art in ML-based network modeling. Particularly, we focus on finding the generalization limits of this model for network performance evaluation. To this end, we perform an extensive evaluation of the model under diverse network scenarios, including different topologies, traffic intensities, and queue scheduling configurations. This allowed us to describe how the model generalizes to these different parameters of the network.

The remainder of this paper is as follows. In Section II we provide background information on the neural networks we use (Graph Neural Networks and, more specifically, RouteNet). Section III presents an overview of existing ML-based solutions for network modeling. In Section IV we evaluate the current limits of RouteNet for the prediction of communication network performance. We present the main findings of our evaluation in Section V and provide an outlook in Section VI.

[1]https://omnetpp.org
[2]https://www.nsnam.org

II. Background

This section provides some background information on Graph Neural Networks (GNN) and RouteNet [1] necessary to understand our evaluation. Readers familiar with the topics can skip this section.

A. Graph Neural Networks

GNNs are machine learning models, which are made to process data in the form of graphs appropriately. Here, a graph is a mathematical object, consisting of nodes and (directed or undirected) edges between them. The problem of determining quantities $y_v \in \mathbb{R}$ which are assigned to individual nodes v often arises. For this paper only this case is relevant. The task of a *generic* GNN can then be subdivided into two subtasks: First, a *hidden state vector* h_v needs to be computed for each node v. This vector h_v lies in some chosen state space $\mathbb{R}^m$ of dimension m and consists of information on the state of v at some level of granularity. It is computed by a *message passing* scheme: After an initialization of h_v with node-level features related to v, each node sends its state to all of its neighbors. Hence, each node v receives a certain number of states $h_{v_1}, \dots, h_{v_k}$, where k is the number of neighbors of v. These states, together with the state h_v of v itself, are converted into an *aggregated message* m_v:

$$m_v = \sum_{i=1}^{k} M(h_v, h_{v_i}) \tag{1}$$

by a *message function* $M : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^n$. The dimension n of the codomain of M is again a chosen value. Using this aggregated message, an *update function* $U : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ computes a new hidden state $U(h_v, m_v)$ for each node v. This message passing is repeated T times until the states of all nodes have (approximately) reached stationary values. Then, secondly, a *readout function* $R : \mathbb{R}^m \to \mathbb{R}$ computes y_v for each node v by applying R to h_v.

A generic GNN thus essentially consists of three functions: M, U and R. The mapping rules of these functions are given by the application of certain independent neural networks (like feed forward or recurrent neural networks). This justifies the name GNN and allows the execution of a training process: The internal parameters of the neural networks are updated via supervised learning in order to compute the target quantities y_v accurately. To achieve this, instances of graphs together with the target quantities y_v for all nodes have to be provided. Note that the way in which such a generic GNN is modeled enables an application to graphs of different sizes and structures (both during training and predicting).

Further details on this generic GNN architecture can be found in literature [8], [9] and in the references provided there. RouteNet deviates from the architecture of standard GNN models. Particularly, it considers heterogeneous graphs as input (i.e. with various types of nodes), and builds a double message-passing phase to exchange the information between the different element types.

In the next subsection we recall this architecture.

```
Input: $\mathcal{L}$, $\mathcal{P}$, $\mathbf{x}_p$, $\mathbf{x}_l$
Output: $\hat{\mathbf{y}}_p$
// Initialize states of paths and
   links
1  foreach $p \in \mathcal{P}$ do $\mathbf{h}_p^0 \leftarrow [\mathbf{x}_p, 0 \dots, 0]$;
2  foreach $l \in \mathcal{L}$ do $\mathbf{h}_l^0 \leftarrow [\mathbf{x}_l, 0 \dots, 0]$;
// Message-passing phase
3  for $t = 0$ to $T-1$ do
       // links to paths
4      foreach $p \in \mathcal{P}$ do
5          foreach $l \in p$ do
6              $\mathbf{h}_p^t \leftarrow \mathrm{RNN}(\mathbf{h}_p^t, \mathbf{h}_l^t)$
7              $\tilde{\mathbf{m}}_{p,l}^{t+1} \leftarrow \mathbf{h}_p^t$
8          end
9          $\mathbf{h}_p^{t+1} \leftarrow \mathbf{h}_p^t$
10     end
       // paths to links
11     foreach $l \in \mathcal{L}$ do
12         $\mathbf{m}_l^{t+1} \leftarrow \sum_{k:p \in l} \tilde{\mathbf{m}}_{k,l}^{t+1}$
13         $\mathbf{h}_l^{t+1} \leftarrow U\left(\mathbf{h}_l^t, \mathbf{m}_l^{t+1}\right)$
14     end
15 end
// Readout phase
16 $\hat{y}_p \leftarrow R_p(\mathbf{h}_p^{T-1})$
```

Algorithm 1: General internal architecture of the original RouteNet [1], as well as the model variant in [10].

