Journal of Network and Systems Management (2023) 31:65
https://doi.org/10.1007/s10922-023-09758-9

Improving Network Delay Predictions Using GNNs

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

Received: 25 November 2022 / Revised: 21 June 2023 / Accepted: 27 June 2023 /
Published online: 20 July 2023

Abstract

Autonomous network management is crucial for Fifth Generation (5G) and Beyond 5G (B5G) networks, where a constantly changing environment is expected and network configuration must adapt accordingly. Modeling tools are required to predict the impact on performance (packet and delay loss) when new traffic demands arrives and when changes in routing paths are applied in the network. Mathematical analysis and network simulators are techniques for modeling networks but both have limitations, as the former provides low accuracy and the latter requires high execution times. To overcome these limitations, machine learning (ML) algorithms, and more specifically, graph neural networks (GNNs), are proposed for network modeling due to their ability to capture complex relationships from graph-like data while predicting network properties with high accuracy and low computational requirements. However, one of the main issues when using GNNs is their lack of generalization capability to larger networks, i.e., when trained in small networks (in number of nodes, paths length, links capacity), the accuracy of predictions on larger networks is poor. This paper addresses the GNN generalization problem by the use of fundamental networking concepts. Our solution is built from a baseline GNN model called RouteNet (developed by Barcelona Neural Networking Center-Universitat Politècnica de Catalunya (BNN-UPC)) that predicts the average delay in network paths, and through a number of simple additions significantly improves the prediction accuracy in larger networks. The improvement ratio compared to the baseline model is 101, from a 187.28% to a 1.828%, measured by the Mean Average Percentage Error (MAPE). In addition, we propose a closed-loop control context where the resulting GNN model could be potentially used in different use cases.

Keywords Digital twins · Graph neural networks · Key performance indicator · Network modeling · Network performance

Abbreviations

5G	Fifth Generation
AI	Artificial Intelligence

Extended author information available on the last page of the article

B5G	Beyond 5G
BNN-UPC	Barcelona Neural Networking Center - Universitat Politècnica de Catalunya
DL	Deep Learning
DTN	Digital Twin Network
eMBB	Enhanced Mobile Broadband
GAIN	Girona-Antwerp Intelligence for Networks
GNN	Graph Neural Network
GPU	Graphics Processing Unit
IoT	Internet of Things
IoV	Internet of Vehicles
KPI	Key Performance Indicator
MAPE	Mean Average Percentage Error
MAPE-K	Monitor-Analyze-Plan-Execute over a shared Knowledge
MEC	Multi-access Edge Computing
ML	Machine Learning
MMTC	Massive Machine Type communications
NFV	Network Function Virtualization
NN	Neural Network
QoE	Quality of Experience
QoS	Quality of Service
RNN	Recurrent Neural Network
SDN	Software Defined Networking
SFC	Service Function Chaining
SLA	Service Level Agreement
URLLC	Ultra-Reliable Low-Latency Communication

1 Introduction

Fifth Generation (5G) and Beyond 5G (B5G) networks require comprehensive performance analysis and monitoring to meet the strict demands of Quality of Service (QoS) and Quality of Experience (QoE) for new services, such as Ultra-Reliable Low-Latency Communication (URLLC), Massive Machine Type Communication (mMTC) and Enhanced Mobile Broadband (eMBB) [1]. To realize such services, network operators create and deploy network slices. Network slices are multiple logical virtual networks on top of a shared network domain, using a shared physical network and shared computing resources. However, one of the biggest challenges for network automation is the dynamic resource allocation of network slices [2].

Such slices must be proactively and dynamically created, deployed, and optimized in (near)-real-time [3] to guarantee the expected QoS/QoE (e.g., latency below a maximum, bandwidth above a minimum, and a minimum of connected devices) for demanding applications such as virtual reality, smart stadiums, etc. According to [4], forthcoming infrastructures are expected to incorporate technologies such as Software Defined Networking (SDN), Network Function Virtualization (NFV), Service Function Chaining (SFC), and Multi-access Edge Computing

(MEC). The integration of these advanced features requires a quick analysis of the Key Performance Indicators (KPIs). For example, new usages like Internet of Vehicles (IoV) are demanding a high level of adaptation to the networks [5]. Therefore, a fast and highly automated management system is required to trade between demand, benefits, and capacity for existing and newer slices.

Several tools can aid the management system in achieving automation. Traditional network modeling is based on mathematical analysis of the network behavior, but they rely on simplifications that often do not reflect the complexity and dynamism of current network models [6]. In contrast, network simulators offer highly accurate results at the packet level. However, their processing time and usage of computer resources grow exponentially with the network size and parameter configuration (e.g., constant change of network conditions). Consequently, existing networks are challenging to analyze, monitor, and manage in almost real time.

Recently, Machine Learning (ML)-based predicting models are trained with data to improve network decision-making. These models play a fundamental role in network automation since they can swiftly respond to new data or values with minimal human intervention. This way, data-driven approaches can help improve prediction times and accuracy. There are currently proposals for Artificial Intelligence (AI)/ML algorithms to produce the following generation of network models that learn from data and abstract the underlying network complexities. These models can quickly predict the network KPIs and can be adjusted by retraining them with newly available data [7]. Furthermore, the usage of AI is becoming a key enabler in the current 5G and B5G network slicing landscape [8]. The Neural Networks (NNs) to approach the implementation of Digital Twin Networks (DTNs) are starting to be widely used in the research and development of this type of networks, as presented in [9–11].

It has been shown that NNs are excellent function approximators and can be used to represent complex relationships from data [12]. However, current NN designs are not made to learn from data with a graph structure. To overcome this problem, GNNs are proposed and recently employed to precisely estimate performance in networking domains, respectively [13]. GNNs learn from graph features and structures, creating more accurate data-based models. As the complexity of the GNNs increases with network size, training with larger networks takes longer and uses more resources [14]. Therefore, it is convenient to obtain a GNN-based model capable of generalizing. This model is trained using data from networks with a small number of nodes, providing fast training. At the same time, this model should produce high-accuracy results when predicting network KPIs in larger unseen networks, exhibiting good scalability properties.

This paper describes our step-by-step approach to outperform RouteNet [15], a well-known and state-of-the-art GNN model designed to predict network performance metrics. This baseline was proposed during the 2021 Graph Neural Networking Challenge: Creating a Scalable Network Digital Twin, which was part of the ITU AI/ML in 5G Challenge.[1] Our contributions can be summarized as follows:

[1] https://aiforgood.itu.int/about-ai-for-good/aiml-in-5g-challenge/

- We propose a GNN model that predicts the network delay with high accuracy and is scalable compared to the baseline model RouteNet by exploiting feature engineering techniques and fundamental networking concepts.
- We provide a detailed description of our approach toward the model design and an exhaustive analysis of the network properties related to the dataset used to train the model. This aspect is fundamental to support our decisions on the different models we design and can be used as a methodology to create GNN to solve related problems.
- We perform an extensive set of experimental evaluations where we compare our approach with recent state-of-the-art GNN approaches and analytical models, and discuss how different changes to the baseline model affect the model's performance. Real and synthetic datasets are used for a comprehensive prediction accuracy evaluation.

The rest of the paper is structured as follows. In Sect. 2, we describe the use cases where a GNN used for network KPIs predictions can be helpful. In Sect. 3, we review the main methods for predicting network delay. We introduce the RouteNet baseline in Sect. 4. After that, we present Girona Antwerp Intelligence for Networks (GAIN), an enhanced RouteNet-based architecture in Sect. 6, which can accurately estimate the average per-path delay in large networks while being trained in small networks. We evaluate GAIN and show the obtained results in Sect. 7. Finally, we conclude the paper in Sect. 8.

2 Problem Context

Predictive models are an adequate tool to study how various network configurations and decisions might affect 5G and B5G networks performance before deployment. For instance, a network management system can measure the impact of accepting new traffic on the network throughput or delay. If the existing users are not affected, the request is accepted. Otherwise, the request is rejected. To do this automatically, prediction models should be fast (e.g., in milliseconds) and accurate enough so the decision is timely and adequate.

As discussed in Sect. 1, AI/ML, particularly GNNs, are being proposed as the next generation of predictive models. GNNs have gained significant attention in the networking community due to their ability to learn complex relationships among nodes and links [16]. This makes them more suitable for this domain compared to classical ML techniques or Deep Learning (DL) [7].

Recently, GNNs have also been suggested as DTNs [17]. A DTN represents a virtual real-time depiction of a physical network in the digital world [18]. Digital twins have sparked a revolution in various industries, providing an enhanced view of physical entities to support diagnosis and what-if analyses. The requirements for near-real-time network analysis are closely linked to the concept of DTNs, which is still in the early stages of adoption for 5G and B5G networks [19, 20].

In general, ML is becoming a crucial enabler for developing new DTNs for these networks, as the demands of new services grow increasingly stringent. For

instance, in smart cities, DTNs are utilized to design networks that can ensure QoS without wasting energy, as discussed in [21]. Moreover, the emerging IoV services also rely on IoV for network planning, which can incorporate NNs and even incorporate techniques like Federated Learning [22].

Another interesting use case in 5G and B5G networks is a dynamic, real-time, routing optimization tool that accomplishes the QoS requirements. These networks continuously evolve and require an adaptive behavior, where routing can be decisive in achieving the Service Level Agreement (SLA) requirements. In this case, different routing configurations can be tested, selecting the best result according to the target KPIs. This tool should have the following functionalities:

- Monitor the current resource allocations and performance over the entire network.
- Detect or be notified by an external actor, of new network events that might affect the KPIs.
- Test multiple routing configurations and choose the optimal one for the real network.
- Minimize the used resources while satisfying the SLAs.
- Apply new network routing configurations.

The applicability of DTNs for routing is demonstrated in the state-of-the-art literature, where the usage of AI facilitates the implementation of these solutions. For example, in [23], AI is employed to implement an energy-efficient routing protocol in Wireless Sensor Networks (WSNs). In [24], a cognitive approach for routing is utilized in the context of reliable Internet of Things (IoT) networks, leveraging the capabilities of DTNs. Additionally, in the context of IoV and B5G networks, a message routing scheme is developed in [25], utilizing AI techniques.

The solutions to the use cases described above can be designed following a closed-loop control and leveraging the prediction model. As depicted in Fig. 1, network policies (i.e., SLAs) define the system's constraints or optimal operation points. All the decisions coming from the decision-making algorithm should ensure such optimal operation. A monitor block is introduced, which detects changes in the real network and formats the data to train the prediction model. Then, if a change is detected, the decision-making algorithm is triggered. The prediction model evaluates the algorithm's potential decisions and returns its impact on network policies.

