Predicting network performance using GNNs: generalization to larger unseen networks

Miquel Farreras[1], Paola Soto[2], Miguel Camelo[2], Lluís Fàbrega[1], Pere Vilà[1]

[1]Institute of Informatics and Applications, Universitat de Girona, Girona, Spain
[2]IDLab, University of Antwerp, in collaboration with imec, Antwerpen, Belgium

***Abstract*—Autonomous Fifth Generation (5G) and Beyond 5G (B5G) networks require modelling tools to predict the impact on the performance when new configurations and features are applied in the network. Modeling modern networks through traditional mathematical analysis can lead to low accuracy, while the execution time and resource usage are high in network simulators. Machine Learning (ML) algorithms, and specifically Graph Neural Networks (GNNs), are suggested as a promising alternative since they can capture complex relationships from graph-like data, predicting properties with high accuracy and low resource requirements. However, they cannot generalize to larger networks, as their prediction accuracy decreases when input data (e.g., network topologies) is significantly different (e.g., larger) than the training data. This paper addresses the GNNs scalability issue by following a step-by-step approach, exploiting networking concepts to improve a baseline model. This work is framed in the 2021 International Telecommunication Union (ITU) and Barcelona Neural Networking Center - Universitat Politècnica de Catalunya (BNN-UPC) challenge. Results show that by following the suggested steps, applied on the RouteNet baseline developed by the BNN-UPC, can lower the Mean Average Percentage Error (MAPE) from 187.28% to 1.838%, improving the generalization significantly over larger graphs. Our approach is more simple than other solutions that participated in the challenge, but obtained similar results.**

***Index Terms*—Digital Twins, Graph Neural Networks, ITU AI/ML in 5G Challenge, Network Modeling, Network Performance.**

I. Introduction

The Fifth Generation (5G) and Beyond 5G (B5G) networks require a complex performance analysis and monitoring to support the stringent demands of Quality of Service (QoS) and Quality of Experience (QoE) for new services, e.g., augmented reality and Vehicle-to-Everything (V2X) services [1]. The modeling of concrete QoS parameters such as delay, jitter, loss, throughput, and other network's Key Performance Indicators (KPIs) for each service is crucial to develop prediction models that decision-making algorithms can use to manage and control the dynamic adaptation of the network in a more efficient way and achieve the performance requirements. Network modeling is traditionally based on mathematical analysis [2] of the network behavior, but they are often based on simplifying assumptions to keep tractability. Moreover, mathematical models do not keep up with the increasing complexity and dynamism of current networks, which makes them hard to model, manage, and control [3]. On the other hand, network simulators provide results at the packet level with high accuracy. Still, their execution time and scalability in terms of computational resources are limiting factors for larger and constantly evolving networks [4].

Evaluating network configurations on such models quickly (e.g., milliseconds) is crucial, even more, if we are dealing with time-sensitive services, which must meet a given deadline. To fulfill this purpose, an end-to-end model should be available in which, given a network configuration, this model can predict a given network KPIs as a means of supporting other decision-making processes (e.g., admission of new traffic).

Artificial Intelligence (AI) algorithms, and more concretely Machine Learning (ML) ones, are currently being proposed to create a new generation of network models that abstract the underlying complexities by learning from data. These AI/ML models can be easily updated by training them in newly available data and can offer a fast prediction of network KPIs. Therefore, they can serve to build a digital twin of the network [5], i.e., a virtual representation of a real network, to safely explore different network decisions and configurations and analyze their impact on the customers KPIs before actually deploying them.

It has been demonstrated that Neural Networks (NNs) are good function approximators [6] and can be used to model complex relationships from data. Nonetheless, current NN architectures are not designed to learn from graph-like data. These new data-based models must be as accurate as possible to offer higher reliability. To overcome this difficulty, Graph Neural Networks (GNNs) [7] are proposed and recently used to estimate the performance in networking domains accurately [8]. The complexity of the GNNs increases with the network size, slowing down the training times and requiring more resources if trained with larger networks. A possible solution to this problem is to have a GNN-based model that is trained using data from network topologies of small size, in terms of the number of nodes, and being able to generalize (i.e., to produce accurate results) when tested in larger and unseen networks, offering fast training time and good scalability properties. That is precisely the goal of one of the problem statements of the 2021 edition of the ”ITU AI/ML in 5G Challenge: applying machine learning in communication networks” [9]. This paper relates our experience participating in this challenge. We followed an improving step-by-step approach, where besides following some recommendations from the organizers, we applied feature engineering using basic network concepts to improve the accuracy of a baseline.