B. RouteNet

RouteNet [1] is a GNN-based architecture specifically designed for network performance evaluation. Particularly, this model is intended to predict path-level performance metrics given as input a network state snapshot, defined by: (1) a network topology, (2) a routing configuration, and (3) an end-to-end traffic matrix (Fig. 1). To this end, RouteNet considers a heterogeneous graph as input, where nodes can be *paths* or *links*. This enables to describe the input network state information – i.e., topology, routing, traffic matrix – in a graph-structured manner, which can then be efficiently processed and exploited by the internal GNN architecture of this model.

More in detail, Algorithm 1 shows the internal architecture of RouteNet. In this model, each link of the input network topology ($l \in \mathcal{L}$) and each path resulting from the input routing configuration ($p \in \mathcal{P}$) is represented by a hidden state (that is, h_l or h_p), which is a vector that encodes information about the corresponding element. The input topology and routing configurations are explicitly described as relationships between the *links* ($\mathcal{L}$) and *paths* ($\mathcal{P}$) in the input graph of the model. Thus, each source-destination path $p \in \mathcal{P}$ is related with the sequence of links that form that path (i.e., $p = [l_1, l_2, ..., l_k]$). At the beginning of the execution, the hidden states of links and paths are initialized with some features x_l, x_p (lines 1-2 in Algorithm 1). Then, the model executes a message-passing phase (lines 3-15), where first the state of links are combined with the state of their related paths (lines 4-10), and then the states of paths are shared with their related links (lines 11-14). After T message-passing iterations, the model takes the final path hidden states (h_p^{T-1}) – which are expected to encode relevant information of their corresponding elements – and executes a readout function (R_p) that produces the final path-level outputs ($\hat{\mathbf{y}}_p$). This architecture combines internally a series of neural networks (RNN, U, R_p) that are dynamically assembled according to the elements and connections of the input graph – that is, the set of links, paths, and their relationships. Particularly, the neural networks RNN and U act as universal approximators that learn how to combine and update the hidden states of paths (h_p) and links (h_l) during the message-passing phase, and finally the neural network R_p learns how to translate the hidden states of paths to the final outputs of the model (e.g., path-level mean delays). A more detailed description is in [1].

The authors of [1] showed that RouteNet is able to learn the relationships between the state of the different network elements involved (paths and links) and how they affect the final path-level performance predictions of the model (for example, delay, jitter, loss). More importantly, they show that this GNN model is able to accurately generalize to samples from other networks, in contrast to previous ML-based techniques with limited generalization power (e.g., feed-forward neural networks, autoencoders). In this paper, we focus on evaluating in more detail to what extent this GNN model is able to generalize to different input parameters, by making extensive evaluations varying different key network-related features.

However, note that the evaluation in this paper is not based on the original RouteNet [1], but on an improved variant that is also able to handle various queue scheduling policies and was developed for the Graph Neural Networking challenge 2020 [11], [12]. The corresponding paper [10] provides details on the architecture and training process. The main modifications can be summarized as follows: Change of loss function to mean absolute percentage error from mean squared error, adding more inputs to the initial state (for example, bandwidth and scheduling policy for links as well as average data rate and packet sizes for paths), increase path and link state size as well as the number of neurons per layer in the readout function to 128, and upgrade the internal Gated Recurrent Units (GRUs) of the model to stacked GRUs.

III. Related Work

The paper [1] provides an evaluation of the performance of RouteNet in communication networks that were not used during training and hints to the limits of this approach, but does not provide a comprehensive evaluation on the generalization capabilities. In particular, it does not investigate the performance when the distribution of the validation data differs from the distribution of the training data. Happ et al. [10] extend RouteNet to deal with heterogeneous queue scheduling policies, but do not examine other generalizations. The same holds for the approach of Ferriol et al. [13]: They provide a GNN architecture with states for links, paths and queues. An extension of RouteNet to support different features on forwarding devices was evaluated by Badia et al. [14]. The goal of the Graph Neural Networking Challenge 2021 [15] was to improve the generalization capability of RouteNet to larger communication networks. Our work underscores the need for further work on this problem.