Thus, the decision-making algorithm can select the decision that keeps the network operating closer to the agreed operation point. The changes, if any, are deployed in the network via an executor. The main blocks of the solution in Fig. 1 can be easily mapped to the Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) framework, one of the most influential reference control models for autonomous and self-adaptive systems [26]. MAPE-K defines five main blocks: the monitor, the analysis, the plan, the executor, and the knowledge block. The monitor and the executor are clearly represented in the figure, while the decision-making algorithm and the prediction model represent the plan and analysis block, respectively. The knowledge represents the data shared by all the other four blocks.

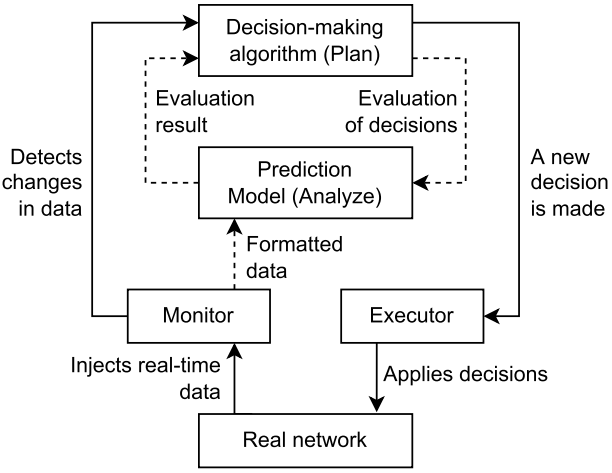


Fig. 1 Closed-loop control context for network optimization, represented by the bold arrows

More concretely, in the case of network slicing, once a new slice request arrives at the network, the decision-making algorithm should evaluate whether the slice request should be accepted. Then, the decision to accept or reject the request is simulated in the prediction model. According to the evaluations, the decision that allows fulfilling the network policy is then taken. In the case of routing optimization, the same procedure is followed, but instead of adding/rejecting slices, a routing possibility is assessed. Then, the best decision is selected and applied to the network.

3 Related Work

A predictive model and a decision-making algorithm are the main blocks needed to achieve the automated network management tasks described in Sect. 2. In this work, we focus on improving a prediction model called RouteNet, which predicts the network performance according to a given topology, routing, and offered traffic. However, other techniques can and have been used in predicting network performance. In the literature, network models are used for this objective and are typically created using analytical modeling, which includes queuing theory [27], Markov chains [28], and other similar techniques. Such models can produce fast but poor predictions as a result of relying on unrealistic and static assumptions about real-world networks. Packet-level network simulators are often utilized to model network behavior under specific cases and get more realistic results. Nevertheless, their longer computations and execution times when dealing with larger networks [29] make them unsuitable for the required fast predictions. Despite this, network simulators produce highly accurate data which can be employed to train different ML models.

AI/ML solutions such as shallow Feed Forward Neural Networks, Support Vector Machine (SVM), Random Forest (RF), Reinforcement Learning (RL), Multi-Layer Perceptrons (MLP), or Convolutional Neural Networks (CNN) have been used to

resolve similar 5G challenges. Some state-of-the-art solutions apply these methods for predicting network delay, throughput, or other KPIs. In [30], 3D-CNNs are applied to the problem of provisioning resources for network slices in real-time, predicting network capacity requirements depending on the time of the week.

In [31], a zero-touch control for network slicing is proposed, predicting each slice's network capacity needs. The authors in [32] implement a Logistic Regression (LR), an SVM, and a Decision Tree (DT) to predict the delays experienced by the users. Following a DL approach, [33] predicts the traffic that a network will support at a specific time, similarly to [30, 31], but using a Recurrent Neural Network (RNN). [34] also uses RNNs to predict delays in 5G networks for IoT and Tactile Internet. All these ML solutions work well with static scenarios but are less adapted to create graph models in comparison to GNNs. The main advantage of GNNs over other ML methods is their expressive power [35], allowing to model different graphs (e.g., different topology, number of nodes) with higher accuracy. Some additional motivations to apply GNNs to the network performance prediction are:

- GNNs have been successfully applied to combinatorial optimization problems [36], and can achieve relational reasoning [37].
- A graph representation can better capture the relationship of the nodes in a network, i.e., there is a one-to-one mapping between the network topology and the graph representation.

In that sense, RouteNet [15] is a GNN model that can directly learn from graph-like data, including network topologies, routing configurations, and offered traffic, to predict network KPIs. In recent research, other GNN solutions have been applied to solve autonomous network management [38], network slicing monitoring [39], network slicing control [40], or SDN end-to-end delay prediction [41]. However, GNNs, as ML algorithms, are sensitive to the mismatch between training data and testing data, i.e., training and testing data do not come from the same distribution. Therefore, one of the main challenges in using GNNs for predicting network performance is to improve the out-of-distribution generalization, understood as the difference in accuracy between training and testing data [42].

Several works already attempt to improve the generalization capabilities of GNNs. The authors of [43, 44] presented PARANA, a GNN model that extracts path and link-level features related to Queuing Theory (QT). The model trains two message-passing procedures. The first procedure focused on learning features from larger networks, while the second focused on learning features from smaller networks. The output is the average of both procedures. SOFGNN team [45] averaged two models and used data augmentation and feature engineering to solve the out-of-distribution problem. To obtain a higher amount of samples for the training dataset, data augmentation is applied, creating new samples similar to the ones found in the training sets. Regarding feature engineering, 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. The average result of two trained models was used, one using as features the square value of *link load*, and the other model including both the square and cube values of *link load*. Finally, the hyperparameters were fine-tuned.

BNN-UPC also proposed a solution to improve RouteNet's out-of-distribution generalization [46]. The proposed model implements an extra message-passing procedure and introduced a scaling factor combined with the data augmentation to estimate the delay values of higher-capacity links. Unfortunately, this model required a long time to train, in part due to applying only feature selection and not applying feature engineering to improve the model generalization.

Table 1 provides a summary of the mentioned recent methods for using ML, DL, and GNNs for network modeling. Many works have tried to address the network performance prediction problem. Among the most used solutions, GNNs are the ML models that are best suited to the graph representation seen in networks. Nevertheless, GNNs suffer an out-of-distribution generalization problem and some works have tried to resolve it. As shown in the next sections, the main differences in our model are fast training, high accuracy, the usage of a simple and single model, and the public availability of the code and the datasets.

4 Background

Text, social networks, photographs, molecules, and computer networks are just a few examples of daily life whose interactions can be represented as graphs. Recently, GNNs [13], a scope of deep learning, have been used to solve various issues with graphs as an input [48, 49]. In a GNN, a graph is defined as $G = (V, E)$, with V representing the set of nodes and E representing the set of edges. The edges represent the interactions between nodes. Each node has an associated set of features represented by a one-dimensional vector. The vector's length is the number of input features of the node. Edge features are also represented in a one-dimensional vector whose length is the number of input features of the edge.

Like other neural networks, a GNN is a layered architecture whose one principle is the message-passing. The idea is that neighboring nodes affect the behavior of a concurrent node. Then, a message-passing method is applied to predict the target values based on neighbor nodes. Firstly, the information of the neighbor nodes is collected, then combined, and finally, the target node information is updated. These three steps are message transformation, aggregation, and update. Message transformation functions define how each node, as well as its neighbors and edges, produce messages. The aggregation function determines how messages are grouped using the vertices' maximum, average, or element-wise sum. Simpler aggregations are possible inside a GNN, but more sophisticated aggregations, like Long Short-Term Memory (LSTM) [50], are also possible. The updating function describes how messages are updated in the target node. Figure 2 represents a basic GNN, where the colored nodes include their features in the vectors. The target A node values are predicted in this case. The transformation phase to each message is applied first, then the aggregation function, and finally, the values update for the target node. When one of the attributes of a specific node is missing, the pooling method is used to obtain information from it [51].

A sequential neural network is typically used in the readout phase to compute the output for the predictions. Depending on the prediction needs, the readout functions

Table 1 Comparison of our work and other contributions focusing on GNN

Contribution	Deep learning model	Architecture	Comparison against	Source code availability	Hyperparameter tune	Fast train	Feature engineering	Multiple models
[43, 44]	GNN	Custom with MLPs, GCGR and GAT	RouteNet	Public	Yes	No	Yes	Yes
[45]	GNN	Modified RouteNet	RouteNet	Public	Yes	Yes	Yes	Yes
[47]	GNN	RNN + GRU	OMNeT++ simulation	Public	No	No	No	No
[30]	CNN	3D-CNN + MLPs	Real scenario	Private	N/A	N/A	Yes	No
[31]	CNN	4D-CNN + MLPs	Real scenario	Private	N/A	N/A	Yes	No
[33]	RNN	SLSTM	Real data	On request	N/A	N/A	Yes	No
[34]	RNN	NARX	AnyLogic Simulator	Private	N/A	Yes	Yes	No
[32]	Classification models	SVM, DT, LR	Real data	Private	N/A	Yes	Yes	No
[38]	GNN	GCN + MLPs	City simulations	Public	N/A	N/A	Yes	No
[40]	GNN	GraphSAGE	OMNeT++ simulation	Private	No	No	Yes	No
[39]	GNN	GAT + MLPs + genetic algorithm	Genetic Algorithm + greedy solution	Private	Yes	N/A	Yes	Yes
[41]	GNN	STGCN	OMNeT++ + simulation	Private	Yes	No	Yes	No
This work	**GNN**	**Modified RouteNet**	**RouteNet**	**Public**	**Yes**	**Yes**	**Yes**	**No**

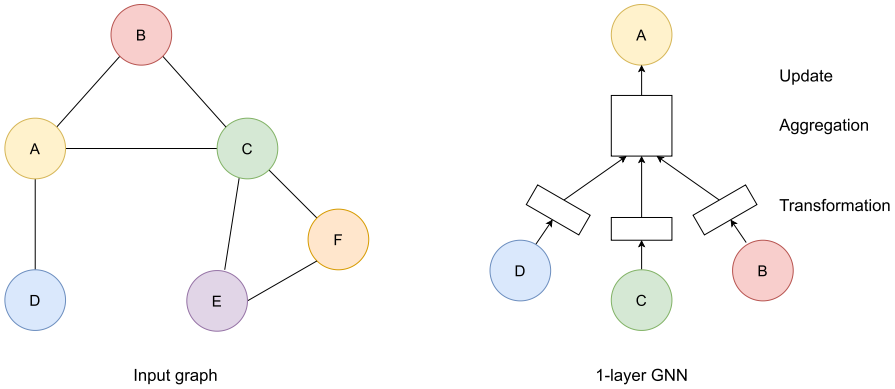


Fig. 2 GNN basic structure with message transformation, aggregation and update [52]

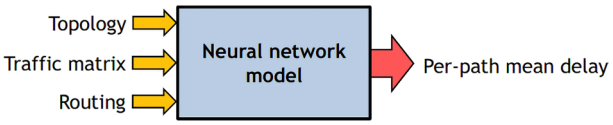


Fig. 3 RouteNet model inputs and output [15]

can be modified as classifiers or regression functions. The output of the readout function converts the values into the desired prediction for each node.

