

Network Calculus Boosts Open World Learning Capability for Graph Convolution in Routing Networks

Yifei Jin

KTH Royal Institute of Technology & Ericsson AB
Division of Theoretical and Computer Science & Ericsson Research
 Stockholm, Sweden
 yifeij@kth.se

Marios Daoutis

Ericsson AB
Ericsson Research
 Stockholm, Sweden
 marios.daoutis@ericsson.com

Sarunas Girdzijauskas

KTH Royal Institute of Technology
Division of Software and Computer Systems
 Stockholm, Sweden
 sarunasg@kth.se

Aristides Gionis

KTH Royal Institute of Technology
Division of Theoretical and Computer Science
 Stockholm, Sweden
 argioni@kth.se

Abstract—Accurate routing network status estimation is a key component in Software Defined Networking. However, existing deep-learning-based methods for modeling network routing are not able to extrapolate towards unseen feature distributions. Nor are they able to handle scaled and drifted network attributes in test sets that include open-world inputs. To deal with these challenges, we propose a novel approach for modeling network routing, using Graph Neural Networks. Our method can also be used for network-latency estimation. Supported by a domain-knowledge-assisted graph formulation, our model shares a stable performance across different network sizes and configurations of routing networks, while at the same time being able to extrapolate towards unseen sizes, configurations, and user behavior. We show that our model outperforms most conventional deep-learning-based models, in terms of prediction accuracy, computational resources, inference speed, as well as ability to generalize towards open-world input.

Index Terms—Graph Convolution, Software Define Networks, Open World Learning

I. INTRODUCTION

Software-defined networking (SDN) is a state-of-the-art approach to network management, which permits dynamic and programmatic network configurations (e.g., routing), aimed at improved network performance and monitoring, abstracted away from individual network elements into a centralized network control layer. Consequently, the orchestration and SDN control software is expected to operate on and adapt to unseen network deployments and topologies. Precise routing-network modeling is a prerequisite for SDN to efficiently estimate and forecast network state. As a result, in certain scenarios, such as during network expansion, it is expected that: (i) several network features (possibly used in model training) become greatly imbalanced; (ii) unseen attribute values are expected to be out-of-distribution; (iii) varying, often larger, network sizes are encountered; and (iv) relational dependencies expand

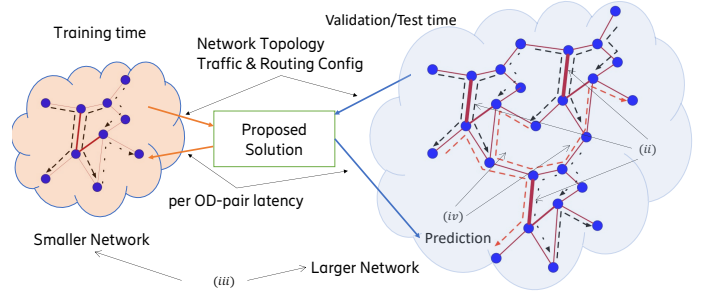


Figure 1: Schematic Representation of SDN Network State Estimation Problem. Directed line denotes network traffic flows with different throughput in the given topology, where one can see that there exist different distributions of line type between training and validation/test time, as mentioned in (i). Undirected line denote the link communicating pair of network elements, with the line's strength denoting the link's capacity level.

among longer node chains. An illustrative example of above is shown in Fig. 1. A successful approach to deal with the above-mentioned challenges, is to employ ideas from the domain of open-world machine learning (OWL) [1]. Compared with traditional machine learning, OWL is expected to extrapolate towards unseen input distribution, classes, and scaling in the training set. Note, that the problem of estimating the network state is EXPTIME-complete, as shown by [2], and has been extensively studied in the literature [3], where most commonly conventional deep-learning methods are employed in the context of reinforcement learning (RL) based network orchestration as well as deep-learning-based service optimization.

In this paper we consider the problem of estimating the latency between a source and a destination network node (denoted as *OD* pair) in a routing network. The problem setting is to learn a model on smaller networks and extrapolate the predictions on larger networks assuming open-world input, as illustrated in Fig. 1.

Conventional deep learning (DL) methods have focused in predictive user behavior modeling on service demand and data usage. In this setting, a line of work focuses on leveraging deep neural networks for adaptive optimisation of routing schemes, as by [4], which typically require similar data distribution in the test environment. Only few papers have considered to extrapolate their proposed model on a larger network [5], something which, this far, has been considered infeasible in real-world scenarios. Moreover, successful extrapolation would require introducing task-specific non-linearity (i.e., queuing theory and network calculus in the routing network [6]) in the model [7], something which, this far, has not been considered in state-of-the-art DL solutions.

Graph Neural Networks (GNNs) have been used successfully for incorporating domain knowledge into different problem settings, which in turn, help to devise robust models better suited for OWL in learning complex-system representations. Moreover, our work is primarily inspired by [8] who reveal that by solely observing key nodes in the SDN topology, the deep network can produce effective embeddings capable of predicting network performance.