The remainder of this paper is organized as follows. Section II briefly introduces the most used approaches for network delay modeling, including the RouteNet baseline. Section III introduces the RouteNet baseline and the Graph Neural Networking challenge. In Section IV, we present Girona Antwerp Intelligence for Networks (GAIN), an improved RouteNet-based architecture able to infer more accurately the average per-path delay in large networks while being trained in small networks. Section V shows the comparative results we obtained in five different scenarios, presenting the solution that outperforms the RouteNet baseline reducing its Mean Average Percentage Error (MAPE) from 187.28% to 1.838%. Section VI exposes the uniqueness of the GAIN solution compared with other competitors. Finally, we conclude the paper in Section VII.

II. Related work

To predict performance of a network for a particular configuration, a model and an optimization algorithm are required. Analytical modeling, including queuing theory [10] and other algorithms applied to 5G and B5G [11], are traditionally used to develop network models. However, in such modeling the assumption of non-realistic and static properties of real-world networks leads to poor predictions.

Packet-level network simulators are also used to simulate network behaviors in specific conditions with more accurate results. Unfortunately, their performance is tied to assumptions and their computational complexity and higher execution time [4] are a constraint for developing a fast and scalable Digital Twin. Nevertheless, given their high accuracy, network simulators are used to generate relevant data in which ML models can be trained.

AI/ML solutions have been used to solve similar 5G challenges, such as network slicing [12], [13], applying techniques like reinforcement learning, multilayer perceptrons, or convolutional neural networks. Lately, GNNs are being proposed to create networking models directly from data (i.e., a data-driven approach) since GNNs are able to learn the complex relationships that arise from nodes and links [14], which makes them more suitable to this domain than classical ML techniques or Deep Learning (DL) [8]. Based on these properties, RouteNet [15] model can predict KPIs by learning directly from graph-like data such as network topologies, routings, and served traffic as described in the next section.

III. RouteNet and the ITU AI/ML 5G challenge

A. *Baseline*

RouteNet is a GNN model whose principal function is to predict per-source-destination network KPIs given a particular network configuration. In particular, as shown in Fig. 1, RouteNet learns the network model from the data collected over different network topologies such as their traffic and their routing configurations, which can later on be used to predict their performance. The resulting model captures the complex relationships between properties of links and source-destination paths in topologies.

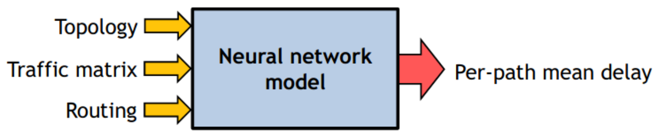

Fig. 1. RouteNet model inputs and output [15]

The main working principle of RouteNet is the differentiation of the data between the path-level (e.g., end-to-end delay, end-to-end packet loss) and the link level (e.g., link delay, link utilization). This information is encoded in learnable vectors. Based on this assumption, the message-passing procedure used by RouteNet [15] follows the principles:

1) The state of a path depends on the state of all the links that lie on the path.
2) The state of a link depends on the state of all the paths that traverse the link.

The learnable vectors are used to execute a message-passing procedure (with a determined number of rounds) to collect messages from all the links and paths. This message-passing procedure, mixed with the usage of Recurrent Neural Networks (RNNs), make GNN architecture able to infer path- and/or link-level metrics. In RouteNet, each RNN's hidden state represents a function with the information of the path or link. These RNNs have modifiable hyper-parameters to adapt to the specific use case.

Prediction errors are measured in terms of Mean Average Percentage Error (MAPE), which is defined as the average value of the relative errors, in percentage as shown in Eq. (1). The relative error is calculated by the difference between the predicted value $\hat{y_l}$ and the real value y_i, divided by y_i and obtaining the result in absolute value. The lower the MAPE value, the better the predictions.

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

B. *Challenge*

For the challenge, we used RouteNet as a baseline model. Unfortunately, the RouteNet model exhibited poor scalability properties. For instance, when trained with small networks and then used to predict results for larger networks, RouteNet shows very high errors in the per-path average delay, in terms of MAPE.