RouteNet was used as a baseline model for this work. RouteNet is a Recurrent Neural Network (RNN) model whose principal function is to predict per-source-destination network KPIs given a specific network configuration. An RNN GNN exchanges information of the graph and saves the states to the memory cell of each recurrently until convergence. In particular, as shown in Fig. 3, RouteNet learns the network model from the data collected over different network topologies, such as their traffic and routing configurations, which can later be used to predict their performance. The resulting model captures the complex relationships between the properties of links and the source-destination paths in topologies.

The representation of the graph that will feed the GNN is the first issue to be addressed in its design. Edge prediction is a limitation of current state-of-the-art GNNs, despite GNNs being able naturally to support them. Then, the input graph has to be adapted to read and predict link features. The input graph of RouteNet converts links into nodes, creating a hyper-graph for each sample, where two node types exist: path and link. 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 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, combined with the usage of RNNs, makes the GNN architecture capable of inferring path 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 hyperparameters to adapt to the specific use case.

In this paper, we use Mean Average Percentage Error (MAPE) as the prediction error. The MAPE is 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.

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

Despite the efforts to enhance RouteNet, the mismatch in accuracy between the training and testing data is still significant, which indicates room for improvement. For instance, when trained with small networks and then used to predict results for larger networks, RouteNet shows a very high error level in the per-path average delay, in terms of MAPE (cf. Sect 7). Therefore, the BNN-UPC proposed a problem statement in the context of the ITU AI/ML for 5G Challenge 2021 [53]. The challenge's goal was to create a scalable network model by tackling a fundamental limitation of existing GNNs: the lack of generalization capabilities to larger network topologies. Participants in the challenge needed to modify the RouteNet baseline or create a new GNN-based model. Such a model should be trained with a dataset containing small network topologies and later validated with larger ones, maintaining a low prediction error. This paper shows the step-by-step approach to outperform RouteNet by leveraging networking concepts and analysis.

5 Dataset

The dataset for training and testing the GNN model was provided by BNN-UPC [54], and was generated with OMNeT++, a packet-level network simulator. The 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).

The dataset also includes a preset split into training, validation, and testing as follows:

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

Moreover, the testing dataset is subdivided into three subsets called settings, having different features each:

1. Setting 1 (S_1). Longer paths. Focused on the artificially generated longer paths regarding 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 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 S_1 and S_2 properties are mixed. All source-destination pairs transmit traffic using longer paths with higher capacity links than the training dataset.

Dataset class balancing is relevant to train, validate and test the model with all the cases that it might have to predict [55]. In addition, the number of samples predicted later has also to be balanced, to check the accuracy stability in different model use cases. In the following subsections, the training, validation, and test datasets are analyzed, in order to check the feature distributions and the dataset balancing.

5.1 Training Dataset Analysis

The training dataset consists of approximately 173 million path samples. The graph size represents the number of nodes for each network. As evidenced in Fig. 4, each graph size is directly related to the number of paths (i.e., the number of samples to train the model). The paths on bigger graphs are proportionally longer. The paths are mainly of lengths 3 and 4, as illustrated in Fig. 5.

The traffic values in bits per second are balanced through the entire dataset for each graph size and path length, as displayed in Fig. 6 and Fig. 7. Packets per second are correlated with traffic in bits per second, as they depend directly on constant packet size. Other path features values, including packets dropped and the predictor variable, delay, are more dispersed in terms of value distributions. The selected link features of utilization, maximum queue occupancy, and offered traffic intensity (O_t),

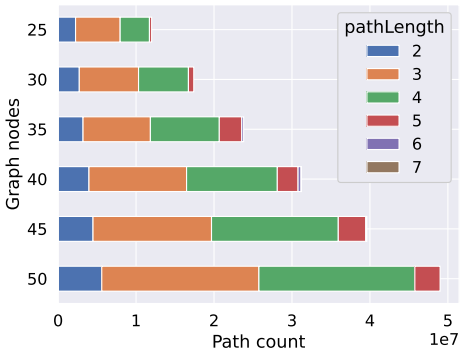


Fig. 4 Number of samples per graph size, and proportion of path lengths per graph size in training dataset

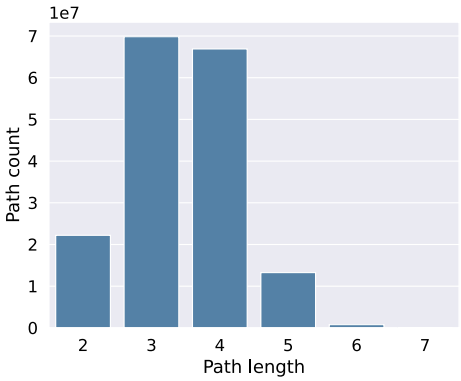


Fig. 5 Number of samples of per-path length in training dataset

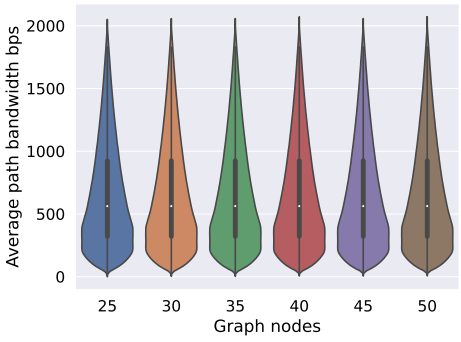


Fig. 6 Average path bandwidth distribution per graph size in training dataset

a variable created in Sect. 6.4, are also correctly balanced in the dataset, as shown in Fig. 8.

5.2 Validation Dataset Analysis

The validation dataset contains approximately 52 million path samples and follows the structure of the training dataset. The main difference is the increased number of graph nodes, ranging from 50 to 300 nodes, to test the generalization of the trained

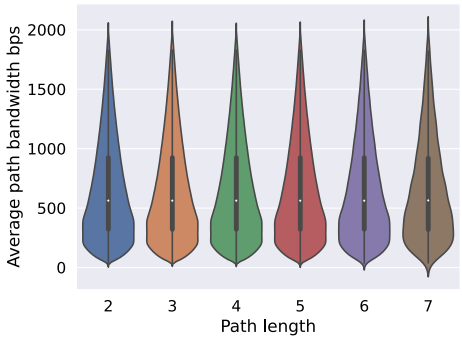


Fig. 7 Average path bandwidth per path-length in training dataset

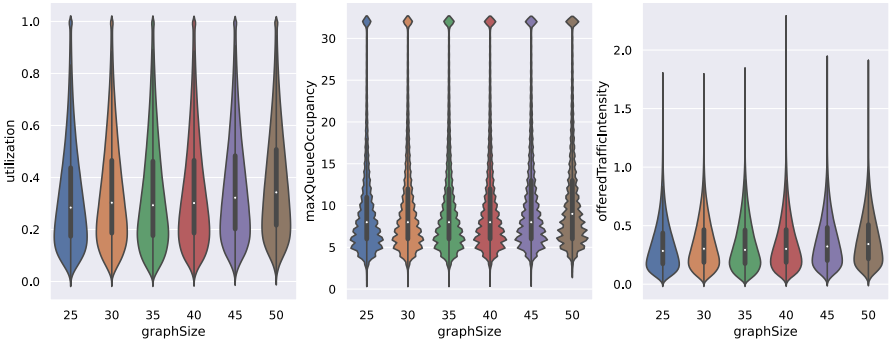


Fig. 8 Link utilization, maximum queue occupancy and O_t values distributions compared to graph size in training dataset

model. As in the training dataset and represented in Fig. 9, the paths on bigger graphs are proportionally longer. The paths mainly have a length between 3 and 10, as shown in Fig. 10.

Also similarly to the training dataset, the values of traffic in bits per second are also balanced through all the dataset for each graph size and path length as detailed in Fig. 11 and Fig. 12. In the case of longer paths, a larger deviation in the values can be observed. In terms of the analyzed link features, utilization, maximum queue occupancy, and O_t, the validation dataset is also correctly balanced, as shown in Fig. 13.

5.3 Testing Dataset Analysis

The testing dataset consists of approximately 26 million path samples and follows the structure of the train and validation datasets. As in validation, it includes the increased number of graph nodes, from 50 to 300 nodes. The main difference, in this case, is the partition in the three different subsets, S_1, S_2, and S_3. For S_2 and S_3, the paths on the bigger graphs are proportionally longer, as shown in Fig. 16 and

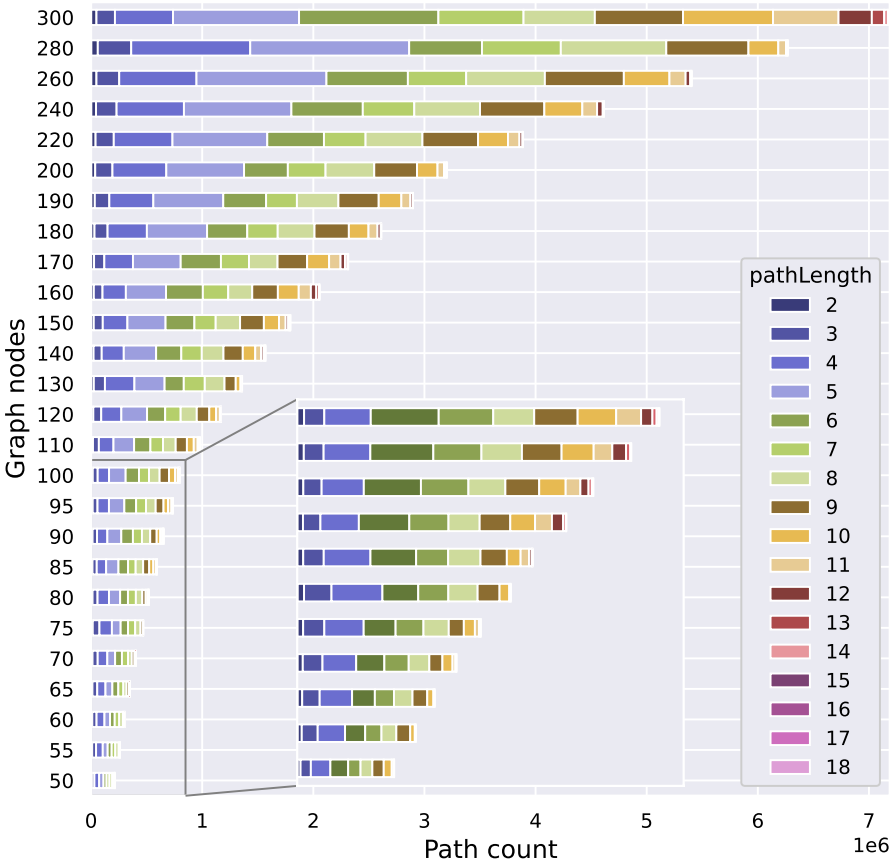
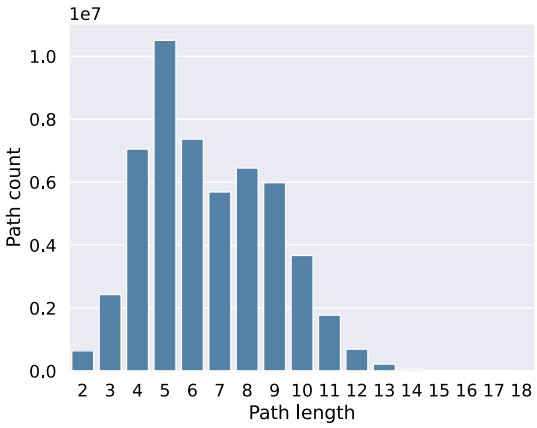