Our study is among the first approaches that attempt to model the routing network state snapshot by learning the given topology structure for the task of estimating the network latency using open-world input. Our proposed solution transforms the task of estimating the latency between source and destination nodes, to a *link-attribute prediction task*, i.e., predicting each link that can potentially contribute to traffic delays occurring in the routing trajectory. Estimating every link's latency contribution requires three aspects: (i) the traffic amount passing through the link; (ii) the capacity of the link; and (iii) link's structural features. The link traffic and capacity jointly denote the data throughput, and the congestion state of the given link. The link's structural features can be learned from the network topology and they serve two purposes: (i) to identify key links that can become bottleneck of the network performance; and (ii) to compute pair-wise similarities, which are used to discover similar links. We train our model on distributions of a particular network size, yet, the model achieves at least 75% reduction in mean absolute percentage error (MAPE) measured in different sized networks during inference, i.e., without explicit training using the larger (up to 6 \times) network topologies. Last but not least, our solution requires no GPU resource while still ensures a faster training as well as 3–7 times faster inference speeds, with only 6–25% embedding size required, when compared, for example, with the best-performing benchmark models. The remaining sections are organised as follows: In Sec. II we discuss the related approaches in the domain of size generalisation of GNNs and graph-based ML in SDN routing, while in Sec. III

we present our problem formulation in more detail. In Sec. IV we present our methodology followed by Sec. V, which contains the experimental evaluation of our algorithm. Finally Sec. VI concludes with a discussion of our main contributions and considerations for future work.

II. RELATED WORK

Graph Convolutional Networks (GCN) is a known feature-extraction technique for non-Euclidean spaces [9]. GCNs are capable of learning the graph structure by aggregating each node's embedding with its neighbors. In the conventional architecture, GCN is proven to be capable of incorporating node and edge information in a undirected graph scope. We are currently witnessing an increasing number of publications on GCN models, which, on the one hand reveal the success of this model, and on the other hand expose its limitations.

GCN models are constrained due to aggregating only local neighbors. For large enough graphs, GCN may fail to capture distant (long-range) dependencies entirely. Efforts have been made towards both building deeper GNN models [10] and building expressive GNN layers on long-range dependencies. For example, [11] proposed the first effective method to encode long-range dependencies, between any two nodes, of the graph in the node embedding. Furthermore, [12] propose to apply graph attention network (GAT), to compute shortest-path-to-node attention, based on a transformer [13] architecture. This work is one of the most related approaches to the one presented here. Contrary to the aforementioned contributions, which are based on transductive settings, our work focuses instead on an inductive setting. In our model, we are incorporating path-to-node attention, but passively between the path's structurally similar node and the targeting node, to tackle longer chains of dependencies in the test data. In addition, we do not build very deep GNN solutions to enlarge the receptive field up to the length of dependency, for balancing between model accuracy, resource usage and inference speed trade-off.

In a survey by [7], the extrapolation ability of GNNs regards mainly two aspects: the ability to extrapolate towards *unseen structural features* as well as to extrapolate towards *out-of-distribution attributes*. [14] conclude that the size generalization can only be applied on graphs with certain structural features. Moreover, they suggest that the expressiveness of the GNN model must be still sufficient for applying it to larger graphs. [15] further elaborate from the perspective frequency response, pointing out that spectral graph convolution is inherently transferable for graphs with similar degree distribution.

In our model, we leverage the conclusions from the aforementioned works and cast the graph-size-generalization problem to a transferred formulation of graph, which is considered to be learnable by spectral graph-convolution methods. On the other hand, extrapolation towards out-of-distribution data is a fundamental challenge for most neural networks. For out-of-distribution attributes, the extrapolation ability relies more on the embedding and readout function of the GNN architecture. [7] have shown the multi-layered perception models (MLP)

can only extrapolate if the test set is expanded from the training set in all geometrical directions in the latent space. Both [16] and [17] have studied on approximating simple arithmetic computation (+, −, ×, and /) when extrapolate towards drifted numerical values. In our proposed model, we utilize building blocks from [16] to encourage the model to learn the arithmetic impact, when tackling drifted numerical input features.

More recent related works focus on learning the tuned graph representation of routing networks. [18] propose RouteNet, which is regarded as a first work that introduces message passing in deep-learning-based network modeling, in a bipartite graph formulation. [19] and [20] extend RouteNet to be adaptive to heterogeneous scheduling policies and routing scheme, while improving its accuracy and inference speed on similar topology structures. Finally, [5] present their work on topology size generalization for latency estimation of Origin-Destination (OD) pairs, the same problem we focus here. Through improving RouteNet by including queue occupancy state per link, beside the path and link nodes, they formulate a tripartite message passing scheme, which introduces size generalization ability to the model. Yet, none of aforementioned works utilize the topology's structural feature in network-state embedding, as we address in this work.

III. PROBLEM FORMULATION

The task we consider in this paper is to perform *mean latency estimation* on every OD pair p in the network. We summarize the defined notation in Table. I. The input to our problem is the following.

- 1) A sequence of attributed size-varying networks $\mathbf{G} = \{G_i(V_i, E_i; X_i, Y_i)\}$, where V_i denotes the $|V_i| = n_i$ network nodes and E_i denotes the $|E_i| = m_i$ links between pairs of nodes in V_i . For each node $v \in V_i$, a set of node attributes $x(v)$ are provided to describe the heterogeneous routing configuration on the network node level. Similarly, a set of link attributes $y(e)$ is introduced for each link $e \in E_i$ to denote link configuration attributes with respect to routing. The set of all node attributes in G_i is denoted by X_i , and the set of all link attributes is denoted by Y_i . Each network G_i includes k_i OD pairs $P_i = \{p_{i,j}\}$, with $j = 1, \dots, k_i$.
- 2) A sequence of traffic matrices $\mathbf{T} = \{T_i\}$, where each T_i corresponds to a network G_i . Each traffic matrix T_i consists by k_i vectors $\{t_{i,j}\}$, with $j = 1, \dots, k_i$, where each $t_{i,j} \in \mathbb{R}^2$ denotes the mean throughput and peak throughput features for the OD pair $p_{i,j}$. Note that there exist OD pairs with same source and destination nodes but different throughput features.
- 3) A sequence of routing matrices $\mathbf{R} = \{R_i\}$, where each R_i with dimension $\mathbb{R}^{n_i \times n_i}$, with 0's on the diagonal elements, corresponds to network G_i . Each element in R_i , denoted by $R_{i(c,d)} = r_i(v_c, v_d)$; $v_c, v_d \in V_i$ represents the next-hop per OD pair, given the current node v_c and the destination node v_d . Note that R_i contains information

for all possible routing pairs, regardless of the given OD pair elements in T_i .

