IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 November 2023, accepted 5 December 2023, date of publication 7 December 2023, date of current version 13 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3340696

RESEARCH ARTICLE

Recursive Router Metrics Prediction Using Machine Learning-Based Node Modeling for Network Digital Replica

KYOTA HATTORI, TOMOHIRO KORIKAWA, (Member, IEEE), CHIKAKO TAKASAKI, AND HIDENARI OOWADA
NTT Network Service Systems Laboratories, NTT Corporation, Musashino-Shi, Tokyo 180-8585, Japan
Corresponding author: Kyota Hattori (kyota.hattori.vj@hco.ntt.co.jp)

ABSTRACT Future network infrastructures need to safely and rapidly provide network services in complex conditions that include many devices and multiple access lines, such as 5th-generation (5G) and 6th-generation (6G) mobile systems supported by multiple carriers. Additionally, future telecommunications networks will utilize network disaggregation techniques to take advantage of the highest quality technology from various vendors to meet service requirements. Therefore, it is necessary to enhance the verification of combinations of various network equipment and components that constitute network infrastructure. Our motivation is to investigate the potential to enable the verification of network node performance digitally to support future network infrastructures. This study concentrates on improving the accuracy of the metric inference of black-boxed network nodes when only the network node configurations and traffic conditions are available as external conditions. Our main contribution is as follows: We provide a novel method of machine learning based on network node modeling to improve the accuracy of network node metric inference for throughput, packet loss rate, and packet delay by recursively appending inferred other node metrics to the training datasets in accordance with feature importance; we demonstrate the application of the proposed method to 14 baseline machine learning algorithms for evaluating the accuracy of inferred network node metrics; finally, we show improvement in utilization of network resources for accommodating traffic on a fixed network with a traffic policer, whose parameters are set using the proposed method. Additionally, we investigate the impact of appending inferred network node metrics to the training datasets, which is a key feature of the proposed method, on computational time and the possibility of overfitting.

INDEX TERMS Recursive router metrics inference, network node modeling, network digital replica.

I. INTRODUCTION

FUTURE technologies will enable various devices and things connected to the Internet and each other to achieve data exchange and decision-making with automation, prediction, and self-optimization [1]. In response to this, the traffic on future networks will be generated by human communication, connected intelligent machines and things embedded with artificial intelligence. This has motivated the development of the future network, which must handle traffic on unknown networks generated by various devices and things as well as human beings.

This trend will also affect telecommunication networks. Future network infrastructures need to safely and rapidly provide network services in complex conditions that include many devices and multiple access lines, such as 5th-generation (5G) and 6th-generation (6G) mobile systems supported by multiple carriers. For example, a flying ad hoc network [2], [3] based on Internet of Things (IoT) devices could be beneficial for controlling and inspecting rescue operations and disaster areas. Constructing a wireless ad hoc network requires the supervision of numerous resource-limited

The associate editor coordinating the review of this manuscript and approving it for publication was Orazio Gambino.

IoT devices that are in motion and interacting with each other. In most cases, the cost of verifying telecommunication carrier requirements, such as network quality and reliability, is higher when many different IoT devices are deployed as part of a telecommunication carrier network. Consequently, it is challenging to rapidly meet telecommunication carrier requirements while providing these network services.

In addition, network disaggregation technology [4] is proposed as a potential solution for future telecommunication networks. Network disaggregation means separating a previously combined network node into network components. With their proprietary software and hardware, current network components need more flexibility and cost-efficiency to satisfy telecommunication carrier requirements. Network disaggregation techniques will enable telecommunication carriers to avoid the restrictions of vendor lock-in and utilize the most suitable technology from various providers to meet service requirements. However, network disaggregation technologies will result in a significantly higher number of network components and thus require management and assessment of vast combinations of numerous network components. Consequently, the coming network infrastructure must be verified to ensure network quality and reliability for a wide range of devices and to select the most appropriate mix of network components to support network disaggregation.

To achieve future network infrastructure, Digital Twin (DT) technologies may be a key enabler for robust, reliable, and high-performance digital infrastructure [5]. Due to the remarkable improvements in the performance of central and graphics processing units (CPUs and GPUs), the idea of DTs is being utilized in various industries [6], [7]. A DT is a virtual representation of a real-world entity that is constantly updated with data and information gathered from its physical counterpart, so that it can be used to predict its physical counterpart's future behavior. To create a DT, system data such as device configurations and traffic data from the real world needs to be collected in real-time to replicate the state of the physical entity [5]. DTs for jet engines [6] and a ground-steering system for an aircraft [7] have been researched in the aerospace industry to predict future risks in digital space.

Various techniques for applying DTs to telecommunication networks have been proposed [8], [9]. Telecommunication carriers are exploring the potential of DTs to manage and operate networks effectively [8]. Furthermore, many vendors of telecommunication network equipment are attempting to utilize DTs to create and sustain physical entities at remote locations [9]. However, the conventional approaches used in these studies need to consider the internal structure and behavior of network nodes, including hardware logic and implemented software processes. Emulating the internal structure and behavior of network nodes to ensure the performance of each network component for network disaggregation must be considered to meet carrier requirements.

To address this challenge, machine learning, especially deep learning, will be a key enabler. Machine learning has rapidly advanced with featuring cutting-edge algorithms and has firmly established itself as a state-of-the-art technology in diverse domains, including computer vision, natural language processing, chemical engineering, and more. To name just a few examples, researchers have proposed the following techniques in each domain: deep learning techniques with attention mechanisms for many visual tasks, including image classification, object detection, video understanding, and image generation [10]; deep reinforcement learning for text generation, machine translation, and the development of conversational systems [11]; and graph-based neural networks for enhancing the efficiency of the drug discovery process [12]. Despite the remarkable progress in applying machine learning to these domains, the impact of machine learning on the performance of network equipment is still being explored.

Given the above background, our motivation is to enable the verification of network node performance and the combination of a large number of network components digitally to support future network infrastructures. To accomplish this, we introduce Network Digital Replica (NWDR), which is a virtualized version of a physical network that enables verification of network equipment performance, as well as the combination of large numbers of network components in digital space for conditions not yet encountered on physical networks [13], [14]. Using machine learning-based network node modeling for actual node metrics, the proposed NWDR can generate optimal network configurations by creating network node models. To the best of our knowledge, few or no reports on router modeling apply regression prediction for router performance based on training datasets using actual router configurations and traffic data.

Our main contributions include:

- A novel machine learning-based network node modeling method to improve the accuracy of network node metric inference for throughput, packet loss rate, and packet delay by recursively appending the inferred other node metrics to the training datasets in accordance with feature importance.
- Demonstration of the application of the proposed method to 14 baseline machine learning algorithms for evaluating the accuracy of inferred network node metrics using the coefficient of determination (R^2).
- Improvement in network resource utilization for accommodating traffic on a fixed network on which a traffic policer is implemented and whose parameters are set based on the inferred network node metrics using the proposed method.

In this paper, we extend the work in [14] by further studying the application of the proposed method to other machine learning algorithms. We extensively apply the proposed method to 14 machine learning algorithms for evaluating R^2. The proposed method tends to increase the computational time and may cause overfitting due to adding recursively inferred node metrics to the training datasets. Therefore, we present an extensive evaluation of computational time, learning curves, residual plots, and the predicted R^2 to identify the practicality of these values. Also, we present extensive simulation results that show improvement in the number of accommodated traffic flows with a traffic policer, whose parameters are set based on the inferred network node metrics by using the proposed method.

The rest of this paper is organized as follows. Related works and the NWDR architecture to reproduce the internal operation of the actual nodes digitally are outlined in Sections II and III, respectively. The proposed network node modeling to achieve highly accurate network node metric inference, which is the primary role of the NWDR, is described in Section IV. Section V contains an account of the experimental and simulated outcomes. To evaluate the effectiveness of applying the proposed method to machine learning algorithms, we evaluate R^2 and computational time for 14 machine learning algorithms. Then, we show the utilization of network resources to accommodate TCP flows based on the inferred network node metrics by using the proposed method. Section VI discusses the applicability and limitations of the proposed method. Section VII describes challenges and future directions. Finally, this paper is concluded in Section VIII.

II. RELATED WORKS

Several techniques have been proposed to reproduce actual nodes' processes and behavior: mathematical models, network simulators, network emulators, and data-driven approaches. Mathematical models [15] are an invaluable asset for network design and analysis. Unfortunately, these models either do not accurately reflect the actual processing in a network node or are so complicated that they have no practical use. Consequently, a model that is both viable and uncomplicated needs to be developed. Network simulators, such as NS-3 [16] and QualNet [17], are computer programs that can replicate the behavior of a network by computing the effects between various network elements, which are normally discrete event-driven. These are usually used for computer simulations of network traffic. However, these network simulators do not take into account the inner architecture and functioning of network nodes, including the hardware logic and software programs.

In contrast, a network emulator [18] is a device or program that accurately emulates the behavior of a network, enabling the performance of a real program to be tested under various network conditions. However, network emulators typically require the same hardware logic and software programs as actual network nodes to replicate their processes and behavior. Comparing these methods, the accuracy and cost of a network emulator are generally higher than that of a network simulator [18]. On the other hand, a data-driven approach can be used to study the inner workings and behavior of a network node. For example, a graph neural network [19], which is a form of deep learning for graph data structures, has been proposed as a potential network modeling technique. A graph neural network can extract feature information from graphs and make valuable predictions. In general, graph neural network training results depend on the structure of the underlying graph. However, for application to network disaggregation, it is crucial to design the representation of each node in a way that enables performance evaluation at the network node or network component level to be easily isolated and flexible recombination of each entity. Considering application to network disaggregation, the proposed method consists of a twofold approach: node modeling to evaluate the performance of individual network nodes and network components using a machine learning algorithm, and network simulation to mimic the external environment. The proposed approach enables us to evaluate the performance of the network node in the context of network disaggregation.

Our proposed network node modeling aims to improve the accuracy of network node metric inference for throughput, packet loss rate, and packet delay using a machine learning algorithm. This is done by recursively appending inferred other node metrics that correlate with the target node metric to the training datasets according to feature importance. Using feature importance, we select the inferred other node metrics that have a high correlation with the target node metric, as the metrics to be added to the training dataset. On the other hand, ensemble learning [20] comes close to this concept. Ensemble learning selects multiple different machine learning algorithms and combines their predictions to improve the overall prediction accuracy by leveraging the strengths of different algorithms. However, the novelty in this study is that instead of selecting different machine learning algorithms, we select different performances of inferred node metrics based on feature importance between node metrics and consider the order in which they are added to the training dataset. This approach is useful for leveraging the performance data of network node metrics that telecommunication carriers have collected through the operation of carrier networks.