During the second edition of the Graph Neural Networking challenge, the focus was on creating a scalable digital twin for networks by tackling a fundamental limitation of existing GNNs: the lack of generalization capabilities to larger network topologies. Participants were asked to outperform the RouteNet baseline, modifying it or designing a new GNN-based model. The main goal was to train the model with a dataset with network topologies of small size and then run the validation/testing phase over a dataset with larger topologies, maintaining a low prediction error.

This section summarizes the steps to create a scalable solution based on RouteNet implemented in TensorFlow[1].

C. Dataset

The datasets for the challenge were provided by the BNN-UPC. These datasets were generated with OMNet++, a packet-level network simulator. The basic RouteNet model receives the following inputs per each sample in the dataset:

1) Topology: graph including node and link-level properties (e.g., nodes, links, queue sizes, link capacity).
2) Flow performance: flow level and aggregate source-destination measurements (e.g., dropped packets, average delay).
3) Flow traffic: flow level and aggregate source-destination time and size distributions used to generate traffic (e.g., average bandwidth, packets generated, average packet size).
4) Routing: paths connecting source-destination pairs.
5) Link performance: metrics of output ports (e.g., utilization and losses, average port occupancy).

Three datasets proposed by the BNN-UPC were available for the training, validation, and testing stages of the challenge:

- Training: small networks between 25 to 50 nodes.
- Validation: larger networks from 50 to 300 nodes to analyse the ability of the models to generalize.
- Test: equivalent characteristics to the validation dataset.

Moreover, the validation and testing datasets were subdivided by the BNN-UPC into three subsets called settings, having different features each:

1) Setting 1 (S_1). Longer paths. Focused on the artificially generated longer paths feature concerning the training dataset. However, the link capacities have the same value ranges as in the training dataset. Some source-destination pairs do not transmit traffic.
2) Setting 2 (S_2). Increased link capacity. Focus on the larger link capacity feature with respect to the training dataset. All the source-destination pairs transmit traffic using shortest-path routing. Moreover, there is higher aggregated traffic and higher capacity links than in the training dataset.
3) Setting 3 (S_3). Both properties mixed. All source-destination pairs transmit traffic using longer paths with higher capacity links than the training dataset.

IV. Improving RouteNet for scalability

In this section, the improvements to the RouteNet baseline are described step by step, including the improvement achieved. Our work follows step-by-step testing of individual modifications and measuring the enhancement of each applied change, one after the other.

[1]https://github.com/BNN-UPC/GNNetworkingChallenge/tree/2021_Routenet_TF

A. Inferring per-path delay from predicted queue occupancy (GAIN 1 solution)

Since the validation and test datasets contain topologies with higher link capacities, the delay values in larger topologies are lower than in smaller topologies. This implies that the model is trained with data following a particular data distribution but has to predict values that follow another distribution, which is problematic for NNs [16].

The initial idea to solve this problem, which the challenge organizers suggested, consisted in finding an indirect metric that keeps a similar distribution among all the datasets and can be used, after a post-processing step, to predict the primary metric of the challenge (i.e., the path delay) with low error. Based on this idea, our first approach was to use the occupancy of a link (Q_o) as the indirect metric, representing the average utilization of a queue since it encapsulates local relationships between offered traffic, queue size, and link capacity. The first step, titled GAIN 1, was to test the improvement of the original RouteNet implementation, replacing the initial prediction of delay with the prediction of the indirect metric Q_o and inferring the path delay from that prediction.

After predicting Q_o, we estimate each flow's delay by adding the queue delays belonging to a path in a post-processing step. Let's assume a path with three nodes as depicted in Fig. 2. When the packets are sent from the *src* node, they are queued in Q_1 accumulating a delay. Depending on the queuing policy (assumed to be the same in each queue), the queue occupancy, and the link capacity, the packets will suffer more or less delay. If a link has more capacity, the waiting time of a packet in the queue will be lower. Once the packets are sent to the second node using $Link_1$, they are queued in Q_2, adding another delay. Finally, the packets are sent through $Link_2$ to the *dst* node. Therefore, the delay of a source-destination flow can be obtained as the sum of the delays in every flow link.