- 4) *The ground truth*: A sequence of performance matrices $\mathbf{Q} = \{Q_i\}$, where each Q_i corresponds to network G_i . Each performance matrix Q_i consists of k_i vectors $\{q_{i,j}\}$, with $j = 1, \dots, k_i$, denoting the performance of each OD pair $p_{i,j}$ in global (mean latency) and local (per-link occupancy) metrics. The task is to estimate the mean latency value of each OD pair, denote as an element in $q_{i,j} = Q_i(p_{i,j})$, with $j = 1, \dots, k_i$.

Previous works [5], [18]–[20] formulate the problem defined above as a multi-step prediction problem. More specifically, given network snapshots (G_i, T_i, R_i) , the goal is to predict the future Δ steps of the network state $[Q_i^1, \dots, Q_i^\Delta]$ and pool the result of each step for a prediction Q_i' . In contrast, in our model, we consider this problem as a node attribute prediction problem. We introduce a modified problem formulation:

- 1) Given the input to our problem specified by a sequence of tuples $(G_i(V_i, E_i; X_i, Y_i), T_i, R_i)$, we first transform $G_i(V_i, E_i; X_i, Y_i)$ to a directed graph, where each undirected edge $(u, v) \in E_i$ is replaced by two directed edges (u, v) and (v, u) , and then transform it to a line graph $G_i^\mathcal{L}(V_i^\mathcal{L}, E_i^\mathcal{L})$. We define a projection \mathcal{L}^{-1} , to project the representation (i.e: $v_i \in V_i^\mathcal{L}, e_i^\mathcal{L} \in E_i^\mathcal{L}$) in line graph $G_i^\mathcal{L}$, back to its representation in G_i .
- 2) The edge set $E_i^\mathcal{L}$ of the line graph $G_i^\mathcal{L}$ is directed. The node set $V_i^\mathcal{L}$ includes only valid routings in R_i , i.e., for all $v \in V_i^\mathcal{L}$ its respective edge representation $\mathcal{L}^{-1}(v) = e(v_x, v_y)$ is in E_i . Thus, there exists $v_d \in V_i$ such that $r_i(v_x, v_d) = v_y$ holds.
- 3) We can compute the summed traffic T_i^s per-link over all OD trajectory pairs passing through, for all n_i nodes. Each element is denoted by $t^s(i, v) = T_i^s(v)$, for all $v \in V_i^\mathcal{L}$. Given node v 's edge representation $\mathcal{L}^{-1}(v) = e_v \in E_i$, We define the node attributes $X_i^\mathcal{L}(v)$ in $G_i^\mathcal{L}$ as follows:

$$X_i^\mathcal{L}(v) = \frac{t^s(i, v)}{c(Y_i(e_v))} \text{ and } t^s(i, v) = \sum_{p_k; v \in p_k} T_i(p_k), \quad (1)$$

for $v \in V_i^\mathcal{L}$ and $e_v \in E_i$ and $\mathcal{L}^{-1}(v) = e_v$

where $c(Y_i(e_v)) \in Y_i(E_i)$, denotes the capacity per-link in G_i .

- 4) For each edge $e \in E_i^\mathcal{L}$ connecting node v_s to node v_d in $G_i^\mathcal{L}$. We introduce corresponding edge weight $w_i(v_s, v_d)$ as follows:

$$w_i(v_s, v_d) = \frac{\sum_{p_k; v_s, v_d \in p_k} T_i(p_k)}{t^s(i, v_s)} \cdot \frac{Y_i(e_{v_s})}{Y_i(e_{v_d})}, \quad (2)$$

for $v_s, v_d \in V_i^\mathcal{L}$ and $e_{v_s}, e_{v_d} \in E_i$,
and $\mathcal{L}^{-1}(v_s) = e_{v_s}, \mathcal{L}^{-1}(v_d) = e_{v_d}$,

The motivation behind this re-formulation is multi-faceted: (i) Using directed links in $G_i^\mathcal{L}$ introduces OD trajectory pairs information into the graph structure. The new formulation separates the link's adjacency into in-degree and out-degree, which will be used to represent the link's in-coming traffic

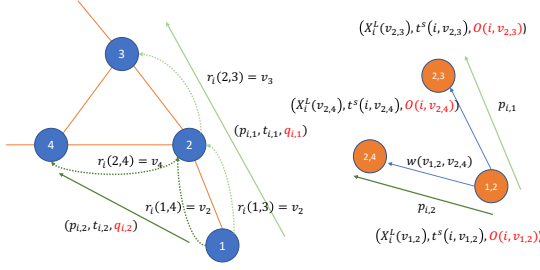


Figure 2: Comparative Example of the Problem Formulation, with Ground Truth being Red. Left: Previous works' formulation. Right: The proposed formulation