For a general overview on the application of machine learning to communication technologies and network measurements, we refer to the survey of Boutaba et al. [16]. The mentioned approaches in particular differ in whether the data used for learning comes from network simulators (e.g. from OMNeT++) or from measurements. In addition, they can be divided into supervised, unsupervised and reinforcement learning. Mestres et al. [17] investigate modeling and prediction of delays in communication networks with feed-forward neural networks. They predict the latency based on the traffic configuration. In contrast to the RouteNet architecture, this neural network model has to be trained for each specific communication network.

IV. Evaluation

To assess the flexibility and limitations of our GNN implementation, we made changes to traffic and network topology settings. For these two categories we performed simulations (called scenarios from here on), each one with a specific change of a single setting. Each scenario consists of 5000 data sets, that is, it represents the results of 5000 simulations.

Each sample consists of per-path delay values for each source-destination pair in the network. For example, in the REDIRIS topology (19 nodes), there are 342 path delay values (corresponding to the 19×18 possible source-destination pairs). We evaluate all our results based on the Mean Absolute Percentage Error (MAPE):

$$\text{MAPE} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \frac{|y_p - \hat{y}_p|}{y_p}, \qquad (2)$$

where y_p is the delay of path p from the simulation (the true value), $\hat{y}_p$ is the value predicted by the GNN model, and the sum is over the set of all paths $\mathcal{P}$.

As delay values and predicted delay values are not independent within one network scenario, we average over all values within one network sample. Note that each network consists of the same amount of delays, therefore weighted and unweighted means are identical. We performed a sample size calculation assuming an approximate normal distribution for the per-network weighted means to detect an effect of $1.5\% \pm \Delta$ for $\Delta = 0.125\%$ with a power of 95% at a two-sided nominal level of $\alpha = 5\%$ (for the MAPE). The required sample size is 4694. As the simulated data sets typically contain a few simulation timeouts, we ran 5000 simulations to make sure that the scenarios we analyze are based on at least 4700 simulations after the removal of timeouts.

Because it is generally difficult to explain why a neural network produces a certain result, we do not try to explain individual results, but only report the accuracy of the prediction. We will, however, describe general characteristics, which provide at least some ideas under which conditions the predictions of the neural network are good. To gain deeper understanding why an individual prediction is good or bad a possible starting point is explainable artificial intelligence [18]–[20].

A. *Underlying Simulations*

For the experiments made in this paper, we generated datasets with the OMNeT++ packet-level simulator [2]. In each simulation we generate an input scenario, defined by a combination of: (1) a network topology, (2) a configuration description, and (3) a source-destination traffic matrix. To generate the traffic matrices, we use the following formula:

$$\mathcal{T}(\text{source}, \text{destination}) = \mathcal{U}(0.1, 1) \cdot \text{TI} \tag{3}$$

Where $\mathcal{U}[0.1, 1]$ is a continuous uniform distribution in the range $[0.1, 1]$, and TI represents a tunable parameter of the traffic intensity in the simulation (maximum per-path bits per time units). In each source-destination path, packet inter-arrival times (i.e., the sending interval) are modeled with an exponential distribution, where the mean is computed from the traffic volumes defined in the traffic matrix $\mathcal{T}$. Also, if not otherwise indicated, the packet size follows a scaled Bernoulli distribution, where 50% of the packets have $300\,\text{bit}$ and the other 50% contain $1700\,\text{bit}$. The configuration description includes a source-destination routing scheme and the queue scheduling configuration per device interface. For the routing configuration, we generate small variants of the shortest path, while for the queue scheduling configuration we consider that all interfaces have three queues with a size of 32 packets, and can implement the following scheduling policies: Strict Priority, Weighted Fair Queuing, or Deficit Round Robin. We made simulations in different topologies with variable link capacities and traffic intensities. More details about the specific samples generated for each specific experiment can be found in the corresponding section.