Each queue delay of this flow was computed as Eq. (3), where Q_o is the occupancy of the queue, Q_s is the queue size in number of packets, A_p is the average packet size in number of bits, and C_o is the capacity of the outgoing link of the queue. The link delay and flow delay formulations are presented below.

$$Delay_{\,flow} = \sum_{k=1}^{N_f} Delay_{\,link} \tag{2}$$

$$Delay_{\,link} = Q_o \times Q_s \times A_p / C_l \tag{3}$$

The post-processing code was optimized using the appropriate TensorFlow data structures to increase the processing speed and reduce code complexity. The resulting MAPE of the GAIN 1 solution was 44.73 %.

B. Normalization of predictor features (GAIN 2 solution)

After analysing the training and validation datasets, it was observed that data distributions of the scalar features used in the model (detailed in Table II) were different for training, compared to validation and test subsets, as shown in Table I.

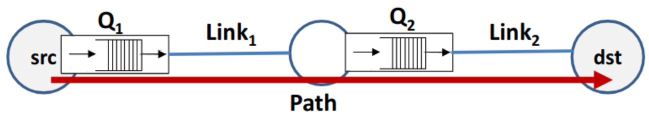

Fig. 2. Queue occupancy example to infer delay

TABLE I
GAIN USED FEATURES MAXIMUM AND MINIMUM VALUES

	Training		Validation		Test	
	Min	**Max**	**Min**	**Max**	**Min**	**Max**
Traffic	30.787	2048.23	30.543	2064.93	0	2061.17
Packets	0.0329	2.03633	0.03107	2.03985	0	2.0543
Capacity	10000	100000	10000	2500000	10000	2500000

Specifically, the traffic of a flow (T_f) and the capacity of a link (C_l) had different value ranges for each dataset and between the training and validation/test datasets, respectively.

Therefore, in GAIN 2, a min-max normalization was applied as a pre-processing in the *transformation* function, using the training dataset min-max values. The max-min normalization was calculated using the formula in Eq. (4). Normalized value N_v equals to the original value V subtracting the minimum value of the feature for all the training dataset V_{min}, and dividing the result by the difference between the maximum V_{max} and V_{min} of the training dataset. The TensorFlow functions were used to find the maximum and minimum values on the training dataset's tensor, which efficiently reduce the tensor to one dimension to return the desired value.

$$N_v = \frac{V - V_{min}}{V_{max} - V_{min}} \tag{4}$$

The result of the GAIN 2 solution was a 28.739 % MAPE.

C. Feature selection (GAIN 3 solution)

The features included in the baseline were T_f and C_l, as described in Table II. After a correlation test between the features available and the expected delay results in the datasets, packets (P_f) revealed a high correlation to path delay. In addition, this feature had value ranges very similar in the three datasets, enhancing scalability. Consequently, it was added to the GAIN 3 path state, also including the min-max normalization pre-processing. The MAPE of the GAIN 3 solution was a 18.471 %.

TABLE II
FEATURES USED IN THE GAIN SOLUTIONS

Path features		
Feature	**Definition**	**Used in**
Traffic (T_f)	Traffic in a path (bits/time unit)	All GAIN and baseline
Packets (P_f)	Packets generated in a path (packets/time unit)	GAIN 3,4,5
Link features		
Capacity (C_l)	Link bandwidth (bits/time unit)	GAIN 1,2,3 and baseline
Offered Traffic Intensity (O_t)	Sum of T_f in a link divided by C_l	GAIN 4,5

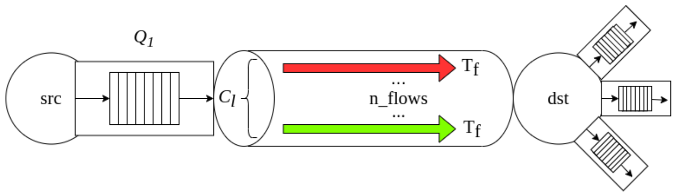

Fig. 3. Offered traffic intensity example of flows in a link

D. Offered Traffic Intensity (GAIN 4 solution)

When approaching the total link capacity, increasing the total T_f in a link directly impacts the queue delay by increasing it too. Fig. 3 shows an example where a source *src* sends traffic to *dst*. If the sum of T_f of each flow in the link is close to or exceeds C_l, the Q_1 delay will increase.