(in-traffic) and out-going traffic (out-traffic). (ii) For the task of estimating path latency $Q_i(p_{i,j})$, the problem can be decomposed into estimating the latency contribution of each hop $v \in V_i^L$ into $\sum_{v; v \in p_k} Q_i(v)$. According to queuing theory, such latency contribution further reduces to estimating v 's queue occupancy state $O(i, v) \in Q_i$. Thus, in the graph G_i^L , the problem is reduced to a node attribute prediction task.¹ (iii) Inspired by previous work, identifying key link that is shared by many OD pairs could have a dominating impact on the whole network performance. In the graph G_i^L , such feature is disclosed within the ego-networks of nodes $v \in V_i^L$, which can be jointly formed by the trajectories of all paths passing through v . Thus, one can discover the importance of node v in network topology by its *role* in G_i^L . (iv) The features computed in Eq. 1 and Eq. 2 are supported by queuing theory;² here we consider domain knowledge to be the key in producing a size-invariant graph signal on G_i^L . An illustrative example of re-formulation, comparing with the previous works' is given in Fig. 2.

IV. METHODOLOGY

Fig. 3 provides an overview of our proposed model for the prediction task. In the first phase (Fig. 3a), we transform the input graph G_i (Fig.3a (i), (ii)) into the line graph G_i^L (Fig.3a (iii), (iv)) for the model to learn in a latent space that generalizes well on different input graph sizes. Next (Fig. 3b), we separate the graph into a directed sub-graph and a un-directed sub-graph that capture the neighboring sequential relation as well as the impact of key nodes. We design the corresponding components that capture each impact and handle out-of-distribution features, so as to get consistent results across different graph sizes. Below, we discuss each component separately.

A. Directed Spectral Graph Convolution

To capture the sequential relation between nodes consisting of the OD pair's trajectory, as well as the OD pair's direction,

¹A detailed transform procedure is shown in Appendix.A. Link to the appendix: https://github.com/bluelancer/RoutingGNN_Appendix/blob/main/IEEE_Graph_Covolution_for_Routing_Appendix_.pdf

²A detailed proof is given Appendix.B-C, Link to the appendix same as above.

Table I: Notation Table

Network Topology Feature Notation:	
$\mathbf{G} = \{G_i\}$	Training/testing set of Topology
$G_i(V_i, E_i; X_i, Y_i)$	Topology snapshot graph
$V_i = \{v_i\}$	Network nodes in G_i
$n_i = V_i $	Size of G_i
$E_i = \{e_i\}$	Communication links between network nodes
$X_i = \{x(v_i)\}$	Network node configuration of G_i
Y_i	Communication link configuration of G_i
$c(Y(e_i))$	Configured link capacity of $e_i \in E_i$
Network Service Feature Notation:	
$P_i = \{p_{i,j}\}$	OD pairs in G_i
$p_{i,j}$	j -th OD-pair in G_i
k_i	Number of OD pairs in G_i
$\mathbf{T} = \{T_i\}$	Training/testing set of user traffic
$T_i = \{t_{i,j}\}$	User traffic matrix for all P_i
$t_{i,j}$	User traffic features for $p_{i,j} \in P_i$
Network Routing Feature Notation:	
$\mathbf{R} = \{R_i\}$	Training/testing set of routing matrices
R_i	Routing matrices corresponds to network G_i
$R_i(c,d)$	Next-hop of current node and destination being $v_c, v_d \in V_i$
Ground Truth Notation:	
$\mathbf{Q} = \{Q_i\}$	Training/testing set of performance matrix
Q_i	Performance matrix correspond to (G_i, T_i, R_i)
$q_{i,j}$	Performance measurement (i.e: latency) of $p_{i,j}$
Novel Feature Notation in reformulating the problem :	
$G_i^L(V_i^L, E_i^L)$	Line graph transformation of $G_i(V_i, E_i)$
\mathcal{L}^{-1}	Projection from representation in G_i^L to its corresponding representation in G_i
T_i^s	Matrix of summed traffic of $p_{i,j}$, who share the same node $v \in V_i^L$
$t^s(i, v)$	Summed traffic for all $p_{i,j}$ passing through node $v \in V_i^L$
$X_i^L(v)$	Line graph node feature, introduced in Equ. 1
w_i	Line graph edge weight, introduced in Equ. 2
$O(i, v)$	node v 's queue occupancy state (ground truth), for $v \in V_i^L$

we use the architecture designed by [21]. It is a spectral convolution block with its reception field encoded with direction information, named Directed Graph Convolution Network (DGCN). Given the processed directed G_i^L , we separate the adjacency matrix into three different formations: A_ℓ^F , $A_\ell^{S_{in}}$ and $A_\ell^{S_{out}}$, which denote first-order proximity, second-order in-proximity and second-order out-proximity, as shown in Fig. 3b. Given the weighted adjacency matrix A_ℓ of the line graph G_i^L , one can compute the aforementioned quantities:

$$A_\ell^F = (A_\ell + A_\ell^T)/2, \quad (3)$$

$$A_\ell^{S_{in}} = \sum_k \frac{A_{\ell(k,i)} A_{\ell(k,j)}}{\sum_v A_{\ell(k,v)}} \text{ and } A_\ell^{S_{out}} = \sum_k \frac{A_{\ell(i,k)} A_{\ell(j,k)}}{\sum_v A_{\ell(v,k)}}, \quad \text{for } (i, j) \in V_i^L \quad (4)$$

We apply graph convolution to the aforementioned adjacency matrix, with adding self-loops, denoted by $\tilde{A}_\ell^F = A_\ell^F + I$, $\tilde{A}_\ell^{S_{in}} = A_\ell^{S_{in}} + I$, $\tilde{A}_\ell^{S_{out}} = A_\ell^{S_{out}} + I$. We concatenate (denote