Fig. 9 Number of samples per graph size, and proportion of path lengths per graph size in validation dataset

Fig. 10 Number of samples per-path length in validation dataset



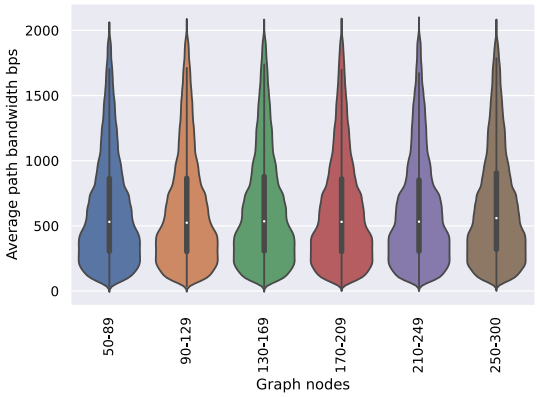


Fig. 11 Average path bandwidth distribution per graph size in validation dataset

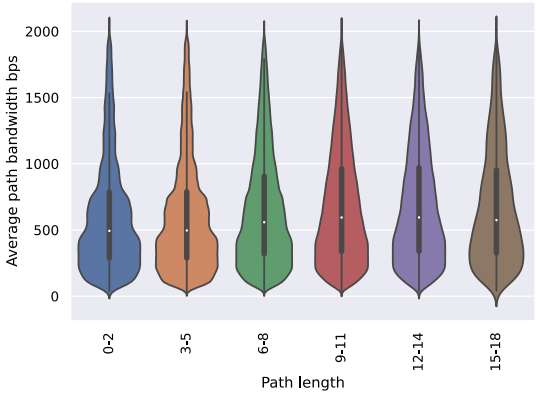


Fig. 12 Average path bandwidth distribution per path-length in validation dataset

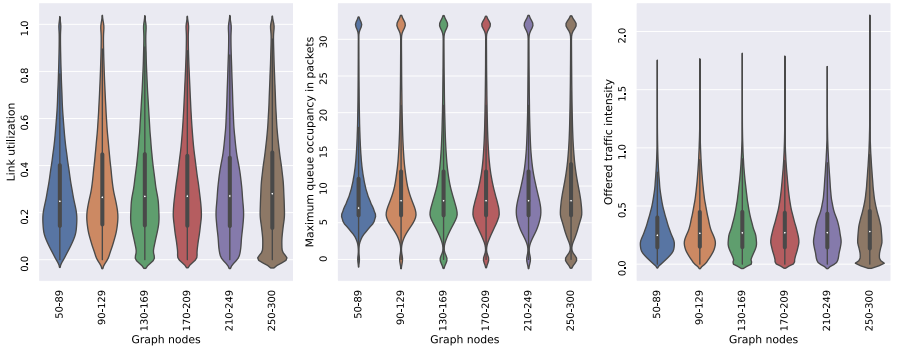


Fig. 13 Utilization, maximum queue occupancy and O_t values distributions compared to graph size in validation dataset

Fig. 17. On the other hand, the quantity of paths in S_1 is not regular as depicted in Fig. 15.

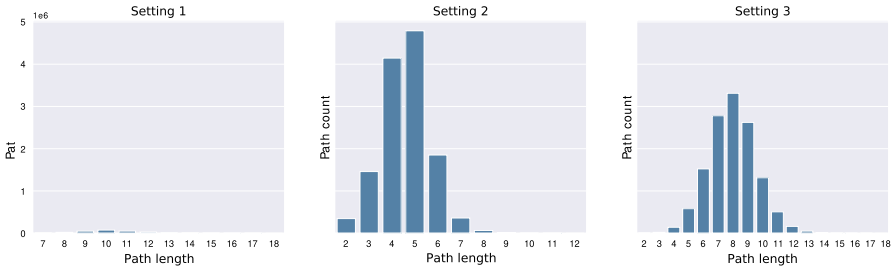


Fig. 14 Per-path length, number of samples in test dataset

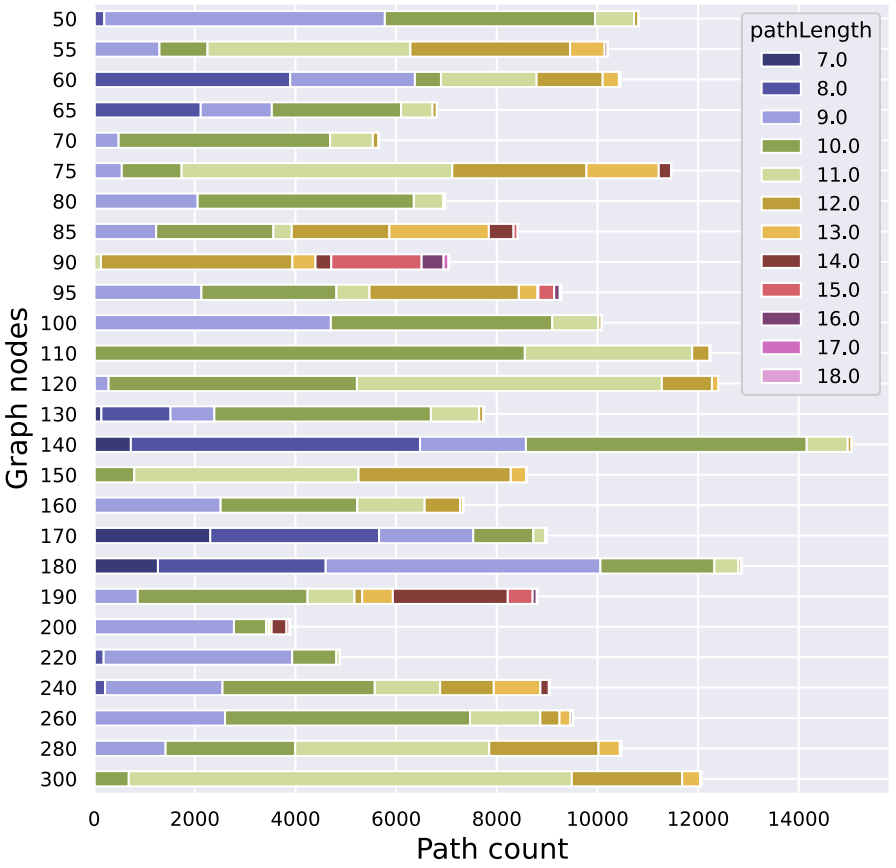


Fig. 15 Number of samples per graph size, and proportion of path lengths per graph size in test dataset, S_1

The paths mainly have a length between 9 and 11 in S_1, between 3 and 6 in S_2 and between 6 and 10 in S_3, as shown in Fig. 14. However, S_1 has significantly fewer path samples than the other settings.

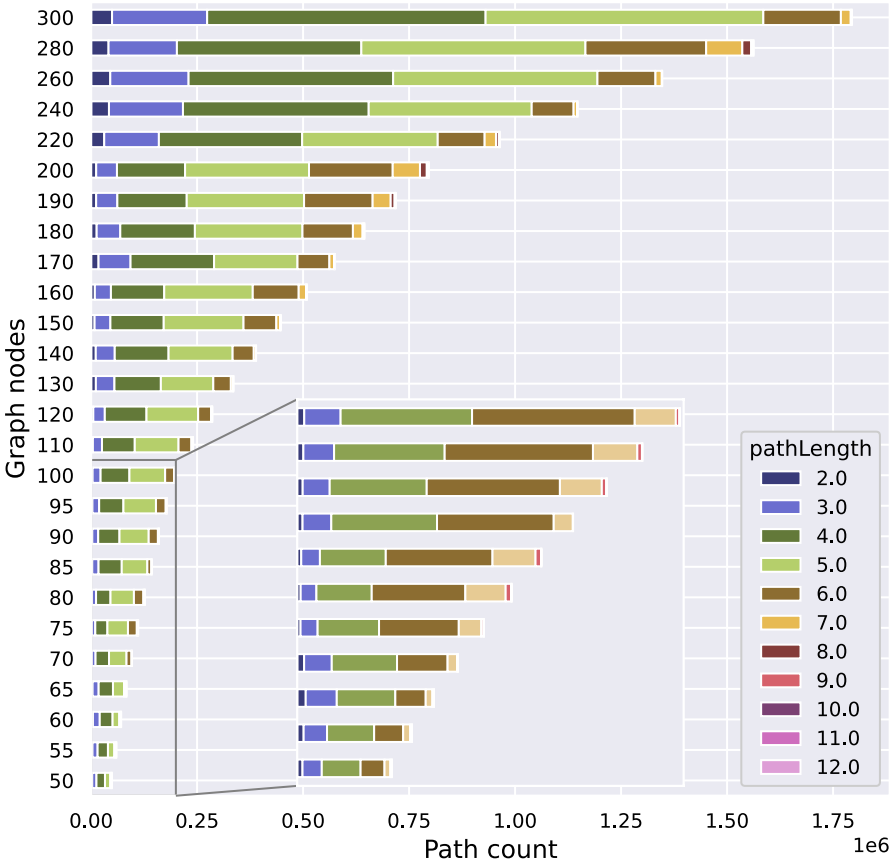


Fig. 16 Number of samples per graph size, and proportion of path lengths per graph size in test dataset, S_2

None of the datasets were modified, as the testing and validation datasets were already balanced. The subsets S_2 and S_3 of the testing dataset were also found to be balanced. On the other hand, the S_1 test dataset had very few samples and with higher deviation. These features were related to the link properties, including link utilization, maximum queue occupancy, and O_t, as shown in Fig. 18. These features were out of the training and validation distributions, mainly in the graph samples with a higher number of nodes. The cause of this deviation was that not all the source-destination paths transmitted traffic, only the particularly long ones. These characteristics lead to slightly higher errors in the S_1 predictions, as described in Sect. 7, compared to S_2 and S_3. However, these errors were specific to this small subset designed for particular testing purposes (i.e., longer path testing with lower traffic).