The dependence between T_f demand in a link and the queuing delay [17], leads to the creation of a new feature named *offered traffic intensity* (O_t), as shown in Eq. (5). It was defined as the sum of the T_f of all the flows f passing through the network link N_l divided by the bandwidth C_l of that link, resulting in a scalar feature assigned to the link state.

$$O_t = \frac{\sum_{f \in N_\ell} t_f}{C_\ell} \tag{5}$$

Finally, only the feature O_t was used in GAIN 4. For that reason, C_l did not require normalization, as it was removed from the model, and the original values were used to calculate the O_t. The resulting MAPE of the GAIN 4 solution was 2.612 %.

E. Hyper-parameters optimization (GAIN 5 solution)

The hyper-parameters, i.e. number of neurons for the path-state and link-state, the readout units, the message-passing iterations and the epochs, were optimized for our model, testing different combined values using a grid search, defining and combining values for each parameter, as detailed in Table III. For each combination of hyper-parameters, a new training followed by a check of the MAPE result was executed.

One of the effects of optimizing the hyper-parameters is the reduction of the training epochs, as the model stabilizes the validation results much earlier than the original 100 epochs. For reference, the best model achieved the best result after seven epochs. An *early stopping* epoch limiter is a technique that stops the training after a number of epochs where the model does not improve its results. The number of epochs monitored to do the early stopping is called *patience*. It was used in the GAIN 4 and GAIN 5 solutions, with a patience of 5 epochs to speed up the hyper-parameter tests.

After hyper-parameter tuning, the GAIN 5 model was trained during seven epochs. We found that reducing the link-state and path-state dimensions and increasing the readout units along the message passing procedure iterations improved the results significantly from 2.612 % to 1.838 %, the best result of all the GAIN solutions, also improving for the different subsets.

Another interesting effect of keeping the hyper-parameters and epochs in a low value is the reduced Graphics Processing Unit (GPU) power consumption, temperature, compute, and memory utilisation as exposed in Section V.

V. Results

This section reports the results obtained following the steps described in the previous section. The GAIN 2 solution was the one presented to the challenge, achieving a 28.739% MAPE. GAIN 3, 4, and 5 are the results of additional improvements after the competition.

The gradual improvement on each step is shown in Table IV, where the RouteNet Baseline is shown to have a bad generalization for larger graphs (S_1, S_2, S_3 with 300 nodes) resulting in a MAPE of 187.28% for the full testing dataset. This is mainly caused by the out-of-distribution values when predicting in the validation and testing dataset, especially in S_2 and S_3 settings. The GAIN solutions reduced the MAPE gradually, applying improvements step by step as described in Section IV. From the RouteNet Baseline and GAIN 1 solution, which was the first change over the baseline, until the GAIN 5 solution, we achieved a improvement of 24x (from 44.73 % to 1.838%) and 101x (from 187,28% to 1.828%) lower MAPE in all settings, respectively.

Concerning the GPU usage, it can be observed that in the GAIN 4 solution is 4% while in the baseline it is 19%, with the added benefit of a much faster training. One of the causes of this reduction is the better convergence of the model, as shown in Fig. 4. In GAIN 5 the GPU consumption is higher than in GAIN 4, because of the higher hyper-parameters, but GPU consumption is still kept under 15%. The hardware of the testbed used for the training was composed of two GPUs Nvidia GeForce GTX 1070 with 8 GB GDDR5 memory, paired with an AMD Ryzen 5 5600X CPU, 32 GB of DDR4 RAM, and a 500 GB M.2 SSD. The time to train and test the GAIN 5 model was about 3 hours 30 minutes, 2 hours 20 minutes for training, which is a significant reduction compared to the original 12 hours training of the baseline.

VI. Comparison against other solutions

Among the many solutions that were submitted to the challenge, here we review and compare the top ones (in the challenge classification) to our best solution (GAIN 5). A common characteristic used by all top solutions was the inference of delay from the queue occupancy prediction, as proposed by the organizers (see Section IV). This comparison is summarized in Table V.