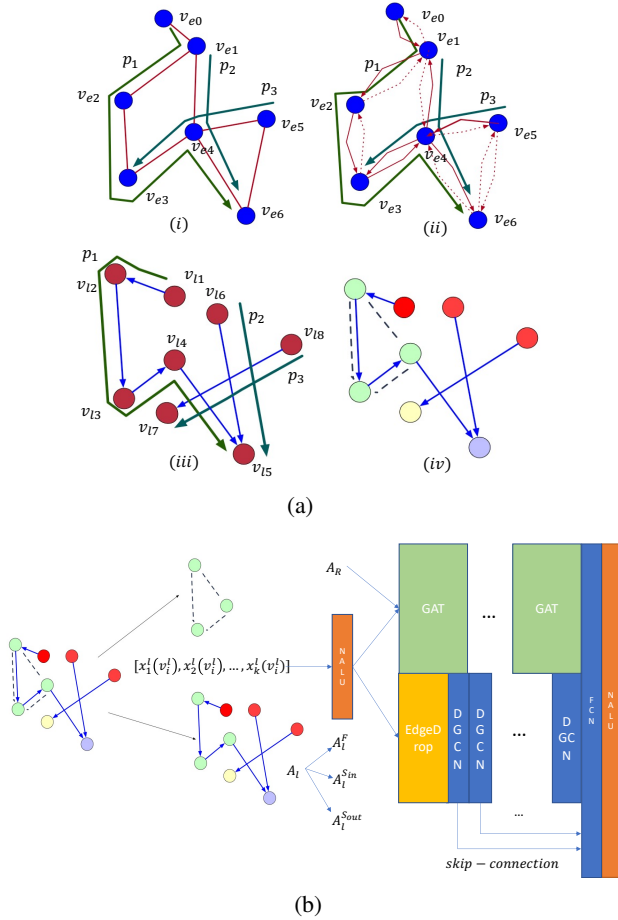


Figure 3: (a) Pre-processing: Incorporate OD Information in Node's Local Structural Feature; (b) Learning and Inference: Extract Neighboring Impact and Long Range Dependencies

as $||$) each result with a train-able weight (α, β) . Θ is a train-able matrix. We define

$$f_x(X, A_x) = \tilde{D}_x^{(-\frac{1}{2})} \tilde{A}_x \tilde{D}_x^{(-\frac{1}{2})} X \Theta, \quad (5)$$

where $\tilde{D}_{xii} = \sum_j (\tilde{A}_{xij})$ is a diagonal matrix. The result of each DGCN block can be formulated as:

$$H^{(l+1)} = ||(f_F(H^{(l)}, \tilde{A}_\ell^F), \alpha f_{S_{in}}(H^{(l)}, \tilde{A}^{S_{in}}), \quad (6)$$

$$\beta f_{S_{out}}(H^{(l)}, \tilde{A}^{S_{out}})). \quad (7)$$

We expect $f_F, f_{S_{in}}$ and $f_{S_{out}}$ to capture the global traffic dependency, in-traffic dependency and out-traffic dependency respectively. One could foresee that the deeper the model is build, such dependency can be more precisely captured. We used EdgeDrop [22] as well as skip-connection to avoid over-smoothing, both of which have been found to help building deeper networks during the experimentation.

B. Embedding and Readout Function with Extrapolation Ability

For handling the out of distribution attribute value per-node in larger graph, we have used NALU [16] to replace

MLP in common GNN setup. Another motivation is to build equivalence in the embedding space between a node's out-traffic attribute and its in-traffic attributes' from its neighbors. This is because the in-traffic and out-traffic of a node should remain equal, unless packet drop happens. [16] have verified its superiority in numerical extrapolation in neural network readout.

C. Routing Role Recognition based Attention

Inspired by the work of [8], identification of key nodes in the SDN is essential to estimate the whole networks performance. For this purpose, we involve role recognition [23] during the pre-processing phase (denote as different colors in Fig. 3a (iv)). By defining $v_s, v_d \in V_i^L$ as nodes in G_i^L , we consider their role to be $\mathcal{R}(v_s), \mathcal{R}(v_d)$. We define the role adjacency matrix $A_{\mathcal{R}}$ of G_i^L :

$$A_{\mathcal{R}(s,d)} = \begin{cases} 1, & \text{if } \mathcal{R}(v_s) = \mathcal{R}(v_d), \text{ and exists } p \in P_i, \\ & \text{such that } v_s, v_d \in p \\ 0, & \text{otherwise} \end{cases}$$

We consider the pair-wise similarity between the role of a node and its neighbors to be crucial, since an OD pair's source or destination can bring its impact towards close neighbours of nodes of interest and formulate their embedding as an "augmented source." GAT is reported to be a well-suited solution to capture such pair-wise similarity impact. Thus, we apply GAT layers on node embedding between each role adjacency pair. The motivation is to capture the long-range dependency of in-trajectory of OD pair, without collecting all hops' feature in the trajectory. For the aforementioned A_ℓ , one can consider that denser connected nodes in G_i^L are usually congested. Thus, its state (e.g., occupancy) is dependant on its neighbors' states. For less connected nodes, one can infer that they have few OD's pairs passing through. Thus, their states are more dependent on the few passing through OD pair's demanding traffic. This information might be submerged for those well-connected nodes in the OD pair trajectory, but more obvious on the weakly-connected nodes sharing the same trajectory. Such pair-wise state similarity forms $A_{\mathcal{R}}$, which enlarges the receptive field of the model beyond topological neighbors. Our results show that this approach improves the model performance as well as it introduces generalization ability when training and inference takes place using samples with longer chains of trajectories.