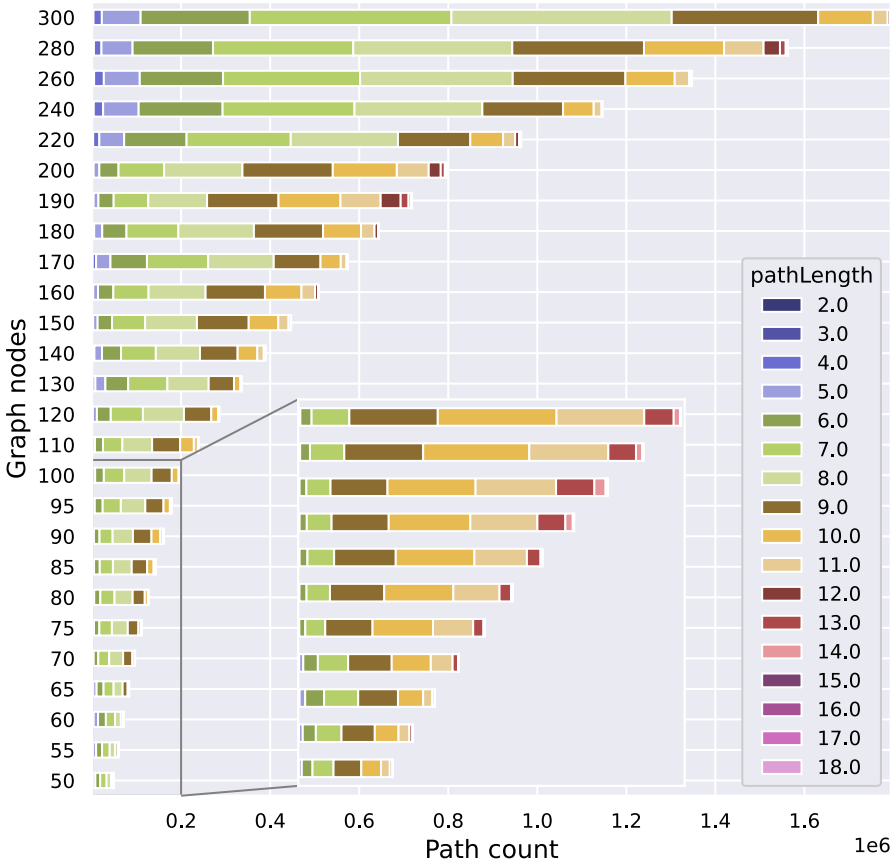


Fig. 17 Number of samples per graph size, and proportion of path lengths per graph size in test dataset, S_3

6 Improving RouteNet for Generalization

In this section, the improvements to the RouteNet baseline implemented in TensorFlow [47] 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.

6.1 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

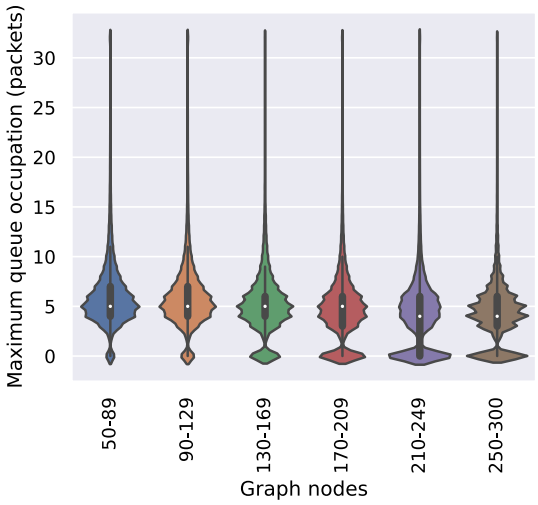


Fig. 18 Maximum queue occupancy distribution per each graph size in S_1

distribution but has to predict values that follow another distribution, which is problematic for NNs [56].

The initial idea to solve this problem consisted in finding an indirect metric that keeps a similar distribution among all the datasets and can be used, followed by a post-processing step, to predict the metric of path delay with low error. Based on this idea, our first approach was to use the occupancy of a link (Q_o) as an indirect metric, representing the average utilization of a queue, as 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. 19. 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, as shown in Eq. (2).

Each queue delay of this flow was calculated using 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.

Table 2 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

Table 3 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

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

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

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 %.

6.2 Normalization of Predictor Features (GAIN-2 Solution)

After analyzing the training and validation datasets, it was observed that the data distributions of the scalar features used in the model (detailed in Table 3) were different for training, compared to the validation and test subsets, as shown in Table 2. 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

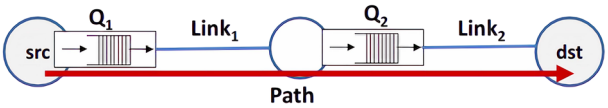


Fig. 19 Queue occupancy example to infer delay

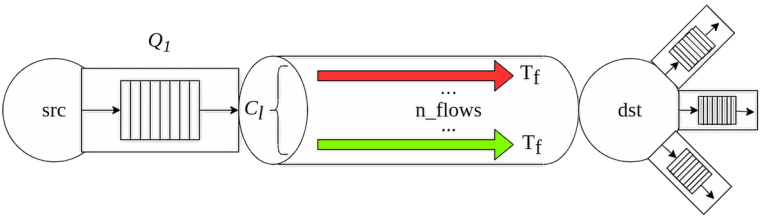


Fig. 20 Offered traffic intensity example of flows in a link

equals 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.

6.3 Feature Selection (GAIN-3 Solution)

The features included in the baseline were T_f and C_l, as described in Table 3. 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. Furthermore, this feature had value ranges very similar in the three datasets, enhancing generalization. Consequently, it was added to the GAIN-3 path state, also including the min-max normalization preprocessing. The MAPE of the GAIN-3 solution was 18.471 %.

6.4 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. Figure 20 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.

Table 4 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

The dependence between T_f demand in a link and the queuing delay [57], 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 T_f of all 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 O_t. The resulting MAPE of the GAIN-4 solution was 2.612 %.

6.5 Hyperparameters Optimization (GAIN-5 Solution)

The hyperparameters, i.e., number of neurons for the path-state and link-state, readout units, message-passing iterations, and 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 4. For each combination of hyperparameters, a new training was performed followed by a check of the MAPE result.

One of the effects of optimizing the hyperparameters 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 hyperparameter tests.

After hyperparameter tuning, the GAIN-5 model was trained during seven epochs. We found that reducing the dimensions of the link state and the path state

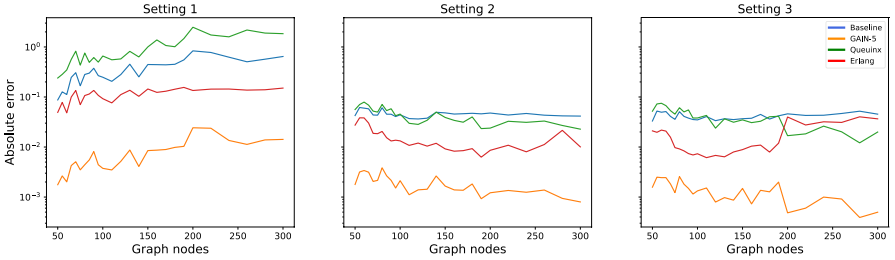


Fig. 21 Baseline, Queuinx, RouteNet-Erlang and GAIN-5 logarithmic absolute errors compared to graph size

and increasing the readout units along the iterations of the message passing procedure significantly improved the results 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 hyperparameters and epochs at a low value is the reduced Graphics Processing Unit (GPU) power consumption, temperature, compute, and memory utilization as exposed in Sect. 7.

7 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. In addition, a dataset with real traces of traffic is included for further evaluation.

7.1 Additional Models for Evaluation

For further analysis and comparison, two models are added to the predictions phase. Apart from the RouteNet baseline and the GAIN-5, two other delay prediction models are used: RouteNet-Erlang [58] and Queuinx [59].

RouteNet-Erlang also uses RouteNet as a baseline, applying improvements to increase the prediction accuracy in larger networks (≈10x), longer paths, and larger link capacities, using queue status to improve the prediction accuracy. The model is trained with the same dataset used with the baseline and GAIN-5.

Queuinx is a model that uses queuing theory, being focused on finite queues, using a fixed point algorithm. This model is used to validate the RouteNet [15] results in comparison to queuing theory computations. Both models are used to predict the test dataset and are included in the analysis of the results.

Table 5 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	12 h 15 m	~19%
GAIN 1	44.73	13.074	11.705	13.108	54.318	16.214	42.374	67.44	70.994	90.739	9 h 45 m	~21%
GAIN 2	28.739	11.719	11.026	11.864	35.067	17.092	39.773	31.754	17.581	31.353	9 h 48 m	~19%
GAIN 3	18.471	9.436	6.893	12.468	26.897	22.106	30.067	18.143	12.569	21.862	9 h 40 m	~15%
GAIN 4	2.612	2.652	1.539	3.687	2.492	1.386	2.567	2.584	1.607	2.363	3 h 25 m	~4%
GAIN 5	1.838	1.407	1.111	1.808	1.929	1.573	1.535	1.756	1.388	1.462	2 h 20 m	~15%

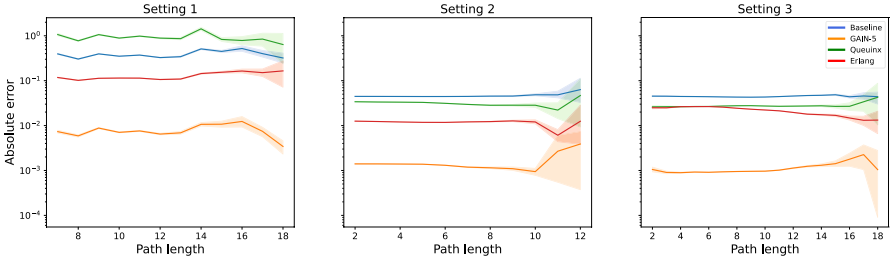


Fig. 22 Baseline, Queuinx, RouteNet-Erlang and GAIN-5 logarithmic absolute error compared to path length

7.2 Models Evaluation with Test Dataset

The gradual improvement in each step is shown in Table 5, where the RouteNet baseline is shown to have a bad generalization for larger graphs (as an example, 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. The MAPE was calculated separately for each column and row in Table 5. The GAIN solutions gradually reduced MAPE, applying improvements step by step as described in Sect. 6, using the three settings with the graph sizes 50 and 300 as a reference. From the RouteNet Baseline and GAIN-1 solution, which was the first change over the baseline, until the GAIN-5 solution, we achieved an improvement by a factor of 24 (from 44.73 % to 1.838%) and by a factor of 101 (from 187.28% to 1.828%) lower MAPE in all settings, respectively.