III. NETWORK DIGITAL REPLICA

A. CONCEPT

The concept of NWDR is illustrated in Fig. 1. An NWDR performs a digital assessment of the effect of new network configurations when devising and examining them without accessing the physical entities. The points below are essential to achieve the NWDR.

- 1) Gathering extensive quantities of data from physical networks: All information on network configuration and statistical data from various network equipment needs to be gathered to construct the NWDR based on the physical network's data.
- 2) Preprocessing for sparse and large amounts of data: To cut down expenses associated with network monitoring or make network monitoring for the various specifications of network equipment easier, the NWDR needs to manage both sparse and dissimilar types of data obtained from the physical network. To manage these data properly, preprocessing techniques such as

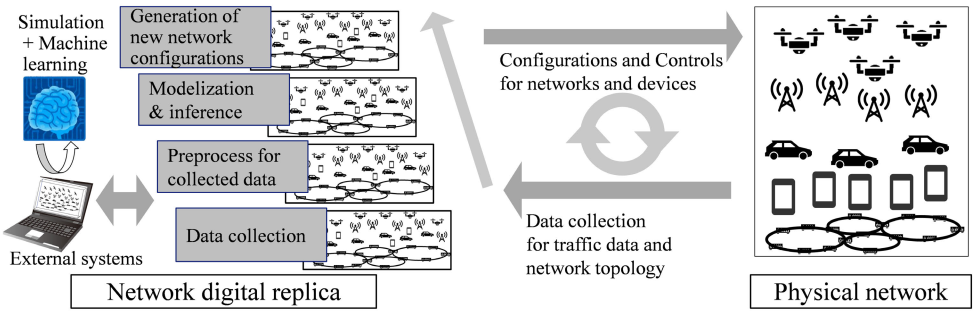

Simulation + Machine learning
Generation of new network configurations
Modelization & inference
Preprocess for collected data
Data collection
External systems
Network digital replica
Configurations and Controls for networks and devices
Data collection for traffic data and network topology
Physical network

FIGURE 1. Concept of proposed network digital replica [14].

data sampling and data interpolation are implemented to make it easier for the network node model.

- 3) Modeling the behavior of a network node and assessing its performance under new external conditions: The NWDR generates a network node model based on the data collected from the physical network using a machine learning algorithm. This model reproduces the same internal operation and the same function as the actual node in digital space to reduce the cost of preparing the same hardware logic and software programs of network equipment. Using this node model, the NWDR infers the performance of network nodes for different external conditions by reproducing the inner workings of actual network nodes.
- 4) Creating new network configurations and controlling the physical entities: Based on the NWDR's inferences and evaluations of the effects of unknown external conditions, optimal configurations for the current network nodes are created. Then, the physical network can be controlled with these optimal configurations.

By realizing these points, the NWDR can be created and linked to the information from a physical network in the digital domain, enabling it to digitally evaluate the performance of new devices and network configurations integrated into the physical network.

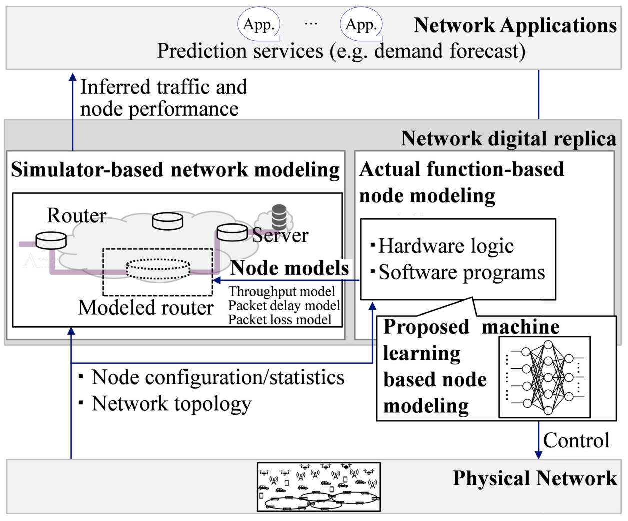

App.
App.
Network Applications
Prediction services (e.g. demand forecast)
Inferred traffic and node performance
Network digital replica
Simulator-based network modeling
Actual function-based node modeling
Router
Server
Node models
Throughput model
Packet delay model
Packet loss model
Modeled router
・Hardware logic
・Software programs
Proposed machine learning based node modeling
・Node configuration/statistics
・Network topology
Control
Physical Network

FIGURE 2. Architecture of the proposed network digital replica [14]. The NWDR connects the physical network to a network application to evaluate the network performance for unknown external network conditions in the digital space to evaluate whether network requirements will be achieved. To accomplish this, the NWDR consists of simulator-based network modeling and actual function-based node modeling. The simulator-based network modeling simulates the traffic from user devices as unknown external network conditions, and the actual function-based node modeling reproduces the inner workings and functioning of the target node using a machine learning algorithm for unknown external network conditions.

B. SCENARIOS USING NETWORK DIGITAL REPLICA

The NWDR will be created in consideration of both technological advances and cost requirements. The expected scenarios and examples are outlined below.

- (Scenario 1) NWDR for adjusting temporal enlargement and restructuring existing networks: The NWDR enables the operators to predict potential changes in network traffic, such as congestion and rerouting due to failures, before the present physical network is changed or updated. To appraise the effect of these external factors, the NWDR examines the impact of temporal bandwidth expansion and alterations of network configurations beforehand.
- (Scenario 2) NWDR for effective maintenance and operations for existing networks: An NWDR can be used as a virtual environment that enables the network to be managed and operated without directly interacting with physical entities. This can help reduce operating costs by digitally testing various network equipment configurations.
- (Scenario 3) NWDR for the assessment of new network concepts: Using the NWDR to create a model for a new network system and protocol enables rapid validation of configurations of network equipment that do not exist on a current network. This is especially helpful when rolling out mobile and ad hoc networks comprised of numerous IoT devices with limited computing capacity. The NWDR can then identify the best choice and location of these IoT devices, as well as manage the routing of the ad hoc network.

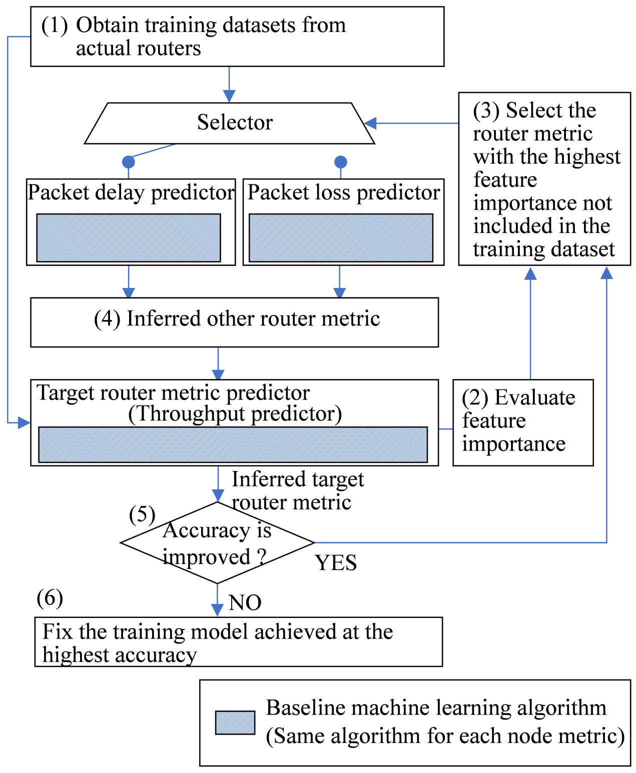

(1) Obtain training datasets from actual routers
Selector
Packet delay predictor
Packet loss predictor
(3) Select the router metric with the highest feature importance not included in the training dataset
(4) Inferred other router metric
Target router metric predictor (Throughput predictor)
(2) Evaluate feature importance
Inferred target router metric
(5) Accuracy is improved ?
YES
NO
(6) Fix the training model achieved at the highest accuracy
Baseline machine learning algorithm (Same algorithm for each node metric)

FIGURE 3. **Flowchart of the proposed method.**

C. ARCHITECTURE OF NETWORK DIGITAL REPLICA

The architecture of NWDR is illustrated in Fig. 2. The NWDR, which connects the physical network to a network application, digitally assesses the impact of new network configurations on the physical network before they are implemented on actual network nodes. It then supplies the network application with the inferred network metrics and controls the physical entities in accordance with the optimal configurations based on the inferred network metrics. Hence, the NWDR needs to reproduce the behavior of the actual network node when exposed to different external network conditions. To accomplish this, the NWDR creates network node models considering external factors such as traffic conditions on the physical network. The NWDR then uses the generated network node model to emulate the internal operation of physical network equipment under conditions not yet encountered on physical networks. Subsequently, the NWDR can evaluate the performance of the network node, including the throughput of the traffic and packet loss rate, despite unpredictable external conditions. Finally, the results of the NWDR inference are used to calculate the optimal network configurations and control network resources in the physical networks.

The primary focus of this paper is a fixed network scenario. In particular, we examine router modeling, which is a critical element of the fixed network, to digitally determine the best router configurations for traffic with unknown external conditions. The scenario assumed in this evaluation corresponds to "Restructuring Existing Networks" in Section III-B (Scenario 1). This scenario is related to network design and enables efficient network design by digitally assessing in advance whether the existing network design will meet service requirements by predicting potential changes in network traffic. For example, the NWDR supports expansion of network bandwidth based on current and near-future network conditions. Before updating or changing the configuration of a physical network, the NWDR evaluates the performance of network nodes while searching for appropriate network configurations in the digital domain, such as the allocated number of virtual CPUs on a server and the channel bandwidth between each node, for unknown external network conditions, and adjusts the physical network in accordance with the results of the node metrics predictions.

In the future, with the emergence of more dynamic traffic, such as 5G/6G, efficient network design will become even more important on carrier networks. In this context, we first propose a node modeling method for fixed networks that serve as the basis for carrier networks based on existing traffic conditions.