V. EXPERIMENTS

A. Dataset and Baselines

Dataset. We verify the performance of the model on a simulated dataset provided by [24]. The training, validation and test sets are produced by a packet level simulator (OMNeT++ v5.5.1 [25]). Though lacking of other benchmark dataset, this dataset includes a summary of patterns of real-world network topologies from The Internet Topology Zoo [26]. The training set includes a variety of networks sized from 25 to 50 network nodes, while the test and validation set are sized from 50 to

300 network nodes. The larger network brings the following attribute changes: (1) Network topology samples introducing a few longer OD pair’s trajectories; (2) Network topology samples introducing larger capacity attribute per link, as well as different “next-hop” choice encoded in R_i , for OD pairs to generate their trajectory in a different routing preference; (3) Network topology samples introducing both sampling methods (1) and (2), as described above. A description of the used dataset is shown in Table II.³

Baselines. We have considered several state-of-the-art models to compare our proposed model.

- 1) **RouteNet** [18] is a message-passing-based GNN solution to estimate per OD pair’s latency in SDN on bipartite graph problem formulation.
- 2) **RouteNet-Salzburg** [19] is an optimization model based on RouteNet that scales to different routing policies on same sized network topologies on estimating per OD pair’s latency. This is a winning solution of [24].
- 3) **RouteNet-GA**⁴ is a fine-tuned solution based on RouteNet with better generalised performance on various OD pair trajectory length, this is a winning solution of [24].
- 4) **DGCN** [21] is a graph spectral convolution solution generalized for directed graph scenario. In our experiment, we apply NALU as its embedding and readout function, as well as same hyper-parameters and pre-processing scheme as the proposed model for a fair comparison.
- 5) **Scalable-RouteNet** [5], [27]⁵ is a RouteNet based improved solution to estimate per OD pair’s latency in size-variant SDN on tripartite graph problem formulation.

B. Experiment Settings

We have implemented our proposing solution with Keras as well as tf-geometric [28]. Furthermore, our solution requires no-GPU, whereas for the other benchmark solutions that do require GPU, we have been training using one Nvidia Geforce 1080 Ti and GNU-Parallel for resource management. For training the comparative benchmark models, we used the recommended parameter sets re-

³Detailed information on feature distribution as well as comparison of the training and testing dataset is given in Appendix.D. Link to the appendix same as above.

⁴Code available at <https://github.com/ITU-AI-ML-in-5G-Challenge/PS-014.2-GNN-Challenge-Gradient-Ascent>

⁵We report their published results, however, we have not been given access to their implementation to reproduce their results.

Table II: Network Topology Graph Set Statistics

Dataset	Training Set	Validation & Test set
Number of Samples	120000	3120
Graph size	[25, 50]	[50, 300]
Average Degree (mean, std)	(9.778, 0.9491)	(9.523, 1.268)
Average Shortest Path (mean, std)	(2.379, 0.1372)	(3.384, 0.456)
Diameter (mean, std)	(4.458, 0.6441)	(6.667, 1.280)
Cluster Coefficient (mean, std)	(0.131, 0.0243)	(0.0427, 0.0342)

ported by the authors, but with the unified learning rate $r = 10^{-3}$, epoch number = 250, epoch size = 4000, loss function as *MAPE* as well as input feature space for a fair comparison. We train five independent random seeds for each model and report the averaged solution. The proposed method, based on aforementioned preprocessing and formulation, are taking the $G_i^L(V_i^L, E_i^L)$ as well as node feature $X_i^L(v)$ and edge weight w_i in its input layer, with a 32-dim output layer for each node embedding in V_i^L . A detailed list of training parameters is published on <https://github.com/bluelancer/GNNET.git>.

C. Experiment Result

We evaluate the performance of our model in the following experiments: (1) Table III reports the overall performance of all the test set and we note their difference in *MAPE* through *Delta*; (2) Fig. 4 reports model performance on each interval of size samples from the test set; (3) Table IV reports mean inference speed on a selection of certain sized samples from test set. Here we are only concerned with the time interval of inference, without any pre/post-processing time, as they can be parallelized in an actual application. We do not include results from **Scalable-RouteNet** due to the use of different hardware. (4) Fig. 5 reports an ablation study towards the decision of adopting NALU instead of MLP in proposed model when extrapolating towards different size of graph. Fig. 6 reports a sensitivity study result on most affecting parameter and configuration in training the proposed model.

Table III: Average Performance (%)

Model	MAPE	Delta
Our model	2.317	—
RouteNet	274.5	272.2
RouteNet-Salzburg	596.8	594.5
RouteNet-GA	557.8	555.5
DGCN	2.935	0.618
Scalable-RouteNet ⁶	10	7.683

Table IV: Inference Speed(*ms*) per Topology vs. Topology Size

Topology Size	50	100	200	300
Our model	161.5	373.3	1115	2718
RouteNet	485.3	1972	6084	21264
RouteNet-Salzburg	0.41 hr	0.4 hr	0.39 hr	0.44 hr
RouteNet-GA	1363	4786	19913	71767
DGCN	105.5	220.3	861.9	1770

Based on the above results, one can see that our model outperforms all baselines in terms of accuracy, inference speed and generalization ability to open-world input. For Table III, we observed that our model outperforms in terms of accuracy. We consider this verifies the validity of both our modified problem formulation, and the superiority of the proposed model. One can observe that other well-studied GNN benchmark (i.e:DGCN) can also outperform greatly against the carefully designed task-specific model, which can be concluded into the effect of problem formulation.