The improvement between the baseline and GAIN-5 can be compared in Fig. 21, where the subsets S_1, S_2 and S_3 are represented. The absolute error is represented in a logarithmic scale for comparison reasons. For GAIN-5, the error is 32x smaller than in the baseline. GAIN-5 is compared to the baseline in terms of graph size and path length, two features of a graph that scale up when the sample is bigger, as shown in Sect. 5. It can be observed that in setting S_2 and S_3, the tendency of the prediction error is to become lower when the graph increases the number of nodes, improving the generalization compared to the baseline.

As introduced in the test analysis in Sect. 5.3, the S_1 subset had a lower number of samples including features with high deviation compared to S_2 and S_3. These characteristics lead to increased errors in the S_1 predictions, both in baseline and GAIN-5 implementations, but still much lower in GAIN-5. The small datasets make it hard to rigorously evaluate GNNs, as they are data-hungry models [60], requiring significant amounts of data to check the correctness of the prediction accuracy. This is the main problem in analyzing the results of S_1, where the high deviation paired with a low amount of samples resulted in more variability in the accuracy of the predictions.

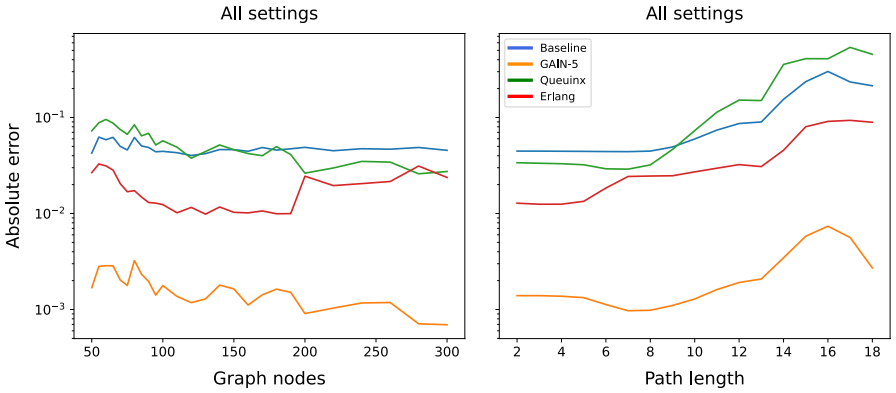


Fig. 23 Baseline, Queuinx, RouteNet-Erlang and GAIN-5 all settings, logarithmic absolute errors compared to path length

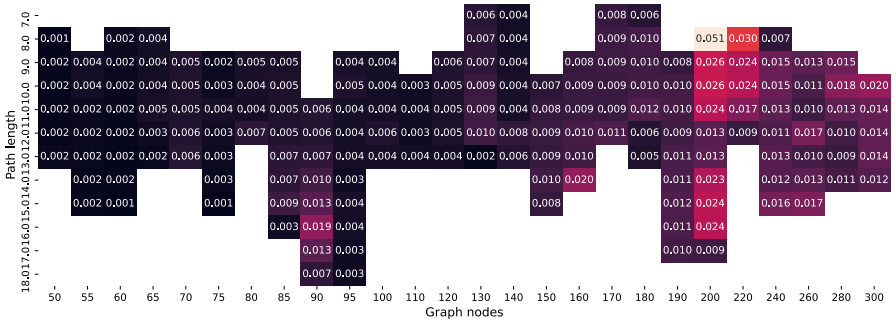


Fig. 24 GAIN-5 absolute error on S_1 compared to path length

On the other hand, the predictions on S_2 with higher capacity links and on S_3, with long paths mixed with higher capacity links, resulted in a low overall error for all the graph sizes, becoming lower as the graph was bigger. As shown on the error scales, the absolute error variability range on the GAIN-5 solution on all settings is much smaller than in the baseline results.

Also in Fig. 21 the prediction results for the Queuinx and RouteNet-Erlang models are shown. Queuinx reveals similar results compared to the baseline. It can be concluded that both models exhibit a higher error than GAIN-5, with RouteNet-Erlang being better than the challenge baseline in every setting. Comparing the errors linked to the path lengths in Fig. 22, it can be observed that with longer paths, the error deviation is larger. This is due to the low amount of samples and their particularity in the S_1 of the dataset. In Fig. 23 this problem is shown to have a small impact on the solution precision. On the left, all settings are mixed to obtain the average absolute error per each graph size. GAIN-5 is shown to have a lower error on all settings mixed in comparison to the other tested models, lowering the error as

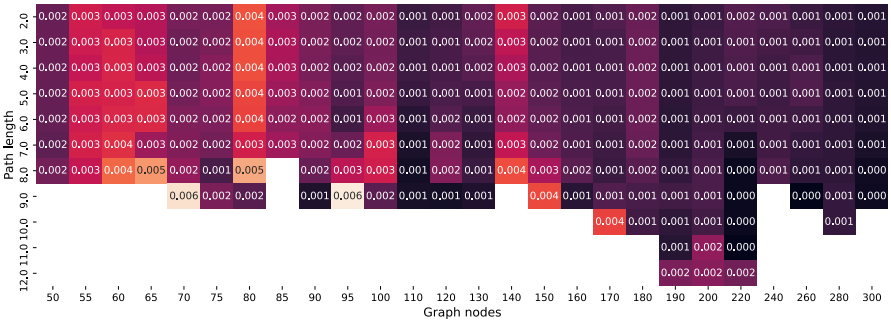


Fig. 25 GAIN-5 absolute error on S_2 compared to path length

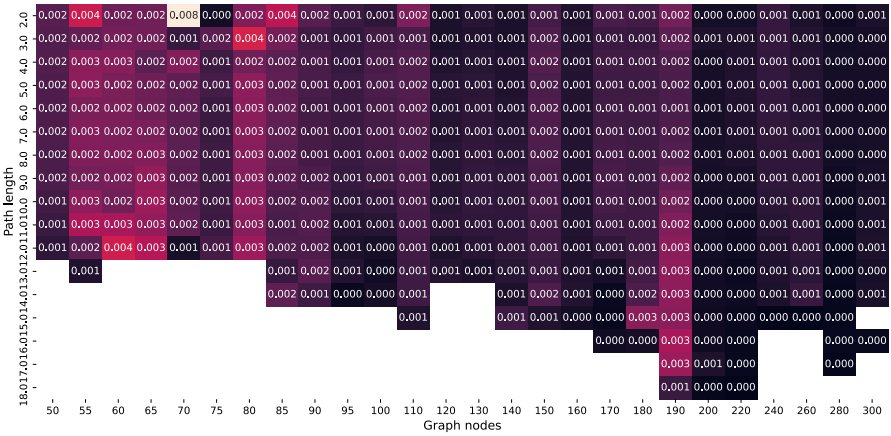


Fig. 26 GAIN-5 absolute error on S_3 compared to path length

the graph size increased. On the right, all settings compared to the path length are checked, where a higher error with the out-of-distribution samples is observed, as a small number of samples were above 15 hops.

In a further error analysis to confirm the generalization of the model, heat maps for the three settings were generated. In Fig. 24, S_1 was found to scale slightly worse than the other settings as detected in the previous analysis, especially with the graphs of 200 and 220 nodes. Inversely, Fig. 25 and Fig. 26 show the generalization of the model, keeping the errors low and even lowering the absolute errors as the networks become bigger.

7.3 Models Evaluation with Real Traffic Traces Dataset

To validate the accuracy of the tested models, a real traffic dataset is also used. This real traces dataset [61] is generated using real-world traffic matrices and realistic packet inter-arrival times.

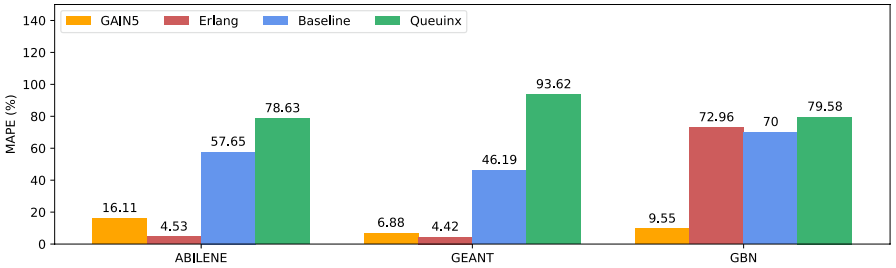


Fig. 27 Baseline, GAIN-5, Queuinx and RouteNet-Erlang MAPE results with a real traces dataset

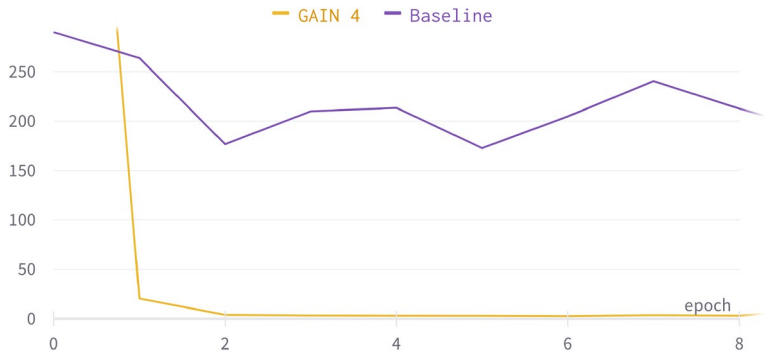


Fig. 28 Validation MAPE of the RouteNet baseline vs the GAIN-4 solution

Table 6 Comparison of the best solutions for the BNN-UPC challenge

Solution properties	GAIN 5	PARANA	SOFGNN
RouteNet baseline	X		X
Hyperparameter tune	X	X	X
Fast train/execution	X		X
Feature engineering	X	X	X
Reimplemented baseline		X	
Message passing redesign		X	
Multiple models		X	X
Data augmentation			X
MAPE (%)	1.838	1.267	1.389

This dataset contains three different topologies: ABILENE, GEANT, and GBN [62]. These topologies are not present in the training dataset, and traffic matrices are completely different from the ones used in training. The topologies have fewer nodes than the training dataset, with 11, 22 and 17 nodes respectively. The models used in the previous test dataset evaluation phase are executed without retraining them, predicting the values directly.