TABLE III
Values tested for optimization of the hyper-parameters

Hyperparameter	Values tested	Original RouteNet values	Optimized GAIN 5 values
Link state dimension	4, 8, 16, 32, 64	16	4
Path state dimension	4, 8, 16, 32, 64	32	16
Readout units	4, 8, 16, 32, 64	8	64
Message passing iterations	6, 8, 10, 12, 14, 16	8	10
Epochs	100, variable	100	7

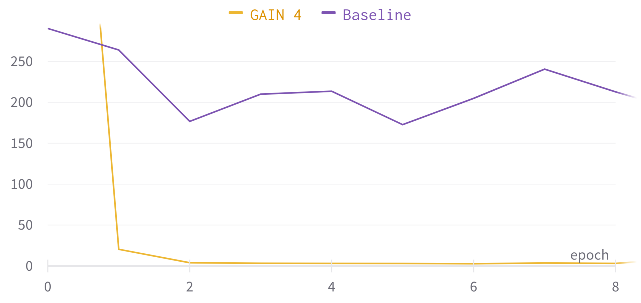

Fig. 4. Validation MAPE of the RouteNet baseline vs the GAIN 4 solution

PARANA team[2] presented a completely implemented from scratch solution to generalize to larger graphs. The generalization was based on baseline features (path, link-level prediction). Coded on Pytorch and Pytorch Geometric, two message-passing procedures were trained. The first model was focused on learning from larger networks and used paths, links, and nodes to perform the message-passing procedure. The second model focused on smaller networks where only paths and links were used. Their final submission was the average of both.

SOFGNN team[3] also used an average of two models, as well as data augmentation for out of distribution input data and feature engineering. The data augmentation is achieved using the training dataset as a base, creating new examples similar to the samples found in the validation sets. They created a new feature called *link load* (%), defined as the sum of the traffic of all flows traversing the link and divided by link capacity, which is the same approach as the offered traffic intensity we use in GAIN (see Section IV-D). They used the average of two trained models, one with the square of link load and the other including both the square and cube values of *link load*. Finally, they fine-tuned the hyper-parameters, obtaining different values than the baseline.

Out of the challenge, BNN-UPC also proposed a solution to the scalability problem of RouteNet's GNN [18]. The proposed model implements another message-passing procedure and data augmentation. A scaling factor combined with the data augmentation was introduced to estimate the delay values of higher capacity links. This model required 200 epochs to train, longer than any of the submitted solutions of the challenge.

The uniqueness of our solution (GAIN 5) is the preservation of the original RouteNet model architecture and data since the message-passing procedure was not modified and data augmentation was not used, resulting in implementation with lower overhead than in the other solutions. In addition, our approach did not use the average of multiple models to keep the simplicity of the baseline. The use of the new feature *offered traffic intensity*, designed from the knowledge of traditional queuing theory, significantly improved the accuracy of

[2]https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-001-PARANA

[3]https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-001-SOFGNN-Graph-Neural-Networking-Challenge

TABLE IV
TEST DATASET MAPE (%) RESULTS AND PERFORMANCE FOR BASELINE AND EACH GAIN SOLUTION

	Full testing dataset	S_1	S_1 **50 nodes**	S_1 **300 nodes**	S_2	S_2 **50 nodes**	S_2 **300 nodes**	S_3	S_3 **50 nodes**	S_3 **300 nodes**	**Train time**	**Avg. train GPU**
Baseline	187.28	79.145	68.481	92.979	253.075	68.481	345.135	247.217	44.669	368.019	12h 15m	$\sim$19%
GAIN 1	44.73	13.074	11.705	13.108	54.318	16.214	42.374	67.44	70.994	90.739	9h 45m	$\sim$21%
GAIN 2	28.739	11.719	11.026	11.864	35.067	17.092	39.773	31.754	17.581	31.353	9h 48m	$\sim$19%
GAIN 3	18.471	9.436	6.893	12.468	26.897	22.106	30.067	18.143	12.569	21.862	9h 40m	$\sim$15%
GAIN 4	2.612	2.652	1.539	3.687	2.492	1.386	2.567	2.584	1.607	2.363	3h 25m	$\sim$4%
GAIN 5	**1.838**	**1.407**	**1.111**	**1.808**	**1.929**	**1.573**	**1.535**	**1.756**	**1.388**	**1.462**	**2h 20m**	$\sim$15%

the predictions. The fine-tuning of GNN hyper-parameters, set smaller in our solution, allowed lower training and prediction times, less memory and power usage, making our solution more suitable in case of frequent retraining requirements.

VII. Conclusions and Future Work

A new generation of network models are being proposed based on GNNs, as they are able to i) accurately predict network KPIs, and ii) be easily changed and updated to reflect the increasing complexity and dynamism of current networks. Nonetheless, out-of-the-box GNNs are not able to generalize and scale to larger topologies, that is, when trained with small networks and then used to predict results for larger networks, their accuracy drops. In this paper we proposed GAIN, a GNNs-based model that predicts the average per-path network delay with good scalability properties.