For Table IV, one can conclude that our model is faster

⁶The reported result is referenced from [5]

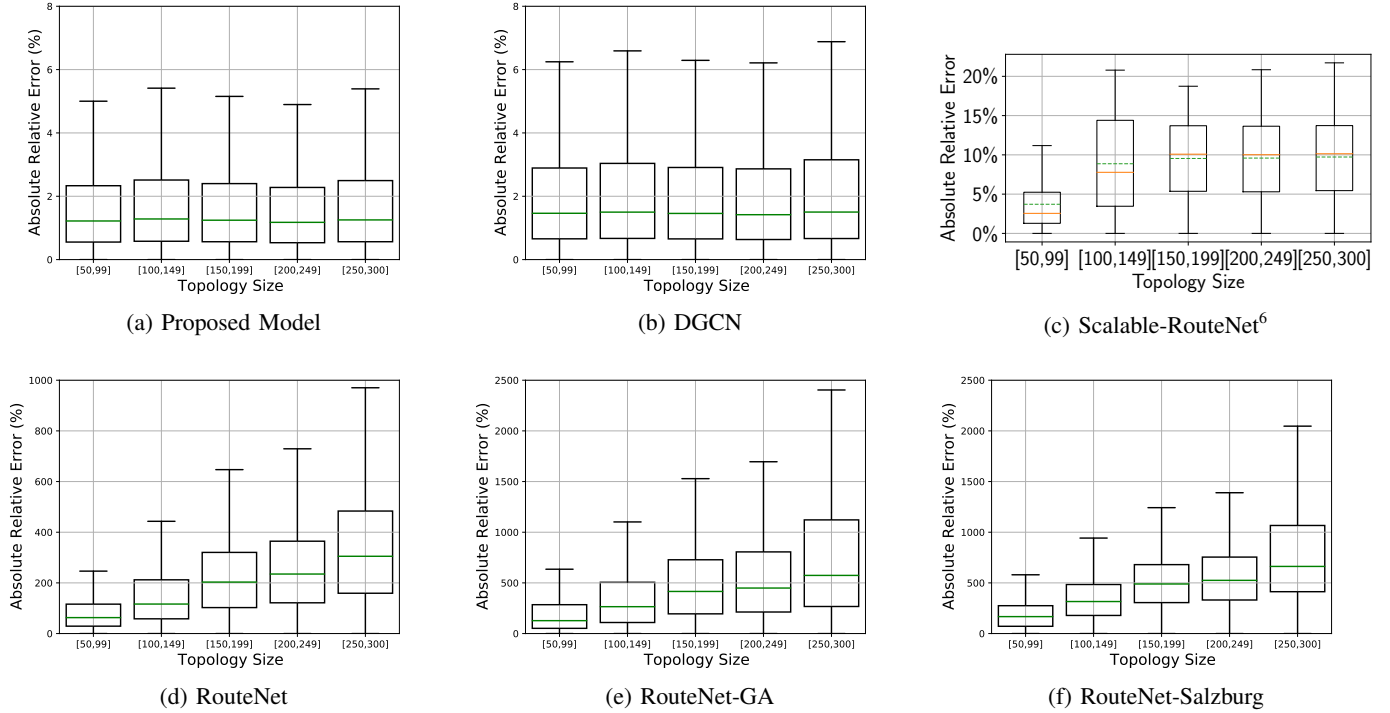


Figure 4: Model Generalization Ability Towards Open-World Input

than most task-specific benchmark models (i.e: Routenet, RouteNet-Salzburg, and RouteNet-GA). The proposed model’s inference time remains tolerable for SDN routing estimation component, even for the largest network we have (300 nodes). DGCN performs marginally faster as it is one of the building blocks of the proposed model, while the proposed model is more accurate and has a more stable generalization performance against open-world input.

For Figure 4, we conclude our proposed solution performs best in generalization ability to open-world input. Note that comparing solely DGCN and our model, we could see $A_{\mathcal{R}}$ not only introduces more accurate estimation, but also increases the performance stability. We believe that this improvement is due to introducing role inference in our model, which provides an easy way to diffuse long-range dependencies.

For Figure 5, we verified the benefit that we bringing in NALU instead of MLP in common GNN architecture, we observed using NALU obtains marginal gain on median accuracy but more on variance of the accuracy. This verified the conclusion from [16] that NALU assists in modeling numerical extrapolation in larger routing network samples.

VI. CONCLUSION

The paper offers the following contributions from the state of art: (i) We proposed a novel formulation of the latency prediction task in SDN. (ii) We proposed a solution that incorporates domain knowledge in SDN optimization aimed at an open-world learning setup. (iii) Our proposed model achieves better accuracy, inference speed and generalization

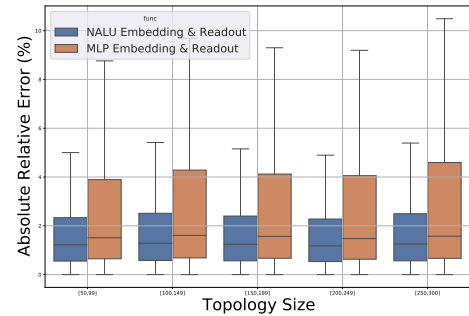


Figure 5: Ablation Study of the Proposed Model w.r.t NALU and MLP Embedding & Readout Function

ability beyond state of the art, all while using less computational resources. A limitation in our approach is that we do not incorporate the scenarios of peak hours, where unseen amount and duration of burst traffic triggers the network orchestration scheme, which eventually results in a dynamic-routing topology. Generalization to dynamic and size-variant graphs with GNN models remain to be studied in future work.