B. *Traffic Settings*

There are several ways in the OMNeT++-based simulator to change the simulated traffic. The first change we considered was the size distribution for packets (Table I). Note that the first line in all following result tables with a MAPE value of 1.50% shows the configuration settings and the performance of our GNN solution on the original test data sets that we used for developing the model. The original training and test data (just like the baseline data) were created using a Bernoulli distribution that assigns with equal probability either a packet size of $300\,\text{bit}$ or of $1700\,\text{bit}$. Note that the configuration settings used to generate the training, test and baseline data were chosen to a generate a somewhat realistic dataset for the development of RouteNet, however those settings were not directly derived from real-world traffic captures. We then considered a uniform distribution that randomly assigns a packet size within the interval $[300, 1700]$. In particular, packet sizes occur that do not exist in the training data or evaluation data. The MAPE in that case is 9.55%.

We also considered deterministic packets sizes, that is, all packet sizes are equal to $1000\,\text{bit}$. The average packet size in the baseline data is also $1000\,\text{bit}$ (expected value of the Bernoulli distribution described above). However, the result of 18.41% is worse than the first case with the uniform distribution. We also tried different deterministic packet sizes, for $1700\,\text{bit}$ a mean absolute percentage error of 42.92% is obtained and for $300\,\text{bit}$ the error is 6812.96%.

TABLE I: Results for changes of the packet size distribution

Packet size distribution	MAPE [%]
Bernoulli (300 bit and 1700 bit)	1.50
Uniform (300 bit to 1700 bit)	9.55
Deterministic (1000 bit)	18.41
Deterministic (300 bit)	6812.96
Deterministic (1700 bit)	42.92

Another option for varying the traffic is the distribution of packet inter-arrival times (Table II). The training and baseline data is generated by a modified exponential distribution where the right tail is limited to 10 times the mean to avoid arbitrarily large values. We then considered a deterministic setting (with the same mean inter-arrival time) which resulted in a MAPE of 20.24%.

TABLE II: Result for changes of the packet inter-arrival times

Inter-packet time distribution	MAPE [%]
Exponential	1.50
Deterministic	20.24

We also tested different variations of traffic intensity. A higher intensity leads to more congestion in the network and thus higher delay values. The GNN model performs better in cases of lower traffic intensity (Table III, see also Section IV-A for a definition of traffic intensity in this context).

TABLE III: Results for changes of the traffic intensity

Traffic intensity	MAPE [%]
200 - 2000	1.50
400 - 4000	11.24
800 - 2000	1.79
200 - 1000	0.6

Finally, in all simulations we use 3 Types of Service (ToS) for prioritizing traffic. Each port of a network device also has 3 queues, one for each of the 3 ToS classes. A flow is randomly assigned a ToS with equal probability. In the following, these probabilities are changed (Table IV). The results for the MAPE are between 1% and 2%, which is close to the baseline value of 1.5%. This indicates that the GNN model is robust against changes of the ToS distribution. Note that we did not consider the case of increasing the number of ToS classes as we cannot incorporate such a change in the GNN model without retraining.

C. *Network Setup*

In this section, the network topology and network device properties are varied. The simulator and our GNN model

TABLE IV: Results for changes of the ToS distribution

ToS 0 [%]	ToS 1 [%]	ToS 2 [%]	MAPE [%]
33.33	33.33	33.33	1.50
15	40	45	1.16
20	30	50	1.03
45	30	25	1.78

support three different scheduling policies: Strict Priority (SP), Weighted Fair Queuing (WFQ) and Deficit Round Robin (DRR). For the policies WFQ and DRR, 5 different weight profiles are used and randomly assigned to the outgoing ports of network nodes. The weight profiles used in the original training and test data are shown in Table V.

TABLE V: Baseline weight profiles

WFQ			DRR		
Queue 1	Queue 2	Queue 3	Queue 1	Queue 2	Queue 3
90	5	5	80	10	10
33.33	33.33	33.33	33.33	33.33	33.33
60	30	10	60	30	10
50	40	10	70	20	10
75	25	5	65	25	10

TABLE VI: Variant 1 weight profiles

WFQ			DRR		
Queue 1	Queue 2	Queue 3	Queue 1	Queue 2	Queue 3
85	7.5	7.5	70	15	15
35	35	30	35	35	30
70	25	5	70	25	5
40	40	20	85	10	5
80	15	5	55	25	20

TABLE VII: Variant 2 weight profiles

WFQ			DRR		
Queue 1	Queue 2	Queue 3	Queue 1	Queue 2	Queue 3
95	2.5	2.5	88	6	6
40	35	25	40	35	25
50	35	15	50	35	15
55	35	10	55	35	1
80	13	7	80	12	8

TABLE VIII: Variant 3 weight profiles

WFQ			DRR		
Queue 1	Queue 2	Queue 3	Queue 1	Queue 2	Queue 3
87	7	6	85	8	7
40	30	30	37	35	28
65	25	10	65	30	5
55	40	5	75	15	10
70	20	10	60	25	15

In a first step, we investigated the impact of these weights on the GNN's performance by changing the weights to slightly different values as shown in Tables VI to VIII. The results in Table IX suggest that the model has some robustness regarding the modification of scheduling policy weights.