In Fig. 27 the MAPE results for each graph and model are shown. For ABILENE and GEANT networks, the results are similar, with RouteNet-Erlang achieving the best results, followed closely by GAIN-5. Meanwhile, testing the models with the GBN network, which contains smaller links and lower traffic, the results are clearly better using GAIN-5, which demonstrates a better generalization maintaining a lower overall error on the three scenarios.

7.4 Resource Utilization

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 illustrated in Fig. 28. In GAIN-5 the GPU consumption is higher than in GAIN-4, because of the higher hyperparameters, 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 h 30 min, 2 h 20 min for training, which is a significant reduction compared to the original 12 h training of the baseline.

Finally, to compare GAIN-5 with other solutions, Table 6 provides a summary of the best different approaches and MAPE results. The uniqueness of our solution (GAIN-5) [63] is the preservation of the architecture and data of the original RouteNet model, as the message passing procedure was not modified and data augmentation was not used, resulting in an implementation with lower overhead than in the other solutions. Additionally, our approach did not use the average of multiple models to maintain 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 the predictions. GAIN-5 demonstrates that it is possible to obtain a low MAPE utilizing a simpler approach, just improving the original RouteNet model, and avoiding the usage of multiple tuned models. This simplicity allows fast training and easier management of the model, enhancing its usage in new environments by just retraining and deploying it. No other solution in the BNN-UPC challenge has fast training and good precision without using multiple models. The fine-tuning of GNN hyperparameters set smaller in our solution, allowed lower training and prediction times, and less memory and power usage, thus making our solution more suitable in case of frequent retraining requirements.

8 Conclusions and Future Work

The network models based on GNNs are being suggested as a good approach for building an autonomous 5G and B5G networks, as they can both properly forecast network KPIs and easily be modified and updated to account for the current networks rising complexity and dynamism. Out-of-the-box GNNs are unable to

generalize and scale to larger topologies, which causes their accuracy to decline. In this paper, we introduced GAIN, a GNN-based model that accurately forecasts the average per-path network delay. The good generalization of the model, tested with heterogeneous datasets, enables it for predicting per-path delay in many scenarios, thereby proving it useful for managing 5G and B5G services.

In order to achieve good generalization, GAIN builds on RouteNet's GNN and makes a number of additions. These additions include the inference of per-path delay from the predicted link queue occupancy, feature normalization, feature selection, feature engineering (offered traffic intensity), and hyperparameter optimization. Additionally, compared to other solutions currently available, GAIN has less implementation complexity, lower resource requirements, and faster training.

A closer analysis, filtering, and preparation of the datasets is key for achieving improved results in a ML approach. In this case, the training and validation datasets were well generated for the training task. The testing dataset had particularities discovered during the test phase, where a closer analysis helped to know the cause of the slightly higher errors in a specific case.

As part of our ongoing research, we intend to examine additional GNN architectures, frameworks, and performance improvements. We also intend to use these developments in the suggested or another closed-loop control context, where GNN predictions will be used by decision-making algorithms (such as traffic admission control) deployed in 5G and B5G networks.

Author Contributions Conceptualization, all authors; methodology, all authors; software, M.F. and P.S.; writing-original draft preparation, M.F., P.S. and M.C.; writing-review and editing, all authors; All authors have read and agreed to the published version of the manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This project has received funding from the Generalitat de Catalunya through the 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 program under grant agreement no. 101017109 'DAEMON'.

Availability of data and materials The model developed in this study is openly available in GitHub [63]. The data that support the findings of this study is openly available in [47].

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical Approval Not applicable.

References

1. 5GPPP: AI and ML - Enablers for Beyond 5G Networks. (2021). https://doi.org/10.5281/zenodo.4299895
2. Li, X., Samaka, M., Chan, H.A., Bh, D., Gupta, L., Guo, C., Jain, R.: Network slicing for 5g: challenges and opportunities. IEEE Internet Comput. (2018). https://doi.org/10.1109/MIC.2018.326150452
3. Chang, C.Y., Ruiz, T.G., Paolucci, F., Jimenez, M.A., Sacido, J., Papagianni, C., Ubaldi, F., Scano, D., Gharbaoui, M., Giorgetti, A., Valcarenghi, L., Tomakh, K., Boddi, A., Caparros, A., Pergolesi, M., Martini, B.: Performance isolation for network slices in industry 4.0: the 5growth approach. IEEE Access (2021). https://doi.org/10.1109/ACCESS.2021.3135827
4. Poltronieri, F., Stefanelli, C., Suri, N., Tortonesi, M.: Value is king: the mecforge deep reinforcement learning solution for resource management in 5g and beyond. J. Netw. Syst. Manag. (2022). https://doi.org/10.1007/s10922-022-09672-6
5. Xu, X., Yao, L., Bilal, M., Wan, S., Dai, F., Choo, K.-K.R.: Service migration across edge devices in 6g-enabled internet of vehicles networks. IEEE Internet Things J. **9**(3), 1930–1937 (2022). https://doi.org/10.1109/JIOT.2021.3089204
6. Xu, Z., Tang, J., Meng, J., Zhang, W., Wang, Y., Liu, C.H., Yang, D.: Experience-driven networking: a deep reinforcement learning based approach. In: IEEE INFOCOM 2018-IEEE (2018). IEEE
7. Soto, P., Camelo, M., Mets, K., Wilhelmi, F., Góez, D., Fletscher, L.A., Gaviria, N., Hellinckx, P., Botero, J.F., Latré, S.: Atari: a graph convolutional neural network approach for performance prediction in next-generation wlans. Sensors **21**(3), 4321 (2021)
8. Moreira, R., Silva, F.: Towards 6g network slicing, pp. 25–30 (2021). https://doi.org/10.5753/w6g.2021.17231
9. Nguyen, H.X., Trestian, R., To, D., Tatipamula, M.: Digital twin for 5g and beyond. IEEE Commun. Mag. **59**(2), 10–15 (2021). https://doi.org/10.1109/MCOM.001.2000343
10. Bushnaq, O.M., Zhilin, I.V., Masi, G.D., Natalizio, E., Akyildiz, I.F.: Automatic network slicing for admission control, routing, and resource allocation in underwater acoustic communication systems. IEEE Access **10**, 134440–134454 (2022). https://doi.org/10.1109/ACCESS.2022.3231607
11. Dong, T., Zhuang, Z., Qi, Q., Wang, J., Sun, H., Yu, F.R., Sun, T., Zhou, C., Liao, J.: Intelligent joint network slicing and routing via gcn-powered multi-task deep reinforcement learning. IEEE Trans. Cogn. Commun. Netw. **8**(2), 1269–1286 (2022). https://doi.org/10.1109/TCCN.2021.3136221
12. Hornik, K., Stinchcombe, M., White, H., et al.: Multilayer feedforward networks are universal approximators. Neural Netw. **5**, 359–366 (1989)
13. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **1**, 61–80 (2009). https://doi.org/10.1109/TNN.2008.2005605
14. Garg, V., Jegelka, S., Jaakkola, T.: Generalization and representational limits of graph neural networks. In: International Conference on Machine Learning, pp. 3419–3430 (2020). PMLR
15. Rusek, K., Suarez-Varela, J., Almasan, P., Barlet-Ros, P., Cabellos-Aparicio: RouteNet: leveraging graph neural networks for network modeling and optimization in SDN. IEEE J. Sel. Areas Commun. (10), 2260–2270 (2020) arXiv:1910.01508. https://doi.org/10.1109/JSAC.2020.3000405
16. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V.F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, H.F., Ballard, A.J., Gilmer, J., Dahl, G.E., Vaswani, A., Allen, K.R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R.: Relational inductive biases, deep learning, and graph networks. CoRR (2018) arXiv:1806.01261