IV. NETWORK NODE MODELING IN NETWORK DIGITAL REPLICA

The NWDR emulates the behavior of physical network equipment based on a network node model. This model combines simulator-based network modeling and actual function-based node modeling based on hardware and software components (Fig. 2). This combination can digitally emulate the traffic conditions in the target area before implementing the actual equipment, thus enabling the performance of the target network equipment to be efficiently evaluated. Environmental factors, including user device specifications and network configurations, are considered when simulating the traffic from user devices in the simulator-based network modeling. On the other hand, actual function-based node modeling reproduces the inner workings and functioning of the target node. However, replicating the functions of physical nodes digitally necessitates the preparation of appropriate hardware logic and software programs, resulting in longer development time and a greater outlay to attain the function-based node model. Therefore, we propose a method of network node modeling to infer the performance of network nodes using the machine learning technique. This approach is advantageous to model the network node when the same hardware logic and software program of the actual network node cannot be created in the digital space.

A. PROPOSED ALGORITHM OF NETWORK NODE MODELING

We propose a method of machine learning-based network node modeling for the inference of the network node metrics, such as packet loss rate (P_{loss}), packet delay (P_{delay}), and packet throughput (P_{th}) with a balance between cost-effectiveness and accuracy [14]. This study concentrates on improving the accuracy of metric inference of black-boxed network nodes when only the network node configurations and traffic conditions are available as external conditions. To tackle this, we propose a machine learning-based node modeling method to increase the accuracy of network node metric inference. This is done by recursively appending the inferred other node metrics into the training datasets in line with feature importance. Fig. 3 describes the proposed method using the flowchart. In this flowchart, the proposed method aims to infer the target router metric (e.g., throughput) by incorporating inferred other router metrics (e.g., packet delay and packet loss) based on their respective feature importance. The main steps are as follows:

- 1) Obtain training datasets containing router configurations, input traffic, and measured router metrics.
- 2) Set the target router metric (e.g., throughput) from the measured router metrics and calculate feature importance for the training dataset, considering other router metrics (e.g., packet delay and packet loss).
- 3) Select the router metric with the highest feature importance that is not included in the training dataset.
- 4) Perform inference for the selected router metric (e.g., packet delay) and input the inferred values into the predictor for the target metric (throughput).
- 5) Evaluate whether the inference accuracy of the target metric (throughput) improves before and after adding the inferred other router metric (packet delay) to the training dataset. If the accuracy improves, continue by selecting the next router metric with high feature importance and repeat steps 3) to 5).
- 6) If the accuracy does not improve, fix the training model that achieved the highest accuracy (end).

This process iteratively incorporates router metrics with high feature importance into the training dataset to improve the accuracy of inferring the target router metric. If accuracy improves, this process continues. Otherwise, it stops with the best-performing training datasets so far. The proposed method applies the same baseline machine learning algorithm to each router metric. Therefore, the observed differences in the inference accuracy for the target metric are attributed to the specific selection of additional router metrics to the training datasets.

In addition to the flowchart, we present the detailed algorithm for the proposed method (Algorithm 1). Training datasets of router metrics; P_{loss}, P_{delay}, and P_{th} are obtained from a router. To measure these metrics, a traffic generator is deployed while changing the router configurations and levels of incoming traffic. Afterward, machine learning is utilized to create inference models for these metrics. This node modeling is applied to the datasets that include router configurations R, input traffic T, and the measured router metrics M. Then, the proposed method utilizes machine learning on the training datasets (R, T, and M) to infer the router metrics. The proposed algorithm recursively appends the inferred other router metrics to training datasets to increase the accuracy of router metric inference. The proposed algorithm utilizes the feature importance for each

Algorithm 1 Predictor of Router Metrics

```
Input: $M = \{m_i|_{i=1,\dots,I}\}$: $I$ dimensional datasets of the actual router metrics, $R = \{r_j|_{j=1,\dots,J}\}$: $J$ dimensional datasets of router configurations, $T$: Input traffic, $L$: Base machine learning algorithm
Variables: $\mathrm{FI}^{(m_i)} = \{\mathrm{fi}_1^{(m_i)}, \mathrm{fi}_2^{(m_i)}, \dots\}$: Ranked feature importance for $m_i$, $A$: Inferred router metrics to use as training datasets
Output: $P^{(m_i)} = \{P(r, t, m)|r \in R, t \in T, m \in M \setminus m_i\}$: Predictor for router metrics $m_i$
1  Measurement of the router metrics $M$ for router configurations $R$ and input traffic $T$
2  $A \leftarrow \emptyset$, $R^2_{\text{temp}} \leftarrow 0$
3  for each model of router metrics $m_i$ in $M$ do
4      $P^{(m_i)} \leftarrow L(R_{\text{train}}, T_{\text{train}}, m_i \cup A_{\text{train}})$ /* Train a predictive model */
5      $\mathrm{FI}^{(m_i)} \leftarrow \mathrm{FI}(P^{(m_i)})$ /* Calculate feature importance for $m_i$ */
6      $\hat{m}_i = P^{(m_i)}(R_{\text{test}}, T_{\text{test}}, A_{\text{test}})$ /* Predict router metric $m_i$ */
7      $R^2 = R^2(\hat{m}_i, m_i)$ /* Calculate $R^2$ of $m_i$ */
8      if $R^2_{temp} > R^2$ then
9          exit for
10     end
11     $R^2_{\text{temp}} = R^2$
12     $k = \operatorname{argmax}_{k \in I} \mathrm{fi}_k^{(m_i)}$ /* Select the biggest $\mathrm{fi}^{(m_i)}$ */
13     $\mathrm{FI}^{(m_i)} \leftarrow \mathrm{FI}^{(m_i)} \setminus \mathrm{fi}_k^{(m_i)}$ /* Exclude selected router metrics from FI */
14     $A \leftarrow A \cup \hat{m}_k$ /* Append the inferred other router metrics to training datasets */
   end
```

router metric model to see how relevant each router metric is to other ones. In this study, permutation feature importance [21] is used to calculate the feature importance. Permutation feature importance assesses the importance of individual features in a model by measuring the decrease in accuracy when the values of a single feature are randomly shuffled, disrupting the original relationships. If the model's error remains unchanged with shuffling, the model perceives that feature as less important for prediction.

Then, the inferred router metric with the highest feature importance is chosen as the training dataset. Then, the accuracy of each router metric inference is evaluated while the selected metric is incrementally appended to the training datasets. This calculation is repeated as long as the accuracy of router metric inference is improved by evaluating R^2 that represents the correlation between the measured router metric and the inferred one. R^2 is described as follows: $R^2 = 1 - \sum(y_i - \hat{y})^2 / \sum(y_i - \bar{y})^2$, where $\hat{y}$ and $\bar{y}$ are the inferred value

TABLE 1. Experimental conditions.

Router configurations (R)	Number of physical ports	2 to 8
	Number of flow entries	0.4 k to 770 k
	Number of allocated CPU cores	4 to 36
	Size of memory allocation (GB)	12 to 64
Input traffic (T)	Ethernet frame size (Bytes)	64 to 1518
	Number of traffic flows	0.4 k to 770 k
	Input traffic rate per flow (bps)	64 k to 80 M
Evaluation router metrics	Packet loss rate (per second)	
	Throughput (bits per second)	
	Packet delay (maximum per second)	

and mean measured value of router metrics y, respectively [22]. Therefore, $R^2 = 1$ indicates that the model perfectly fits the datasets. Once the accuracy reaches its peak, the training datasets with the inferred router metric are decided.

The proposed method is a supervised learning approach. Supervised learning is a machine learning method that learns relations from input to output based on labeled data constructed with determined input-output pairs. The proposed method is based on regression models to predict router metrics using router configurations and input traffic conditions as input datasets and measured router metrics such as throughput, packet loss rate, and packet delays as labeled data. The reasons why supervised learning was selected for the proposed method are as follows: To date, telecommunication carriers have kept performance data for each network equipment configuration through the operation of their carrier networks. This study aims to leverage these labeled performance data to model the network node. As a first step, we tried to determine if it is possible to digitally infer the performance of existing routers on a fixed network using the models trained on acquired labeled data. Furthermore, we aimed to enhance the accuracy of router metric inference by appending inferred other router metrics to the training datasets in addition to the labeled metrics data.

V. EVALUATION AND RESULTS

Here, we evaluated the accuracy of router metric inference by using the proposed method. We also evaluated the computational time of the proposed method because the proposed method tends to increase the computational time due to the characteristic of appending recursively inferred router metrics to the training datasets. Therefore, we evaluated the computational time to train the model (T_{train}) and the one to infer the router metrics (T_{infer}) whether these times are practical. Then, learning curves, residual plots, and predicted R^2 for each metric model were calculated to clarify whether the model using the proposed method is overfitting. The predicted R^2 serves as a measure to evaluate the predictive performance of the regression model [23]. Its importance lies in identifying potential overfitting problems within the model. A notable dissimilarity between the R^2 and the predicted R^2 indicates the presence of overfitting in the model. The predicted R^2 is calculated within observations not included in the model calculation. Specifically, the process involves (1) excluding a data point from the dataset, (2) calculating the regression equation, (3) calculating how well the model estimates the removed observation, and (4) repeating this for each data point. Finally, the predicted R^2 is expressed as follows: $R^2_{\text{pred}} = 1 - \sum(y_i - \hat{y}_{(i)})^2 / \sum(y_i - \bar{y})^2$, where $\hat{y}_{(i)}$ and $\bar{y}$ are the inferred value in which data point i is not included and mean measured value of router metrics y, respectively [23]. In general, overfitting happens when the model is too complex relative to the amount and noisiness of the training datasets. By increasing the number of features, which are the inferred router metrics, including the errors of inference, overfitting may occur due to the model being too complex for the training datasets. Finally, we evaluated the utilization of network resources for accommodating TCP flows based on the inferred router metrics by using the proposed method.