In a next step we increased the link capacities of every link in the network by a factor of 2, 5 or 10, respectively. To ensure a comparable degree of link utilization, the respective traffic intensities have also been multiplied by the same factor. The MAPE increases with higher link capacities (Table X).

TABLE IX: Results for changes of the scheduling policy weight profiles

Weight profile	MAPE [%]
Baseline	1.50
Variant 1	3.45
Variant 2	3.76
Variant 3	2.97

TABLE X: Results for changes of the link capacities and accordingly scaled traffic intensities

Traffic factor	MAPE [%]
1	1.50
2	35.34
5	181.60
10	744.07

Finally, we measured the performance of the model on different network topologies from the Internet Topology Zoo[3] (BASNET and Bell Canada) and two artificially-generated networks (Testnet 20 and Testnet 60). Particularly, the latter topologies were generated according to the Power-Law Out-Degree algorithm [21].

BASNET with only 6 nodes represents a smaller network than the baseline REDIRIS network with 19 nodes, while Bell Canada with 48 nodes represents a larger network. Testnet 20 is a scale-free network with 20 nodes, about the same size as our baseline network; Testnet 60 is a scale-free network with 60 nodes. Note that the routing of flows through a network was always based on shortest path routing. If there were multiple shortest path in a network, one of them was randomly selected for each simulation run. Also note that despite varying the network sizes, the link capacities in the networks were not adapted.

While our GNN-based solution demonstrated the ability to perform well in unseen network topologies [10], it has difficulties scaling to larger networks (Bell Canada) and networks of different characteristics (Testnet 20 and 60).

TABLE XI: Results for other network topologies

Topology	MAPE [%]
REDIRIS	1.50
BASNET	0.48
Bell Canada	79.57
Testnet 20	71.13
Testnet 60	146.84

V. Results and Discussion

As seen in the previous section, the GNN model performs worse in certain scenarios. One possible explanation is that the