17. Almasan, P., Ferriol-Galmés, M., Paillisse, J., Suárez-Varela, J., Perino, D., López, D., Perales, A.A.P., Harvey, P., Ciavaglia, L., Wong, L., Ram, V., Xiao, S., Shi, X., Cheng, X., Cabellos-Aparicio, A., Barlet-Ros, P.: Digital twin network: opportunities and challenges (2022)
18. Zhou, C., Yang, H., Duan, X., Lopez, D., Pastor, A., Wu, Q., Boucadair, M., Jacquenet, C.: Digital twin network: Concepts and reference architecture. Internet draft, Internet Engineering Task Force (July 2022). Work in Progress. https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/01/
19. Ahmadi, H., Nag, A., Khar, Z., Sayrafian, K., Rahardja, S.: Networked twins and twins of networks: an overview on the relationship between digital twins and 6g. IEEE Commun. Stand. Mag. **5**(4), 154–160 (2021). https://doi.org/10.1109/MCOMSTD.0001.2000041
20. Khan, L.U., Saad, W., Niyato, D., Han, Z., Hong, C.S.: Digital-twin-enabled 6g: vision, architectural trends, and future directions. IEEE Commun. Mag. **60**(1), 74–80 (2022). https://doi.org/10.1109/MCOM.001.21143
21. Ak, E., Duran, K., Dobre, O.A., Duong, T.Q., Canberk, B.: T6conf: digital twin networking framework for ipv6-enabled net-zero smart cities. IEEE Commun. Mag. **61**(3), 36–42 (2023). https://doi.org/10.1109/MCOM.003.2200315
22. Khan, L.U., Mustafa, E., Shuja, J., Rehman, F., Bilal, K., Han, Z., Hong, C.S.: Federated learning for digital twin-based vehicular networks: architecture and challenges. IEEE Wirel. Commun. (2023). https://doi.org/10.1109/MWC.012.2200373
23. Goswami, P., Mukherjee, A., Hazra, R., Yang, L., Ghosh, U., Qi, Y., Wang, H.: Ai based energy efficient routing protocol for intelligent transportation system. IEEE Trans. Intell. Transport. Syst. **23**(2), 1670–1679 (2022). https://doi.org/10.1109/TITS.2021.3107527
24. Ghosh, S., Dagiuklas, T., Iqbal, M., Wang, X.: A cognitive routing framework for reliable communication in iot for industry 5.0. IEEE Trans. Ind. Inform. **18**(8), 5446–5457 (2022). https://doi.org/10.1109/TII.2022.3141403
25. Kumbhar, F.H., Shin, S.Y.: Novel vehicular compatibility-based ad hoc message routing scheme in the internet of vehicles using machine learning. IEEE Internet Things J. **9**(4), 2817–2828 (2022). https://doi.org/10.1109/JIOT.2021.3093545
26. Gheibi, O., Weyns, D., Quin, F.: Applying machine learning in self-adaptive systems: a systematic literature review. ACM Trans. Auton. Adapt. Syst. **3**, 1–37 (2021)
27. Ciucu, F., Schmitt, J.: Perspectives on network calculus - no free lunch, but still good value. Comput. Commun. Rev. (2012). https://doi.org/10.1145/2342356.2342426
28. Bylina, B., Bylina, J.: Using markov chains for modelling networks. Annales UMCS Informatica AI, 27–34 (2005)
29. Weingartner, E., vom Lehn, H., Wehrle, K.: A performance comparison of recent network simulators. In: 2009 IEEE International Conference on Communications, pp. 1–5 (2009). https://doi.org/10.1109/ICC.2009.5198657
30. Bega, D., Gramaglia, M., Fiore, M., Banchs, A., Costa-Perez, X.: Deepcog: cognitive network management in sliced 5g networks with deep learning. In: Proceedings—IEEE INFOCOM (2019). https://doi.org/10.1109/INFOCOM.2019.8737488
31. Bega, D., Gramaglia, M., Fiore, M., Banchs, A., Costa-Perez, X.: Aztec: anticipatory capacity allocation for zero-touch network slicing. In: IEEE INFOCOM 2020—IEEE Conference on Computer Communications, pp. 794–803 (2020). https://doi.org/10.1109/INFOCOM41043.2020.9155299
32. Khatouni, A.S., Soro, F., Giordano, D.: A machine learning application for latency prediction in operational 4g networks. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 71–74 (2019)
33. Gao, Z.: 5g traffic prediction based on deep learning. Comput. Intell. Neurosci. (2022). https://doi.org/10.1155/2022/3174530
34. Abdellah, A.R., Mahmood, O.A., Kirichek, R., Paramonov, A., Koucheryavy, A.: Machine learning algorithm for delay prediction in iot and tactile internet. Future Internet (2021). https://doi.org/10.3390/fi13120304
35. Sato, R.: A survey on the expressive power of graph neural networks. CoRR (2020) arXiv:2003.04078
36. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. CoRR (2016) arXiv:1611.09940
37. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V.F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, H.F., Ballard, A.J.,

Gilmer, J., Dahl, G.E., Vaswani, A., Allen, K.R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M.M., Vinyals, O., Li, Y., Pascanu, R.: Relational inductive biases, deep learning, and graph networks. CoRR (2018) arXiv:1806.01261

38. Gammelli, D., Yang, K., Harrison, J., Rodrigues, F., Pereira, F.C., Pavone, M.: Graph neural network reinforcement learning for autonomous mobility-on-demand systems. In: Proceedings of the 60th IEEE Conference on Decision and Control, CDC 2021, pp. 2996–3003. Institute of Electrical and Electronics Engineers Inc., United States (2021). https://doi.org/10.1109/CDC45484.2021.9683135
39. Rkhami, A., Hadjadj-Aoul, Y., Rubino, G., Outtagarts, A.: Mongnn: A neuroevolutionary-based solution for 5g network slices monitoring. In: Proceedings–Conference on Local Computer Networks, LCN, 185–192 (2021). https://doi.org/10.1109/LCN52139.2021.9524880
40. Wang, H., Wu, Y., Min, G., Miao, W.: A graph neural network-based digital twin for network slicing management. IEEE Trans. Ind. Inform. (2020). https://doi.org/10.1109/TII.2020.3047843
41. Ge, Z., Hou, J., Nayak, A.: Gnn-based end-to-end delay prediction in software defined networking. In: 2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 372–378 (2022). https://doi.org/10.1109/DCOSS54816.2022.00066
42. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: datasets for machine learning on graphs. Adv. Neural Inf. Process. Syst. **33**, 22118–22133 (2020)
43. PARANA.: Improved GNN Generalization to Larger 5G Networks By Fine-Tuning Predictions From Queueing Theory. Federal University of São Paulo. [Online; accessed 26-01-2022] (2021). https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-001-PARANA
44. Klaus, B., Afonso, A., Berton, L.: Qt-routenet: Improved gnn generalization to larger 5g networks by fine-tuning predictions from queuing theory. ITU J. Future Evol. Technol. (2022)
45. SOFGNN.: Concordia University and Computer Research Institute of Montreal. (2021). https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-001-SOFGNN-Graph-Neural-Networking-Challenge. Accessed 26 Jan 2022
46. Galmés, M.F., Suárez-Varela, J., Rusek, K., Barlet-Ros, P., Cabellos-Aparicio, A.: Scaling graph-based deep learning models to larger networks. CoRR (2021)
47. BNN: RouteNet - Graph Neural Networking challenge 2021. (2021). https://github.com/BNN-UPC/GNNetworkingChallenge/tree/2021_Routenet_TF. Accessed 31 Jan 2022
48. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. CoRR (2017)
49. Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. CoRR (2015)
50. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **8**, 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
51. Murphy, R.L., Srinivasan, B., Rao, V.A., Ribeiro, B.: Relational pooling for graph representations. CoRR (2019)
52. Jure Leskovec: CS224W: Machine Learning with Graphs (2021). http://cs224w.stanford.edu. Accessed 17 Sept 2022
53. ITU: AI for Global Summit. ITU AI/ML in 5G Challenge: Graph Neural Networking Challenge 2021. https://aiforgood.itu.int/about/aiml-in-5g-challenge/. Accessed 19 Jan 2022
54. BNN.: Dataset description. (2021). https://bnn.upc.edu/challenge/gnnet2021/dataset/. Accessed 26 Oct 2022
55. Borovicka, T., Jr., M.J., Kordik, P., Jirina, M.: Selecting representative data sets. In: Karahoca, A. (ed.) Advances in Data Mining Knowledge Discovery and Applications. IntechOpen, Rijeka (2012). Chap. 2. https://doi.org/10.5772/50787
56. Hendrycks, D., Dietterich, T.G.: Benchmarking neural network robustness to common corruptions and perturbations. CoRR (2019)
57. Liebeherr, J., Ghiassi-Farrokhfal, Y., Burchard, A.: On the impact of link scheduling on end-to-end delays in large networks. IEEE J. Sel. Areas Commun. **5**, 1009–1020 (2011). https://doi.org/10.1109/JSAC.2011.110511
58. Ferriol-Galmés, M., Rusek, K., Suárez-Varela, J., Xiao, S., Cheng, X., Barlet-Ros, P., Cabellos-Aparicio, A.: Routenet-erlang: A graph neural network for network performance evaluation. arXiv preprint arXiv:2202.13956 (2022)

59. Rusek, K.: GitHub - Queuinx: A library for performance evaluation in Jax. https://github.com/krzysztofrusek/queuinx. Accessed 17 Jun 2023
60. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? CoRR (2018)
61. BNN-UPC.: NetworkModelingDatasets - datasets v4. https://github.com/BNN-UPC/NetworkModelingDatasets/tree/master/datasets_v4. Accessed 17 Jun 2023
62. Ferriol-Galmés, M., Suárez-Varela, J., Paillissé, J., Shi, X., Xiao, S., Cheng, X., Barlet-Ros, P., Cabellos-Aparicio, A.: Building a digital twin for network optimization using graph neural networks. Comput. Netw. **217**, 109329 (2022). https://doi.org/10.1016/j.comnet.2022.109329
63. GAIN: GAIN 5 GitHub code. Girona-Antwerp Intelligence for Networking. (2022). https://github.com/mfarreras/gain-gnn. Accessed 10 Oct 2022

Miquel Farreras received the bachelor's degree in computer science engineering and the master's degree in computer science from the University of Girona in 2017 and 2019 respectively. He has been an Associate Professor in computer science with UdG in 2020. He is currently pursuing the Ph.D. degree with the University of Girona, Catalonia, Spain. His research interests include graph theory, 5G network applications and technologies, and machine learning applied to networks.

Paola Soto is a Ph.D. researcher at the University of Antwerp-imec. She received her B.Sc. in Electronics and her M.Sc. in Telecommunications Engineering from the University of Antioquia, Colombia, in 2014 and 2018, respectively. Her current research is focused on developing network management strategies using artificial intelligence and machine learning.

Miguel Camelo received a master's degree in systems and computer engineering (University of Los Andes, Colombia, 2010) and a Ph.D. degree in computer engineering (University of Girona, Spain, 2014). He has authored several papers in international conferences/journals. He is a Senior Researcher at the University of Antwerp-imec, Belgium, where he leads the research on applied artificial intelligence (AI) in networking. His research interests are in the field of applied AI in communication networks.

Lluís Fàbrega received the bachelor's degree in telecommunications engineering and the master's degree in mobile communications from the Polytechnical University of Catalonia, in 1995 and 1996, respectively, and the Ph.D. degree in computer science from the University of Girona (UdG) in 2008. He has been an Associate Professor in computer science with UdG since 2008. He has co-authored several papers in journals and international conferences and participated in several Spanish and European research projects. His research interests include the design and performance evaluation of routing, traffic engineering, and quality of service mechanisms in the Internet and in connection oriented network technologies.

Pere Vilà received the degree in computer science engineering from the Polytechnic University of Catalonia in 1997 and the Ph.D. degree in computer science from the University of Girona in 2004. He is currently an Associate Professor in computer science with the University of Girona. He has authored or co-authored around 50 papers in his areas of interest. His research interests include new routing algorithms for future Internet, network protection and robustness, complex networks, network management, and control. He served as a member for program committee in several international conferences and an Associate Editor for the International Journal of Communication Systems.

Authors and Affiliations

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

✉ Miquel Farreras
miquel.farreras@udg.edu

Paola Soto
paola.soto-arenas@uantwerpen.be

Miguel Camelo
miguel.camelo@uantwerpen.be

Lluís Fàbrega
lluis.fabrega@udg.edu

Pere Vilà
pere.vila@udg.edu

[1] Institute of Informatics and Applications, University of Girona, Girona, Spain

[2] IDLab, University of Antwerp, in collaboration with imec, Antwerp, Belgium