Fig. 4 shows the experimental environment. The proposed method applies machine learning to the performance measurements for four kinds of software routers: Cisco Cloud Services Router 1000V [24], Juniper vMX Virtual Router [25], Vector Packet Processor [26], and Kamuee router [27] using an x86-based server; two Xeon E5-2697 18-cores 2.30 GHz CPUs, and 192 GB RAM with 8 SFP+ ports. First, we measured the performance of these routers in the packet forwarding process in a laboratory environment to acquire training datasets under the conditions shown in Table 1. These conditions are set up to evaluate how much the relationship between the router configurations and traffic conditions affects an increase in packet processing delay and packet losses due to queue overflow. The Keysight Ixia platform with 8 SFP+ ports generates the traffic in the experiments. In total, 930 samples were acquired from these routers. In this experiment, we measured P_{loss} (per second), P_{delay} (maximum per second), and P_{th} (bits per second) for each T and R. We then used the measured router metrics to create a predictor of router metrics. The datasets were composed of R (number of physical ports, number of flow entries, number of allocated CPU cores, and size of memory allocation) and T (Ethernet frame size, average rate of input traffic, and number of traffic flows). The datasets were split into 70% training and 30% testing, respectively. In this experiment, we evaluated the 14 baseline machine learning algorithms in Table 2. These fundamental algorithms of machine learning are tree-based models [28], [29], [30], boosting-based models [31], [32], [33], linear regression-based models [34], [35], [36], [37], [38], a multi-layer perceptron neural network (NN) [39], and an extreme learning machine (ELM)-based model [40]. Table 2 lists these 14 base algorithms together with their corresponding abbreviations. We employed PyCaret [41] to carry out 12 learning algorithms: tree-based models, boosting-based models, and linear regression-based models [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38]: testing and optimizing a range of models and exploring feature importance. PyCaret, an auto machine learning technique, streamlines the process of model selection and hyper-parameter tuning, providing the

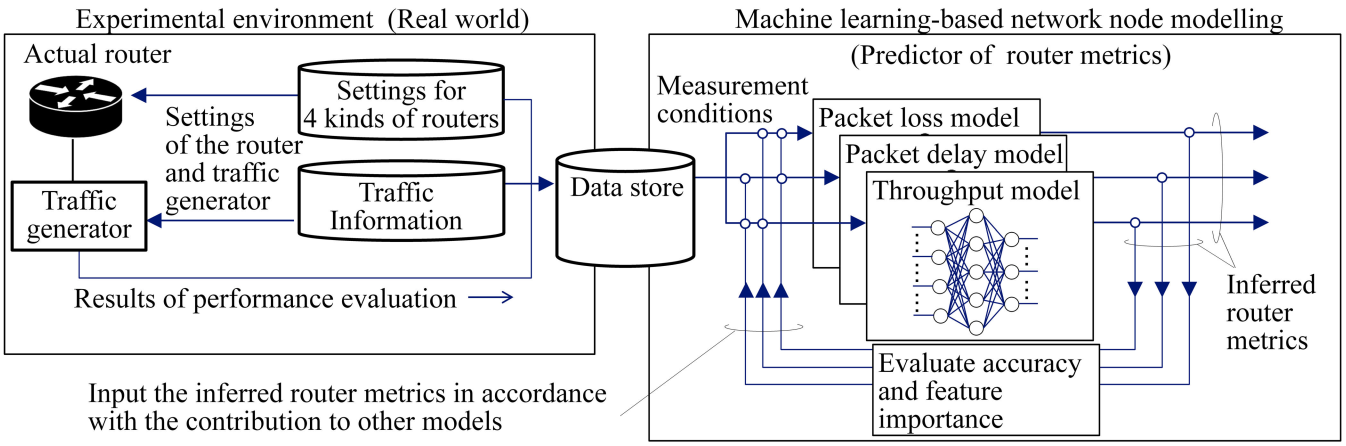

Experimental environment (Real world)
Actual router
Settings for 4 kinds of routers
Settings of the router and traffic generator
Traffic generator
Traffic Information
Data store
Results of performance evaluation
Machine learning-based network node modelling
(Predictor of router metrics)
Measurement conditions
Packet loss model
Packet delay model
Throughput model
Inferred router metrics
Evaluate accuracy and feature importance
Input the inferred router metrics in accordance with the contribution to other models

FIGURE 4. Experimental environment. The left side of the figure represents the laboratory environment where we obtained training datasets from actual routers. The right side shows the predictor of router metrics, which infers the router performance for the data obtained from actual routers. The proposed method incorporates the inferred router metrics with high feature importance into the training dataset to improve the accuracy of inferring the target router metric.

TABLE 2. Base Algorithms.

No	Abbr	Method	No	Abbr	Method
1	dt	Decision Tree Regressor [28]	8	lar	Least Angle Regression [35]
2	et	Extra-Trees Regressor [29]	9	llar	Lasso Least Angle Regression [35]
3	rf	Random Forest Regressor [30]	10	lr	Linear Regression [36]
4	ada	AdaBoost Regressor [31]	11	omp	Orthogonal Matching Pursuit [37]
5	gbr	Gradient Boosting Regressor [32]	12	rid	Ridge Regression [38]
6	lgbm	Light Gradient Boosting [33]	13	nn	Multi-layer Perceptron Neural Network [39]
7	en	Elastic Net [34]	14	oselm	Online Sequential Extreme Learning Machine [40]

benefit of time efficiency and producing a list of models with various metrics. On the other hand, 10 fully connected layers with 100 hidden nodes and a ReLU activation per layer were used for the NN model. For the ELM-based model, we used an online sequential extreme learning machine (OSELM) [40] as a primitive online learning method. For the OSELM, the number of hidden nodes was set to 100, and the sigmoid activation function was chosen. The evaluation with the OSELM was conducted for 20 trials since the OSELM had been initialed with random weights. Then, we evaluated the feature importance for each router metrics model and R^2 to decide which inferred router metrics to append to the training datasets. To increase the accuracy of router metric inference, we recursively appended the inferred other router metric to the training datasets in line with the feature importance. Each model was trained for 1000 epochs on a Nvidia Tesla P100 card with 16 GB of memory.

A. EVALUATION OF FEATURE IMPORTANCE FOR EACH ROUTER METRIC MODEL

Fig. 5 indicates feature importance with the top 5 features for (a) P_{loss} model, (b) P_{th} model, and (c) P_{delay} model. Including other router metrics in each feature importance implies that the accuracy of each router metric inference can be increased by incorporating these inferred router metrics into the training datasets. With respect to the P_{loss} model, P_{delay} was the most influential factor in the inference of P_{loss}, followed by P_{th} and the number of allocated CPU cores in this condition. This suggested that when the input traffic rate surpasses the processing speed to forward the packets to the outputs, the finite packet buffer of the router leads to packet loss and an

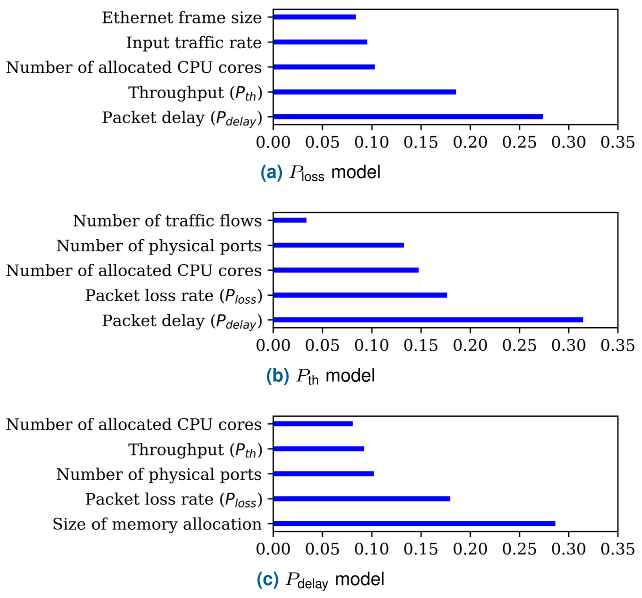


FIGURE 5. Feature importance for each model.

TABLE 3. Results of R^2, T_{train}, and T_{infer} for P_{loss} model.

Model	Vanilla model			With the proposed method		
	R^2	T_{train} (sec)	T_{infer} (sec)	R^2	T_{train} (sec)	T_{infer} (sec)
dt	0.87	11.98	0.19	0.95	23.38	0.20
et	0.83	158.45	0.30	0.95	311.65	0.32
rf	0.84	180.83	0.31	0.95	358.63	0.31
ada	0.72	60.32	0.20	0.91	128.66	0.21
gbr	0.85	26.68	0.20	0.95	51.56	0.21
lgbm	0.84	15.56	0.21	0.96	29.48	0.22
en	0.61	10.56	0.19	0.81	21.43	0.20
lar	0.61	1.68	0.19	0.81	3.43	0.20
llar	0.47	1.72	0.20	0.76	3.50	0.20
lr	0.61	0.80	0.20	0.81	1.60	0.20
omp	0.61	1.36	0.19	0.81	2.73	0.20
rid	0.61	8.24	0.19	0.81	18.16	0.20
nn	0.83	19.57	0.17	0.93	39.10	0.19
oselm	0.84	0.33	0.01	0.94	0.70	0.01

TABLE 4. Results of R^2, T_{train}, and T_{infer} for P_{th} model.

Model	Vanilla model			With the proposed method		
	R^2	T_{train} (sec)	T_{infer} (sec)	R^2	T_{train} (sec)	T_{infer} (sec)
dt	0.90	11.86	0.19	0.98	21.79	0.19
et	0.89	246.82	0.30	0.98	490.32	0.32
rf	0.92	266.29	0.31	0.98	549.87	0.31
ada	0.82	60.18	0.20	0.94	132.39	0.21
gbr	0.90	27.85	0.19	0.98	57.14	0.20
lgbm	0.84	16.37	0.20	0.98	33.99	0.22
en	0.79	11.32	0.19	0.90	21.64	0.21
lar	0.79	1.76	0.19	0.90	3.49	0.21
llar	0.79	1.77	0.20	0.89	3.51	0.21
lr	0.79	0.81	0.19	0.90	1.60	0.20
omp	0.79	1.55	0.19	0.90	3.08	0.21
rid	0.79	8.24	0.19	0.90	17.67	0.20
nn	0.88	19.68	0.21	0.98	40.24	0.21
oselm	0.78	0.35	0.01	0.90	0.72	0.01

TABLE 5. Results of R^2, T_{train}, and T_{infer} for P_{delay} model.