GAIN is based on RouteNet's GNN and adds to it several improvements to achieve good scalability, including the inference of per-path delay from the predicted link queue occupancy, feature normalization, feature selection, feature engineering (offered traffic intensity) and the hyper-parameter optimization. Moreover GAIN has lower implementation overhead, lower resource needs and faster training compared to other existing solutions.

As future work, we plan to investigate other GNN architectures and improvements to achieve better performance, as well as to use it in a closed-loop control context, where GNN's predictions are used by decision-making algorithms (e.g., traffic admission control) deployed in the network.

TABLE V
COMPARISON OF SOLUTIONS FOR THE BNN-UPC CHALLENGE

Solution properties	**GAIN 5**	**PARANA**	**SOFGNN**	**ZTE AIOps**[1]	**EricRe**[2]
RouteNet baseline	X		X	X	
Hyperparameter tune	X	X	X	X	X
Fast train/execution	X		X		
Feature engineering	X	X	X	X	X
Reimplemented baseline		X			X
Message passing redesign		X		X	X
Multiple models		X	X	X	
Data augmentation			X		
MAPE (%)	**1.838**	**1.267**	**1.389**	**1.853**	**1.875**

[1] Missing details. [2] Private solution.

Acknowledgment

This project has received funding from the Red temática Go2Edge (Ref.: RED2018-102585-T), from the Ajut Pont UdG2020/23 and Generalitat de Catalunya through Consolidated Research Group 2017-SGR-1318 and 2017-SGR-1552, the Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement de la Generalitat de Catalunya for the FI-SDUR fellowship funding 2020 FISDU00590 assigned to Miquel Farreras, the Scientific Research Flanders (FWO) under grant agreement No. G055619N, and the European Union's Horizon 2020 research and innovation programme under grant agreement no.101017109 "DAEMON".

References

[1] 5GPPP, *AI and ML – Enablers for Beyond 5G Networks*, 2021.

[2] F. Ciucu *et al.*, "Perspectives on network calculus: no free lunch, but still good value," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 2012, pp. 311–322.

[3] Z. Xu *et al.*, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM 2018-IEEE*. IEEE, 2018.

[4] E. Weingaertner *et al.*, "A performance comparison of recent network simulators," pp. 1 – 5, 07 2009.

[5] P. Almasan *et al.*, "Digital Twin Network: Opportunities and Challenges," pp. 1–7, 2022.

[6] K. Hornik *et al.*, "Multilayer feedforward networks are universal approximators." *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[7] F. Scarselli *et al.*, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[8] P. Soto *et al.*, "Atari: A graph convolutional neural network approach for performance prediction in next-generation wlans," *Sensors*, 2021.

[9] ITU, "AI for Global Summit. ITU AI/ML in 5G Challenge: Graph Neural Networking Challenge 2021," [Accessed 19-01-2022]. [Online]. Available: https://aiforgood.itu.int/about/aiml-in-5g-challenge/

[10] F. Ciucu *et al.*, "Perspectives on network calculus - no free lunch, but still good value," *Computer Communication Review*, vol. 42, 10 2012.

[11] F. Debbabi *et al.*, "Algorithmics and modeling aspects of network slicing in 5g and beyonds network: Survey," *IEEE Access*, vol. 8, 2020.

[12] B. Han *et al.*, "Machine learning for network slicing resource management: A comprehensive survey," *CoRR*, vol. abs/2001.07974, 2020.

[13] D. Bega *et al.*, "Deepcog: Cognitive network management in sliced 5g networks with deep learning," *Proceedings - IEEE INFOCOM*, 2019.

[14] P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," *CoRR*, vol. abs/1806.01261, 2018.

[15] K. Rusek *et al.*, "RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020.

[16] D. Hendrycks *et al.*, "Benchmarking neural network robustness to common corruptions and perturbations," *CoRR*, 2019.

[17] J. Liebeherr *et al.*, "On the impact of link scheduling on end-to-end delays in large networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 1009–1020, 2011.

[18] M. Ferriol-Galmés *et al.*, "Scaling graph-based deep learning models to larger networks," 2021.