[3]http://www.topology-zoo.org

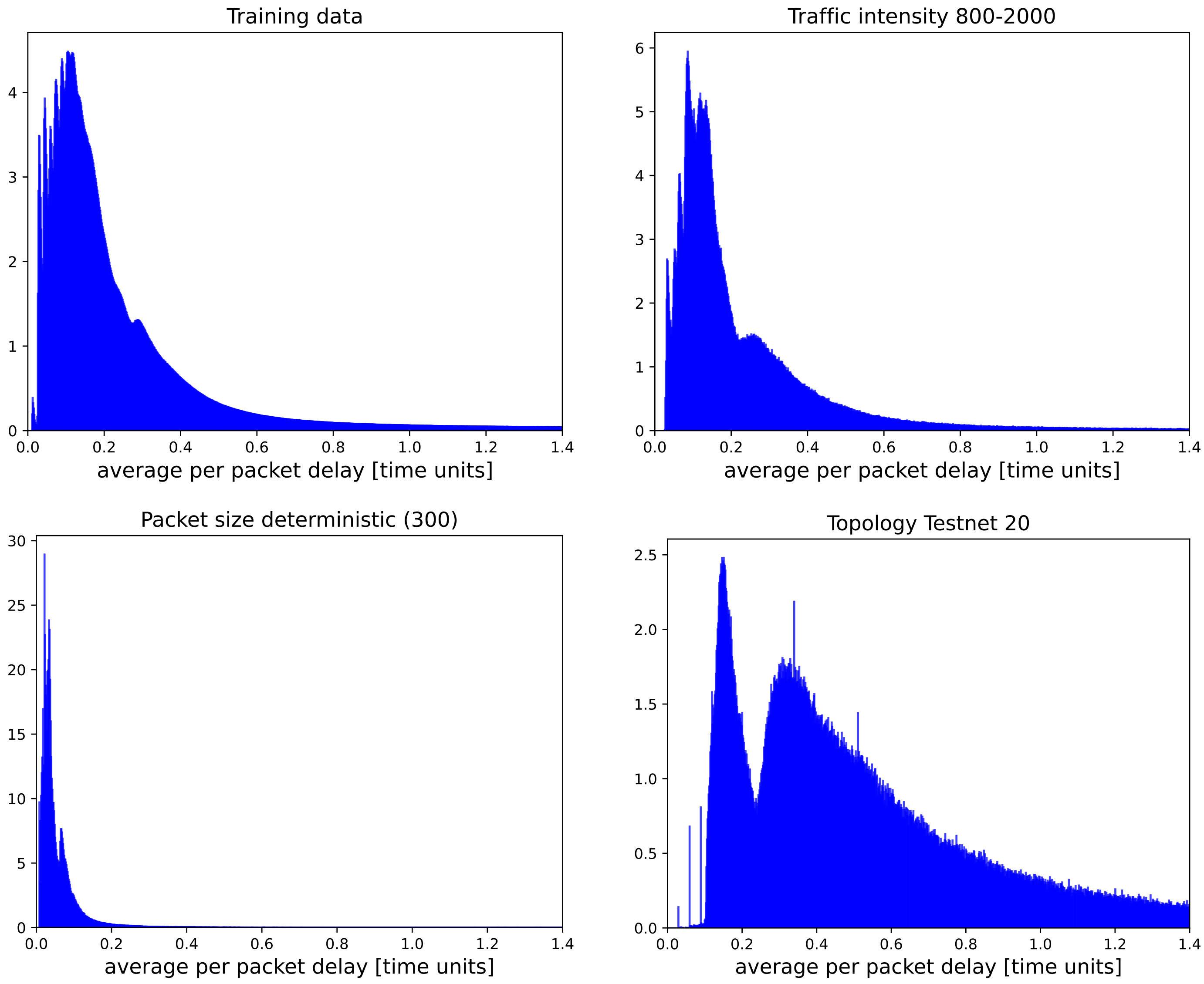

Fig. 2: Empirical probability density functions of all delay values of the training data differs from other scenarios. The distribution of delay values of the training data and of the scenario *Traffic intensity* $800-2000$ are similar ($p^* = 0.01$). In the scenario *Packet size deterministic (*300*)* the delay values tend to be smaller than in the training data ($p^* = 0.35$). In the scenario *Topology Testnet 20* the delay values tend to be larger than in the training data ($p^* = 0.42$).

distribution of delay values for which the model was trained and the distribution that appears in some scenarios are too different (see Fig. 2). To investigate this hypothesis, there are multiple metrics that provide a measure of dissimilarity of two univariate distributions, e.g. the Kullback-Leibler divergence or the Wasserstein metric. However, for this analysis we decided to use the non-parametric, unweighted relative effect, which is identical to the area under the receiver operating characteristics curve (AUC). One reason for that decision is that metrics do not give us any information in "what way" the distributions differ. Our hypothesis is that the model has difficulties handling delay values larger than those seen in the training data. Hence, it would be useful to have a criterion that takes this into account. Another reason is that the relative effect can be efficiently calculated even for large data sets as it is the case for the new test data.

Let us denote the distributions of the delay values for the training set by F_0 and for an alternative scenario by F_1. Then the relative effect is defined as

$$p = \int F_0 dF_1 = P(X_0 < X_1) + \frac{1}{2}P(X_0 = X_1), \qquad (4)$$

where $X_0 \sim F_0$ and $X_1 \sim F_1$ are the corresponding random variables. The receiver operating characteristics curve is the function given by the graph $(1 - F_0(c), 1 - F_1(c))$ where c can be thought of as the cut-off value for the comparison of the distributions, and the area under the receiver operating

TABLE XII: Similarity of distributions of analyzed scenarios versus training data