Model	Vanilla model			With the proposed method		
	R^2	T_{train} (sec)	T_{infer} (sec)	R^2	T_{train} (sec)	T_{infer} (sec)
dt	0.42	11.42	0.20	0.82	22.72	0.20
et	0.68	209.00	0.31	0.80	408.25	0.32
rf	0.68	240.47	0.31	0.83	462.85	0.31
ada	0.65	60.60	0.21	0.74	123.34	0.21
gbr	0.80	29.76	0.21	0.85	55.35	0.21
lgbm	0.75	15.02	0.21	0.84	29.72	0.22
en	0.54	8.03	0.21	0.56	17.19	0.20
lar	0.55	1.70	0.20	0.56	3.37	0.20
llar	0.55	1.75	0.20	0.56	3.45	0.20
lr	0.55	0.80	0.20	0.56	1.62	0.20
omp	0.55	1.46	0.20	0.56	3.08	0.20
rid	0.52	10.16	0.20	0.55	17.98	0.20
nn	0.75	25.23	0.26	0.80	50.90	0.28
oselm	0.68	0.33	0.01	0.75	0.72	0.01

increase in queuing delay. In contrast, the most significant factor for P_{delay} was the amount of memory allocation, as this was the primary element to store and process the incoming packets in the virtual routers. To account for this, the proposed method can increase the accuracy of router metrics inference by appending the inferred other router metric with the most substantial effect on the training datasets in descending order.

TABLE 6. Results of R^2 for each router metric on each training dataset.

Metrics model	Training datasets		R^2
Packet loss rate (P_{loss})	(1)	Router configurations + input traffic	0.84
	(2)	(1) + Inferred P_{delay}	0.96
	(3)	(1) + Inferred P_{th}	0.88
	(4)	(2) + Inferred P_{delay}	0.96
Throughput (P_{th})	(1)	Router configurations + input traffic	0.84
	(2)	(1) + Inferred P_{delay}	0.98
	(3)	(1) + Inferred P_{loss}	0.92
	(4)	(2) + Inferred P_{delay}	0.97
Packet delay (P_{delay})	(1)	Router configurations + input traffic	0.75
	(2)	(1) + Inferred P_{loss}	0.84
	(3)	(1) + Inferred P_{th}	0.78
	(4)	(3) + Inferred P_{loss}	0.88

B. EVALUATION OF THE ACCURACY OF ROUTER METRICS INFERENCE AND COMPUTATIONAL TIME

To evaluate the effectiveness of applying the proposed method to machine learning algorithms, we evaluated R^2 for the 14 machine learning algorithms. Tables 3, 4, and 5 show the results of R^2, T_{train}, and T_{infer} for the P_{loss}, P_{th}, and P_{delay} models, respectively. The training datasets for the proposed method had the inferred other router metric, which was the highest feature importance for each model; i.e., the training datasets for P_{loss}, P_{th}, and P_{delay} models had the inferred P_{delay}, P_{delay}, and P_{loss}, respectively. Overall, it can be seen that R^2 is improved by applying the proposed method to each algorithm. The improvement in R^2 for P_{loss} was 9.1% - 61.7%, that for P_{th} was 6.5% - 16.6%, and that for P_{delay} was 1.8% - 95.2%. Even compared with the largest R^2 in the conventional method for each router metric, the improvement in R^2 with the proposed method for P_{loss} with dt [28] was 9.1%, that for P_{th} with rf [30] was 6.5%, and that for P_{delay} with gbr [32] was 6.2%. Both OSELM and NN models with the proposed method achieved a comparable improvement in the accuracy of inferred router metric compared to other machine learning algorithms. However, they were not the best baseline algorithms in terms of accuracy. The improvement in R^2 using the proposed method for P_{loss}, P_{th}, and P_{delay} with NN and with OSELM were 12.0%, 11.4%, 6.6%, 11.9%, 15.3%, and 10.3%, respectively.

On the other hand, the proposed method tends to increase computational time, T_{train} and T_{infer} due to the characteristic of appending recursively inferred router metrics to the training datasets. This increase in computational time depends on the number of times that the inferred other router metrics are repeatedly input to the training datasets. For example, the inferred P_{th} with the largest R^2 in the proposed method uses inferred P_{delay}, so the process of making the inferred model is repeated twice. In the results, T_{train} took nearly twice as long as the application of the vanilla algorithm, as shown in Tables 3, 4, and 5. Nevertheless, T_{train} for the proposed method took only a few minutes at most. Especially, T_{train} and T_{infer} with OSELM using the proposed method are the fastest among these 14 machine learning algorithms: 0.72 and 0.01 seconds, respectively. This is because the

OSELM does not require iterative parameter optimization, unlike traditional machine learning algorithms such as the NN model, which require iterative training. This approach directly calculates the output weights using a random matrix, which leads to significantly reduced computational time. Regarding the lgbm, which had good overall performance for R^2, T_{train} with the proposed method was less than almost 30 seconds. This increase of several tens of seconds of T_{train} is not a problem because these models are created in advance before being applied to the actual environment. Rather, the impact factor for the application to an actual environment is T_{infer}, which affects processing delay to change the network configurations when models are applied to an actual environment. In this experiment, T_{infer} was 0.4 seconds or less. On the assumed fixed network, this value is not a big problem for calculating and setting network configurations to the physical network.

In the following evaluation, we applied the proposed method to the lgbm, which had the most evenly improved accuracy for each router metric and practical T_{train}, by applying the proposed method with these 14 algorithms.

Table 6 shows the results of R^2 for the inferred router metrics on the training datasets when recursively appending the inferred other router metrics. We found that R^2 of P_{loss} was increased by 14.2%, rising from 0.84 to 0.96, by appending the inferred P_{delay}, which is the top-ranked feature importance to the training datasets. Similarly, the R^2 of P_{th} was increased by 16.6%, rising from 0.84 to 0.98, and that of P_{delay} was increased by 14.6%, rising from 0.75 to 0.88. Despite applying the inferred router metrics to the training datasets, not all metrics showed an improvement in accuracy. The R^2 values for row (2) of P_{loss} and row (2) of P_{th} in Table 6 plateaued, indicating that the inferred router metrics appended to the training datasets contain the inference errors that affect the accuracy of the other router metric inference. Particularly, this is known as the regression tree, which is the building block of many tree-based ensemble methods; the lgbm used for this evaluation also uses tree-based learning algorithms, that depend on both the quality and quantity of the training datasets [42]. In this scenario, the increase in R^2 resulting from the expansion of the training datasets with the inferred router metric is comparable to the decrease in R^2 caused by the errors of the inferred router metric. Therefore, it is essential to balance the improvement in the inferred router metric with the expansion of the training datasets and the deterioration due to errors in the inferred router metric.

As outlined in Algorithm 1, the inferred router metrics are repeatedly added based on their feature importance. However, it is necessary to stop adding these inferred router metrics to the training dataset in accordance with the calculated R^2. The maximum R^2 values for the inferred P_{th}, the inferred P_{loss}, and the inferred P_{delay} were reached with one, one, and two inferred other metrics appended to the training datasets, respectively.

TABLE 7. Results of R^2_{pred} in comparison to R^2.

Target metrics	Vanilla lgbm		With the proposed method	
	R^2	R^2_{pred}	R^2	R^2_{pred}
Packet loss rate	0.84	0.74	0.96	0.92
Throughput	0.84	0.78	0.98	0.96
Packet delay	0.75	0.65	0.88	0.80

C. EVALUATION OF LEARNING CURVE AND VALIDATION CURVE FOR EACH MODEL

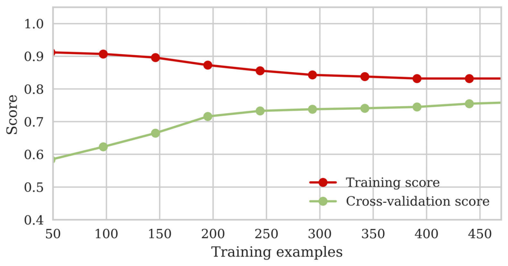
Fig. 6 shows the learning curve of each model using the conventional method (vanilla lgbm) and the proposed method. We used a 10-fold cross-validation to obtain and construct the learning curve. The red line represents the training score, while the green line represents the cross-validation score. The conventional model shows a high bias, as both the training and validation scores converge to a low value. In addition, a noticeable gap between the training and validation scores indicates a high variance. On the other hand, the proposed method offers a relatively better bias-variance trade-off compared to the model using the conventional method. We find that adding inferred other router metrics to the training datasets does not result in worse overfitting compared to the model using the conventional method (vanilla lgbm) on the given dataset.

D. CORRELATION BETWEEN MEASURED AND INFERRED ROUTER METRICS

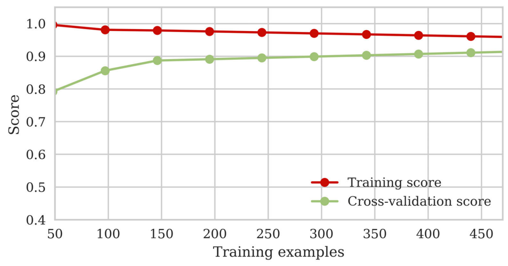
Fig. 7 shows the correlation between measured and inferred router metrics: (a), (b) the inferred P_{loss}; (c), (d) the inferred P_{th}; and (e), (f) the inferred P_{delay} ((a), (c), (e): with vanilla lgbm (conventional method), (b), (d), (f): with the proposed method) on the training datasets with; (a), (c), (e) T and R; (b), (d) T, R, and the inferred P_{delay}; and (f) T, R, the inferred P_{th}, and the inferred P_{loss}, respectively. As you can see, Fig. 7 shows that the inferred router metrics on the training datasets with the inferred other router metrics had a higher correlation than those on the training datasets without the inferred other router metrics. However, each inferred router metric still had some errors. In particular, the proposed method did not significantly improve the accuracy of the inferred P_{delay} caused by fluctuations such as packet jitter, which includes unpredictable factors. However, the proposed method improved the accuracy of the inferred router metrics by selecting and appending the inferred other router metrics to the training datasets according to their feature importance.