ACKNOWLEDGEMENT

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. We thank

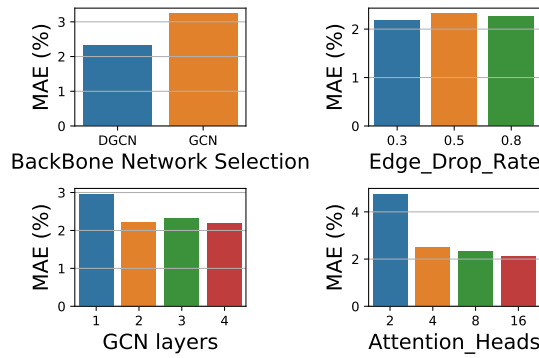


Figure 6: Sensitivity Study on the criteria of Mean Absolute Relative Error (MAE) w.r.t a Selection of Most Performance-affecting Hyper-parameter

Dr. Pedro Batista and Dr. Alessandro Previti from Ericsson Research for the insightful discussions.

REFERENCES

- [1] J. Parmar, S. S. Chouhan, and S. S. Rathore, "Open-world machine learning: Applications, challenges, and opportunities," *arXiv preprint arXiv:2105.13448*, 2021.
- [2] D. Gamarnik and S. K. Gupta, "Algorithmic challenges in the theory of queueing networks," 2011.
- [3] R. Amin, E. Rojas, A. Aqdu, S. Ramzan, D. Casillas-Perez, and J. M. Arco, "A survey on machine learning techniques for routing optimization in sdn," *IEEE Access*, 2021.
- [4] V. Srivastava and R. S. Pandey, "Machine intelligence approach: To solve load balancing problem with high quality of service performance for multi-controller based software defined network," *Sustainable Computing: Informatics and Systems*, vol. 30, p. 100511, 2021.
- [5] M. Ferriol-Galmés, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Scaling graph-based deep learning models to larger networks," *arXiv preprint arXiv:2110.01261*, 2021.
- [6] F. Ciucu and J. Schmitt, "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.
- [7] K. Xu, M. Zhang, J. Li, S. S. Du, K.-i. Kawarabayashi, and S. Jegelka, "How neural networks extrapolate: From feedforward to graph neural networks," *arXiv preprint arXiv:2009.11848*, 2020.
- [8] P. Sun, J. Li, Z. Guo, Y. Xu, J. Lan, and Y. Hu, "Sinet: Enabling scalable network routing with deep reinforcement learning on partial nodes," in *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, 2019, pp. 88–89.
- [9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
- [10] G. Li, M. Müller, B. Ghanem, and V. Koltun, "Training graph neural networks with 1000 layers," *arXiv preprint arXiv:2106.07476*, 2021.
- [11] B. Chen, R. Barzilay, and T. Jaakkola, "Path-augmented graph transformer network," *arXiv preprint arXiv:1905.12712*, 2019.
- [12] Y. Yang, X. Wang, M. Song, J. Yuan, and D. Tao, "Spagan: Shortest path graph attention network," *arXiv preprint arXiv:2101.03464*, 2021.
- [13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
- [14] G. Yehudai, E. Fetaya, E. Meirom, G. Chechik, and H. Maron, "From local structures to size generalization in graph neural networks," in *ICML*. PMLR, 2021, pp. 11 975–11 986.
- [15] M. Balcilar, G. Renton, P. Héroux, B. Gaüzère, S. Adam, and P. Honeine, "Analyzing the expressive power of graph neural networks in a spectral perspective," in *ICLR*, 2020.
- [16] A. Trask, F. Hill, S. E. Reed, J. W. Rae, C. Dyer, and P. Blunsom, "Neural arithmetic logic units," in *NeurIPS*, 2018.
- [17] A. Madsen and A. R. Johansen, "Neural arithmetic units," in *ICLR*, 2019.
- [18] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "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.
- [19] M. Happ, M. Herlich, C. Maier, J. L. Du, and P. Dorfinger, "Graph-neural-network-based delay estimation for communication networks with heterogeneous scheduling policies," *ITU Journal on Future and Evolving Technologies*, vol. 2, no. 4, 2021.
- [20] Y. Kong, D. Petrov, V. Räisänen, and A. Ilin, "Path-link graph neural network for ip network performance prediction," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 170–177.
- [21] Z. Tong, Y. Liang, C. Sun, D. S. Rosenblum, and A. Lim, "Directed graph convolutional network," *arXiv preprint arXiv:2004.13970*, 2020.
- [22] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *arXiv preprint arXiv:1907.10903*, 2019.
- [23] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li, "Rolx: structural role extraction & mining in large graphs," in *Proceedings of the 18th ACM SIGKDD*, 2012, pp. 1231–1239.
- [24] J. Suárez-Varela *et al.*, "The graph neural networking challenge: a worldwide competition for education in ai/ml for networks," *ACM SIGCOMM Computer Communication Review*, vol. 51, no. 3, pp. 9–16, 2021.
- [25] A. Varga, "Discrete event simulation system," in *Proc. of the European Simulation Multiconference (ESM'2001)*, 2001, pp. 1–7.
- [26] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.
- [27] M. Ferriol-Galmés, K. Rusek, J. Suárez-Varela, S. Xiao, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet-erlang: A graph neural network for network performance evaluation," 2022.
- [28] J. Hu, S. Qian, Q. Fang, Y. Wang, Q. Zhao, H. Zhang, and C. Xu, "Efficient graph deep learning in tensorflow with tf-geometric," 2021.