Scenario	p^*	MAPE [%]
Baseline	0.03	1.50
Packet size uniform (300 to 1700)	0.06	9.55
Packet size deterministic (1000)	0.07	18.41
Packet size deterministic (300)	0.35	6812.96
Packet size deterministic (1700)	0.10	42.92
Packet timing Deterministic	0.06	20.24
Traffic intensity 400-4000	0.14	11.24
Traffic intensity 800-2000	0.01	1.79
Traffic intensity 200-1000	0.13	0.6
ToS 15% - 40% - 45%	0.03	1.16
ToS 20% - 30% - 50%	0.03	1.03
ToS 45% - 30% - 25%	0.03	1.78
Scheduling weights variant 1	0.03	3.45
Scheduling weights variant 2	0.03	3.76
Scheduling weights variant 3	0.03	2.97
Traffic factor 2	0.23	35.34
Traffic factor 5	0.39	181.60
Traffic factor 10	0.42	744.07
Topology BASNET	0.17	0.48
Topology Bell Canada	0.39	79.57
Topology Testnet 20	0.42	71.13
Topology Testnet 60	0.48	146.84

characteristics curve can be calculated by

$$\begin{aligned} \text{AUC} &= \int_{-\infty}^{\infty} \big(1 - F_1(c)\big) d\big(1 - F_0(c)\big) && (5)\\ &= \int_{-\infty}^{\infty} F_0(c) dF_1(c) && (6)\\ &= p. && (7)\end{aligned}$$

Hence, an interpretation of the relative effect is the accuracy to distinguish between training data and the alternative scenario with respect to delay values. If the relative effect is

- $p \approx 1/2$, then the delay values for the alternative scenario do not tend to be larger or smaller compared to the training data,
- $p < 1/2$, then the alternative scenario tends to have smaller delay values compared to the training data,
- $p > 1/2$, then the alternative scenario tends to have larger delay values compared to the training data.

Furthermore, it is also noteworthy to consider

$$p^* = \left|p - \tfrac{1}{2}\right| \tag{8}$$

as a measure of dissimilarity between two distributions. Note that the relative effect is between 0 and 1 as it is a probability. A relative effect $p = 0$ or $p = 1$ would imply completely separated distributions. In other words the distributions have no overlap. One drawback of the relative effect is that there are cases where $p = 1/2$ but $F_0 \neq F_1$.

The correlation between p^* and MAPE is 0.32. Without the scenarios "deterministic 300" and "traffic factor*10" (with the two highest MAPE of 6812.96% or 744.07%, respectively), the correlation between p^* and MAPE rises to 0.86. That means p^* can give in most cases a general idea how good the MAPE will be, see Table XII and Fig. 3. In other words, the GNN model's performance depends on the differences on

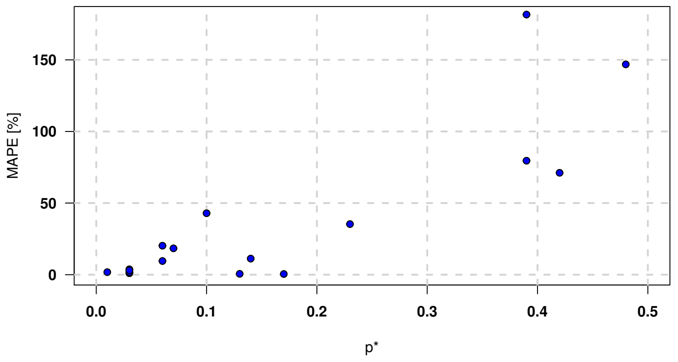

Fig. 3: The scatter plot of p^* and MAPE illustrates the correlation of 0.86. Note that the two outliers with MAPEs of 6812.96% and 744.07%, respectively, are not contained in this plot.

the distribution of delay values between training data and new test data.

VI. Conclusion

In this paper we have performed a comprehensive analysis on the generalization limits of RouteNet, a GNN-based model that represents the state of the art on ML-based solutions for network modeling. In the evaluation we have shown that we can approximately predict the degree of accuracy of this model by assessing the difference in the distribution of delay values between the training and test data. Generally, the results show some weaknesses of the current GNN approach to predict performance metrics and provides hints for future work. For example, the model does not generalize well to networks considerably larger than those seen during the training phase. In particular, this limitation was addressed in the subsequent Graph Neural Networking Challenge 2021 [15]. From the winning solutions, we could observe that an important modification to overcome this limitation was to predict queue utilizations with the GNN model and derive path delays indirectly from these values [22]. The main rationale behind this is that queue utilizations are within a bounded range [0,1], thus avoiding important distribution shifts on the model output depending on the network size. However, to achieve the final goal of creating accurate GNN-based Digital Twins for communication networks, more research will be needed to address and overcome the remaining limitations identified in this paper.