E. EVALUATION OF RESIDUAL PLOTS AND PREDICTED R^2 FOR EACH MODEL

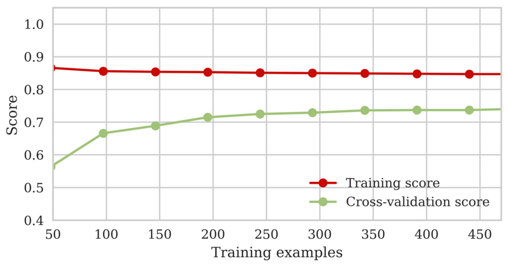
To evaluate the adequacy of the regression model using the proposed method, we evaluated the residual plot and R^2_{pred}. A residual plot shows the difference between the predicted and actual values. Fig. 8 shows the residual plot



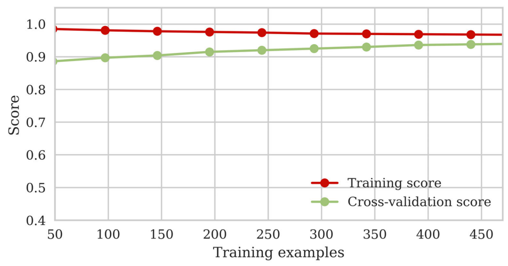
(a) P_{loss} model using the conventional method (Vanilla lgbm)



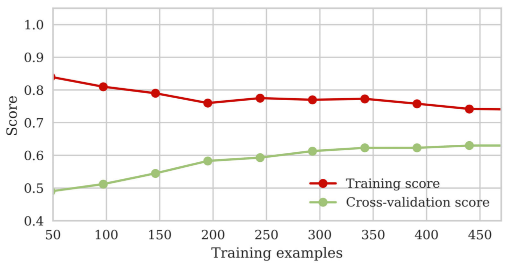
(b) P_{loss} model using the proposed method



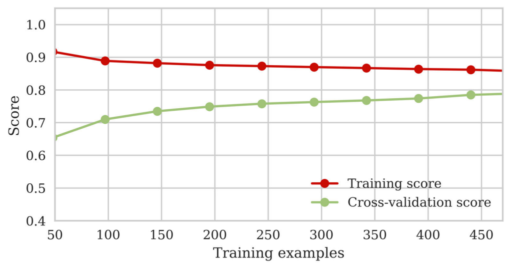
(c) P_{th} model using the conventional method (Vanilla lgbm)



(d) P_{th} model using the proposed method



(e) P_{delay} model using the conventional method (Vanilla lgbm)



(f) P_{delay} model using the proposed method

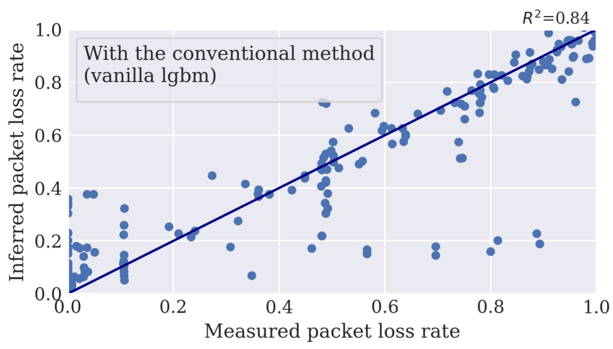
FIGURE 6. Learning curve for each model.

with distribution for inferring the router metrics with vanilla lgbm (conventional method) and the proposed method. Although there are a few incorrect inferences, the residual plots with the proposed method show that most of the fitted data are closely scattered around the zero residual line, confirming the better capability of the prediction model compared to the conventional method. Conversely, the residual plots with the conventional method show systematic patterns, suggesting the presence of underlying factors that influence the dependent variable but are not adequately accounted for in the model with the conventional method.

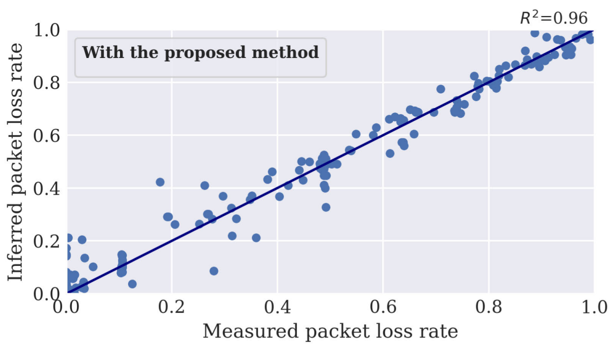
Table 7 shows results of R^2_{pred} in comparison to R^2. We found that the R^2_{pred} with the conventional method (vanilla lgbm) for P_{loss} was 0.74, that for P_{th} was 0.78, and that for P_{delay} was 0.65. On the other hand, the R^2_{pred} with the proposed method for P_{loss} was 0.92, that for P_{th} was 0.96, and that for P_{delay} was 0.80. Although the R^2_{pred} is lower than the individual R^2, the R^2_{pred} value using the proposed method is still closer to the R^2 value compared to the conventional method. This indicates that the proposed method does not result in worse overfitting, and its predictive ability is better compared to the conventional method.

F. EVALUATION FOR THE UTILIZATION OF NETWORK RESOURCES

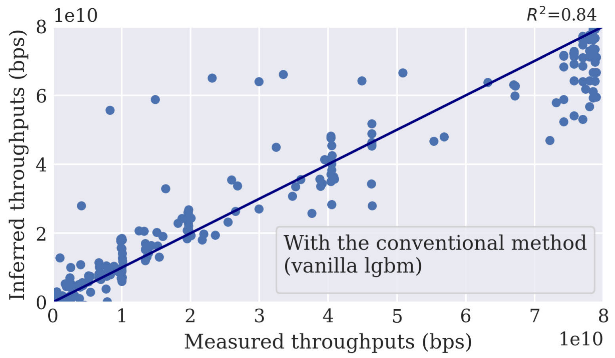
We evaluated the effect of the proposed method for improving the accuracy of router metric inference in network design. In particular, we evaluated how much errors in the inference of P_{delay} affect each traffic flow on a fixed network designed by NWDR. The evaluation index was the throughput of traffic flows under network configurations calculated by the inferred router metrics. We assume that a network service using TCP is added to existing traffic flows on the network, which has a traffic policer to guarantee the required quality of service for each flow. Network equipment can be effectively utilized if the number of traffic flows increases under the conditions of the same network resources. Here, we simulated a simple network scenario using the OPNET [43] simulator to evaluate TCP throughput on a traffic policer whose parameters are set based on inferred router metrics calculated by the proposed method. Fig. 9 shows the simulation setup. We consider the accommodation of TCP CUBIC flows [44] as it is the most widely used congestion control algorithm and is the default in Linux. This scenario has a router with a traffic policer to guarantee the required quality of service for each TCP flow. We used a bufferless token bucket [45], which is



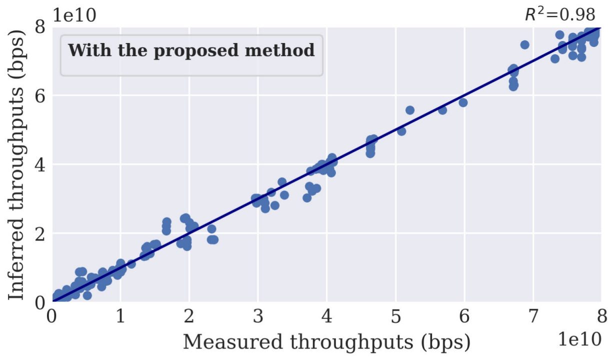
(a) Inferred P_{loss} with vanilla lgbm



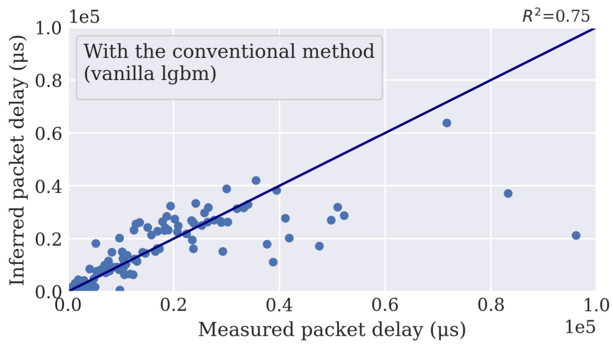
(b) Inferred P_{loss} with the proposed method using the inferred P_{delay}



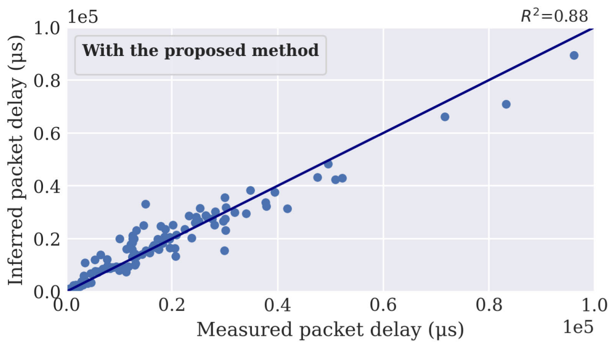
(c) Inferred P_{th} with vanilla lgbm



(d) Inferred P_{th} with the proposed method using inferred P_{delay}



(e) Inferred P_{delay} with vanilla lgbm



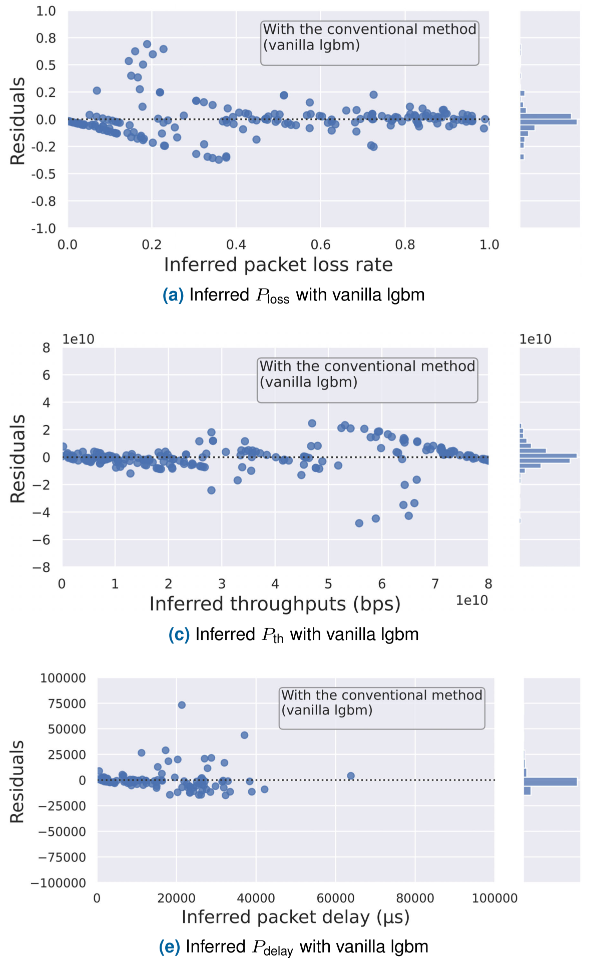
(f) Inferred P_{delay} with the proposed method using the inferred P_{th} and the inferred P_{loss}

FIGURE 7. Correlation between measured and inferred router metrics.

a commonly used implementation. A token bucket policer has a regulator that checks for enough tokens in the bucket, which is filled at a throttle rate. When a packet arrives, the controller evaluates the capacity of the bucket. If the number of tokens exceeds or equals the packet length, the token bucket policer delivers the packet and subtracts tokens from the bucket. On the other hand, if the number of tokens is less than the packet length, the packet is discarded. If the parameters, especially the bucket size, are not tuned, TCP throughput will decrease [45]. In this evaluation, we use the bucket size B described in (1) [45], where D is the round-trip time (RTT) and R is the throttling rate. This bucket size is tuned to accommodate TCP CUBIC. To calculate D, we estimate RTT using the inferred P_{delay} calculated by the proposed method. Therefore, the larger inferred error for the processing delay may affect TCP throughput. In other words, if the accuracy of the inferred P_{delay} is poor, it can result in an unnecessarily large bucket size being prepared, leading to wastage of network bandwidth.

$$B = \frac{0.072}{D}(RD)^{\frac{4}{3}} \tag{1}$$

Here, we define the accommodated TCP throughput ratio (R_{Flow}) defined as T_P/T_C, where T_P and T_C are the TCP throughput using the router metrics inferred by the proposed method and that of the conventional method, respectively. When R_{Flow} is greater than 1, the proposed method can improve the TCP throughput compared with the conventional method. The simulation network is a typical single bottleneck dumbbell topology with 40 senders and one receiver. For each



(a) Inferred P_{loss} with vanilla lgbm
(b) Inferred P_{loss} with the proposed method
(c) Inferred P_{th} with vanilla lgbm
(d) Inferred P_{th} with the proposed method
(e) Inferred P_{delay} with vanilla lgbm
(f) Inferred P_{delay} with the proposed method

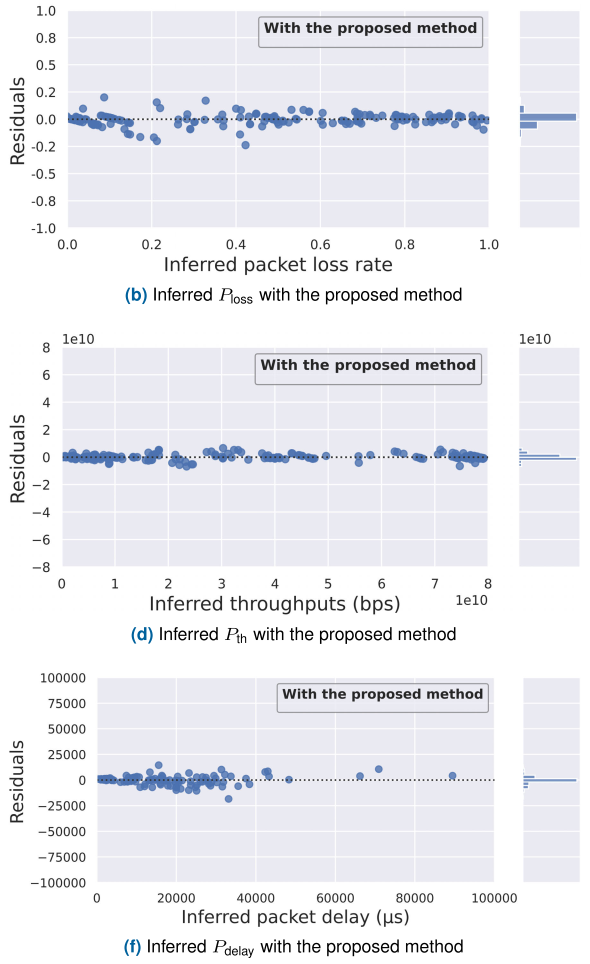


FIGURE 8. Residuals versus inferred router metrics.

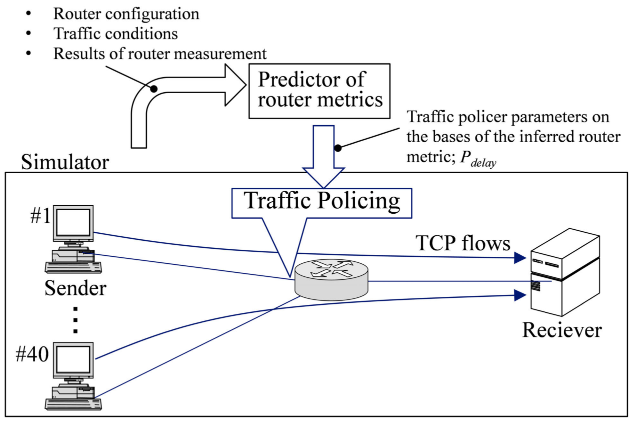

Router configuration
Traffic conditions
Results of router measurement
Predictor of router metrics
Traffic policer parameters on the bases of the inferred router metric; P_{delay}
Simulator
Traffic Policing
#1
TCP flows
Sender
#40
Reciever

FIGURE 9. Simulation setup to evaluate accommodated TCP flow with a traffic policer, whose parameters are set based on the inferred router metric; P_{delay} using the proposed method.

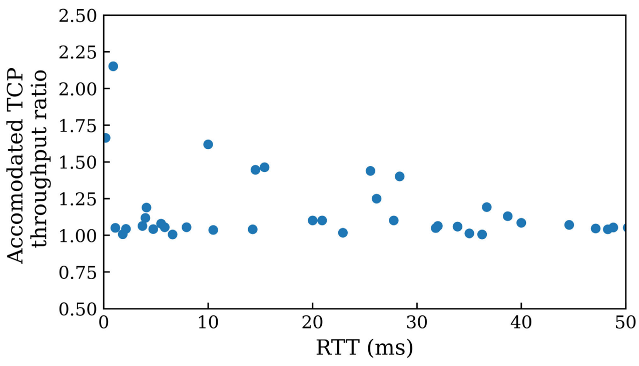


FIGURE 10. Accommodated TCP throughput ratio (R_{Flow}).

sender, there was a TCP flow that ended at the receiver. The links on the sender side were set as the fixed delays ranging from 0.1 to 25 ms. In other words, the RTT of each TCP flow was calculated based on the sum of this fixed delay and the inferred P_{delay} on a router. Other parameters were as follows: throttling rate for the TCP flow, 10 Mbps, and Maximum Segment Size, 1460 bytes.

Fig. 10 shows R_{Flow}. Overall, R_{Flow} was greater than 1 for every RTT. This means the TCP throughput can be increased compared with the conventional method. In particular, the smaller the RTT is, the larger the inferred error of P_{delay} for the RTT. In this experimental condition, R_{Flow} at maximum became 2.15 when RTT was about 1 ms. This means that the proposed method can accommodate twice as many TCP flows as the conventional method under the conditions of the same network resources. This is because the conventional method has a relatively larger inferred error of P_{delay} for the smaller RTT. Therefore, an appropriate bucket size was not set to the token bucket policer. As a result, packets are dropped more than necessary with the smaller RTT, and TCP throughput decreased. On the other hand, the proposed method can set the appropriate bucket size by using the improved inferred P_{delay} at a router and improve TCP throughput. Therefore, the proposed method could reduce the network bandwidth required to absorb the errors of the inferred router metrics. Considering the above results, the proposed method can effectively design network resources by improving the accuracy of router metric inference.

VI. DISCUSSION

Improving the accuracy of network node metric inference will support the implementation of optimal network configurations in increasingly complex network systems, such as 5G/6G supported by multiple carriers and network disaggregation technologies. Before changing or updating the configuration of a physical network, the NWDR evaluates the network performance for unknown external network conditions in digital space to evaluate whether network requirements will be met. Therefore, understanding the impact of node metric inference on network configuration is a critical step in developing appropriate network extensions. In this study, we proposed a machine learning-based network node modeling, which selects and appends the inferred other router metrics to the training datasets recursively according to feature importance, which increases the accuracy of the router metric inferences of P_{loss}, P_{th}, and P_{delay}.

A. REVIEW OF THE PROPOSED METHOD VIA EXPERIMENTAL RESULTS

Here, we review the proposed method through experimental results. The proposed method has the characteristic of appending inferred router metrics to the training datasets recursively. Under the evaluation conditions of this study, the addition of one or two inferred other router metrics was sufficient to improve the accuracy of the inferred router metrics for throughput, packet loss rate, and packet delay by up to 9.1% compared to the largest R^2 in the conventional method, as shown in Section V-B. On the other hand, the side effects of appending recursively inferred router metrics to the training datasets with the proposed method resulted in increased computational time and the possibility of overfitting. We found that the increased computational time to infer the router metrics was about 0.4 seconds or less, as shown in Section V-B. This computational time was not a major problem for computing and setting network configurations to the physical network in the assumed fixed network. Moreover, the overfitting is not getting worse than the vanilla lgbm model, as shown in Section V-C and V-E. We found that this improvement in the accuracy of the inferred router metric led to increased efficiency in network design, and made it possible to effectively allocate network resources in an assumed fixed network where a traffic policer is implemented, and calculate bucket size based on the inferred packet delay. The proposed method reduces the network bandwidth to absorb inferred router metrics errors by up to 50% compared to the conventional method, as discussed in Section V-F.

B. SELECTION OF FEATURE IMPORTANCE

Here, we explain the selection of feature importance, which is essential in the proposed method. The more available features, the better the chances of constructing an accurate network node model if the intricate relationships between the router metric model and the features are clear. Therefore, appending the inferred other router metrics to the training datasets is crucial to enrich the training datasets. If the training dataset is insufficient, the router metrics model cannot learn correctly due to the overfitting problem. In addition, if a model is too complicated, such as having too many features in a small sample size, overfitting can also occur. Therefore, attaching more features than necessary can cause overfitting. Thus, the inferred router metrics must be carefully appended to the training datasets. In this study, we used feature importance [21] for each router metrics model to decide which inferred router metrics to append to the training datasets. Then, we recursively appended the inferred other router metrics to the training datasets according to feature importance while evaluating the accuracy of the router metric inference.

C. LIMITATIONS OF THE PROPOSED METHOD

The proposed method can support the inference of router metrics for interpolation, but the inference for extrapolation is required as a further study. In particular, it is known that methods based on NN are not good at extrapolation [46]. Also, the proposed method requires further study for application to network nodes with environmental changes, such as a non-terrestrial network (NTN). In this paper, we conducted a network simulation assuming how the inference errors of the proposed method affect each traffic flow on a piece of current fixed network equipment: a packet router implementing a traffic policer. The proposed method has some effectiveness in conventional fixed network design. However, application to an NTN, such as satellite communications or high-altitude platform systems, will be required to achieve cost-effective and high-capacity connectivity in future 6G networks [47]. An NTN is generally exposed to environmental changes, such as weather conditions, which cause variations in link quality. This is different from a conventional fixed network. Therefore, network node modeling algorithms need to be robust in changing environments such as those of NTNs and be able to update network configurations immediately.