VII. Acknowledgments

The research was supported by the WISS 2025 (Science and Innovation Strategy Salzburg 2025) project IDALab Salzburg (20204-WISS/225/197-2019 and 20102-F1901166-KZP) and the 5G-AI-MLab by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) and the Austrian state Salzburg. Also, this publication is part of the Spanish I+D+i project TRAINER-A (ref. PID2020-118011GB-C21), funded by

MCIN/AEI/10.13039/501100011033, and it is partially funded by the Catalan Institution for Research and Advanced Studies (ICREA).

References

[1] 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," ser. SOSR, San Jose, CA, USA: Association for Computing Machinery, 2019, pp. 140–151, ISBN: 9781450367103.

[2] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, 2008, p. 60.

[3] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*, Springer, 2010, pp. 15–34.

[4] L. Kleinrock, *Communication nets: Stochastic message flow and delay*. Courier Corporation, 2007.

[5] A. Bouillard, M. Boyer, and E. Le Corronc, *Deterministic Network Calculus: From Theory to Practical Implementation*. John Wiley & Sons, 2018.

[6] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *Proceedings of the Conference on Computer Communications*, IEEE, 2018, pp. 1871–1879.

[7] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *Internet of Things Journal*, 2021.

[8] G. Bernárdez, J. Suárez-Varela, A. López, B. Wu, S. Xiao, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Is machine learning ready for traffic engineering optimization?" *IEEE International Conference on Network Protocols (ICNP)*, 2021.

[9] D. Pujol-Perich, J. Suárez-Varela, M. Ferriol, S. Xiao, B. Wu, A. Cabellos-Aparicio, and P. Barlet-Ros, "IGNNITION: Bridging the gap between graph neural networks and networking systems," *IEEE Network, in press*, 2021.

[10] M. Happ, M. Herlich, C. Maier, J. L. Du, and P. Dorfinger, "Graph-Neural-Network-based Delay Estimation for Communication Networks with Heterogeneous Scheduling Policies," *Journal on Future and Evolving Technologies*, vol. 2, no. 4, 2021.

[11] J. Suárez-Varela, M. Ferriol-Galmés, A. López, P. Almasan, G. Bernárdez, D. Pujol-Perich, K. Rusek, L. Bonniot, C. Neumann, F. Schnitzler, F. Taïani, M. Happ, C. Maier, J. L. Du, M. Herlich, P. Dorfinger, N. V. Hainke, S. Venz, J. Wegener, H. Wissing, B. Wu, S. Xiao, P. Barlet-Ros, and A. Cabellos-Aparicio, "The Graph Neural Networking Challenge: A Worldwide Competition for Education in AI/ML for Networks," *ACM SIGCOMM Computer Communication Review*, vol. 51, no. 3, pp. 9–16, 2021.

[12] Barcelona Neural Networking Center-UPC, *Graph neural networking challenge 2020*, https://bnn.upc.edu/challenge/gnnet2020/, Accessed: 2021-12-20.

[13] M. Ferriol-Galmés, J. Suárez-Varela, P. Barlet-Ros, and A. Cabellos-Aparicio, "Applying Graph-Based Deep Learning to Realistic Network Scenarios," *arXiv preprint arXiv:2010.06686*, 2020.

[14] 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 International Conference on emerging Networking EXperiments and Technologies*, 2019, pp. 14–16.

[15] Barcelona Neural Networking Center-UPC, *Graph neural networking challenge 2021 - creating a scalable network digital twin*, https://bnn.upc.edu/challenge/gnnet2021/, Accessed: 2021-12-20.

[16] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, 2018.

[17] A. Mestres, E. Alarcón, Y. Ji, and A. Cabellos-Aparicio, "Understanding the Modeling of Computer Network Delays Using Neural Networks," in *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, 2018, pp. 46–52.

[18] Z. Meng, M. Wang, J. Bai, M. Xu, H. Mao, and H. Hu, "Interpreting deep learning-based networking systems," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 154–171.

[19] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *International Convention on Information and Communication Technology, Electronics and Microelectronics*, 2018, pp. 0210–0215. DOI: 10.23919/MIPRO.2018.8400040.

[20] D. Pujol-Perich, J. Suárez-Varela, S. Xiao, B. Wu, A. Cabellos-Aparicio, and P. Barlet-Ros, "Netxplain: Real-time explainability of graph neural networks applied to computer networks," in *GNNSys workshop*, 2021, pp. 154–171.

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

[22] Bruno Klaus de Aquino Afonso, *Improved GNN generalization to larger 5G networks by fine-tuning predictions from queueing theory*, https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-001-PARANA, Accessed: 2021-12-20.