VII. CHALLENGES AND FUTURE DIRECTIONS

There are still considerable research topics that require additional exploration and unresolved issues that need to be addressed. Here, we outline topics that may be useful for future investigation in this study. These topics are grouped into three areas.

- 1) *Network adaptation in changing external environments*: Future 6G networks will require NTN applications for efficient and high-capacity connectivity. However, the link quality of NTNs can vary due to external factors such as weather conditions. Therefore, developing network node modeling algorithms that can adapt to changes in the external environment is a critical task for the future. One possible method to achieve this is to use algorithms related to ELM [48] that can learn in real time. The ELM is a machine learning algorithm known for its fast training speed and low computational complexity. It has the potential to enable accurate inference of network node performance and rapid evaluation of network requirements through instantaneous retraining. The algorithms associated with ELM can facilitate rapid adaptation to changing network conditions. Another approach is to use a physics-based approach to capture the multiplexing characteristics of network traffic. For example, based on the idea of integrating physical laws and data-driven models [49], this approach also has the potential to improve data-driven models by using governing equations that represent the multiplexing of packet traffic. These approaches could improve the performance of network nodes on networks with changing external environments such as NTNs.

  In addition, the proposed method relies on models trained through supervised learning on labeled datasets from existing network environments, which may limit its applicability in dynamically changing environments such as NTNs. In particular, the generalizability of the created model is crucial in NTN use cases, where unobserved scenarios arise from changes in the external environment that are not present in the training dataset. To address this challenge, a reinforcement learning method has been proposed to enable unmanned aerial vehicles (UAVs) to operate autonomously and adapt intelligently to rapidly changing conditions [50]. Therefore, when dealing with environments such as NTNs that undergo frequent changes and face challenges in obtaining labeled data, it is crucial to incorporate reinforcement learning by designing rewards that can adapt to unknown environments.
- 2) *Adaptation to field data*: Field data, such as that collected from commercial carrier networks, often contains various types of noise. In addition, collecting large amounts of data for different environmental patterns to train the network node model can be difficult and expensive. Therefore, the key to success is to use clean data obtained in laboratory or simulation environments. To achieve this, techniques such as transfer learning [51], [52] are considered highly effective. Transfer learning is a powerful machine learning technique that uses knowledge from one domain to another to improve the performance of a learning algorithm. Previous studies have successfully applied transfer learning to models for inferring network traffic [51] and detecting network intrusions [52]. Using transfer learning techniques based on simulated environments makes it possible to improve the accuracy of network node metric inference even for field data that may contain noise and other uncertainties.
- 3) *Modeling of performance at the node components for network disaggregation*: Network disaggregation techniques [4] enable the combination of modular hardware and modular software from different vendors, providing the flexibility to meet service requirements when configuring network equipment. In the case of routers, for example, it is possible to combine line cards and switch cards from different vendors. However, the challenge remains to model each module and digitally evaluate performance at the component level.

Since the proposed method is based on a model learned from a dataset acquired from existing network equipment, it is expected to have limitations when dealing with new network equipment models that exhibit different behaviors. Therefore, it becomes crucial to leverage reinforcement learning when modeling new network equipment without labeled data. This approach may enable network equipment modeling even without labeled data. In addition, there is a possibility of applying the concept of ensemble learning [20] to different network node metrics, where different algorithms can leverage distinct network node metric characteristics and the relationship of a given dataset to improve the accuracy of network node metric inference.

VIII. CONCLUSION

We proposed machine learning-based network node modeling for a network digital replica (NWDR). The NWDR generates optimal network configurations for unknown external conditions by creating network node models in digital space. This study focuses on improving the accuracy of inference for the metrics of black-boxed network nodes when only the network node configurations and traffic conditions are available as external conditions. To address this issue, we propose a machine learning-based network node modeling method to increase the accuracy of network node metric inference. This is done by recursively appending the inferred other metrics into the training datasets according to feature importance. The proposed method could improve the coefficient of determination (R^2) for inferred router metrics, throughput, packet loss rate, and packet delay by up to 9.1% compared to the largest R^2 in the conventional method. We also evaluated the impact of adding inferred router metrics to the training datasets, a feature of the proposed method, on the increase in computational time and the possibility of overfitting. For the assumed fixed network, we find that the increase in computational time for inferring router metrics is less than about 0.4 seconds, which is not a significant problem for calculating and setting the network configuration to the physical network, and the overfitting is not getting worse compared to the model using vanilla lgbm. Moreover, we evaluated the effect of the proposed method for improving the accuracy of router metric inference on network design. We assumed a fixed network to accommodate TCP flows with a token bucket policer to guarantee the required quality of service, and whose bucket size is controlled by the NWDR. We found that the proposed method can reduce the network bandwidth for the absorption of the errors of the inferred router metrics by up to half compared with the conventional method.

Limitations of the proposed method include reliance on supervised learning models trained on labeled datasets for fixed network environments, incomplete consideration of various types of noise in the dataset, and reliance on data acquired from existing network equipment. Thus, future research can provide valuable insights and further develop this area of study by exploring adaptation to networks with changing external environments, adaptation to field data, and modeling the performance of new node components for network disaggregation.

REFERENCES

[1] G. Singh and G. Kaur, "Artificial intelligence, Internet of Things, and communication networks," in *Artificial Intelligence to Solve Pervasive Internet of Things Issues*. Cambridge, MA, USA: Academic, 2021.

[2] I. U. Khan, I. M. Qureshi, M. A. Aziz, T. A. Cheema, and S. B. H. Shah, "Smart IoT control-based nature inspired energy efficient routing protocol for flying ad hoc network (FANET)," *IEEE Access*, vol. 8, pp. 56371–56378, 2020.

[3] V. Bhardwaj, N. Kaur, S. Vashisht, and S. Jain, "SecRIP: Secure and reliable intercluster routing protocol for efficient data transmission in flying ad hoc networks," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, p. e4068, Jun. 2021.

[4] K. Ishii, R. Matsumoto, T. Inoue, and S. Namiki, "Disaggregated optical-layer switching for optically composable disaggregated computing [invited]," *J. Opt. Commun. Netw.*, vol. 15, no. 1, pp. A11–A25, Jan. 2023.

[5] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems*. Berlin, Germany: Springer, 2017.

[6] GE Digital. *Digital Twin*. Accessed: Dec. 10, 2023. [Online]. Available: https://www.ge.com/digital/applications/digital-twin/

[7] M. Borgo, S. Elliott, T. Ghandchi, and I. Stothers, "Virtual sensing of wheel position in ground-steering systems for aircraft using digital twins," in *Proc. 38th IMAC, Conf. Expo. Struct. Dyn.*, 2020, pp. 1–12.

[8] J. Deng, Q. Zheng, G. Liu, J. Bai, K. Tian, C. Sun, Y. Yan, and Y. Liu, "A digital twin approach for self-optimization of mobile networks," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Mar. 2021, pp. 1–6.

[9] Nokia. *Digital Design and Deployment*. Accessed: Dec. 10, 2023. [Online]. Available: https://www.nokia.com/networks/mobile-networks/deployment-services/

[10] M.-H. Guo, T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng, and S.-M. Hu, "Attention mechanisms in computer vision: A survey," 2021, *arXiv:2111.07624*.

[11] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter, "Survey on reinforcement learning for language processing," *Artif. Intell. Rev.*, vol. 56, pp. 1543–1575, Feb. 2023, doi: 10.1007/s10462-022-10205-5.

[12] S. A. Kumar, T. D. A. Kumar, N. M. Beeraka, G. V. Pujar, M. Singh, H. S. N. Akshatha, and M. Bhagyalalitha, "Machine learning and deep learning in data-driven decision making of drug discovery and challenges in high-quality data acquisition in the pharmaceutical industry," *Future Med. Chem.*, vol. 14, no. 4, pp. 245–270, Feb. 2022.

[13] K. Hattori, T. Korikawa, C. Takasaki, H. Oowada, M. Shimizu, and N. Takaya, "Network digital replica using neural-network-based network node modeling," in *Proc. IEEE 8th Int. Conf. Netw. Softwarization (NetSoft)*, Jun. 2022, pp. 287–291.

[14] K. Hattori, T. Korikawa, C. Takasaki, H. Oowada, and M. Shimizu, "Recursive router metrics prediction using ML-based node modeling for network digital replica," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2022, pp. 1006–1012.

[15] L. Diez, C. Hervella, and R. Agüero, "Understanding the performance of flexible functional split in 5G vRAN controllers: A Markov chain-based model," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 456–468, Mar. 2021.

[16] *ns-3*. Accessed: Dec. 10, 2023. [Online]. Available: https://www.nsnam.org

[17] Keysight. *QualNet*. Accessed: Dec. 10, 2023. [Online]. Available: https://www.keysight.com/us/en/products/network-test/network-modeling.html

[18] J. Lai, J. Tian, K. Zhang, Z. Yang, and D. Jiang, "Network emulation as a service (NEaaS): Towards a cloud-based network emulation platform," *Mobile Netw. Appl.*, vol. 26, no. 2, pp. 766–780, Apr. 2021.

[19] 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 J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2260–2270, Oct. 2020.

[20] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99129–99149, 2022.

[21] C. Molnar, *Interpretable Machine Learning*. Lulu.com, 2020.

[22] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Comput. Sci.*, vol. 7, p. e623, Jul. 2021.

[23] J. Qian and S. Li, "Model adequacy checking for applying harmonic regression to assessment quality control," *ETS Res. Rep. Ser.*, vol. 2021, no. 1, pp. 1–26, Dec. 2021.

[24] Cisco. *Cisco Cloud Services Router 1000v Series*. Accessed: Dec. 10, 2023. [Online]. Available: https://www.cisco.com/c/en/us/products/routers/cloud-services-router-1000v-series/index.html

[25] Juniper. *vMX Virtual Router*. Accessed: Dec. 10, 2023. [Online]. Available: https://www.juniper.net/gb/en/products/routers/mx-series/vmx-virtual-router-datasheet.html

[26] D. Barach, L. Linguaglossa, D. Marion, P. Pfister, S. Pontarelli, and D. Rossi, "High-speed software data plane via vectorized packet processing," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 97–103, Dec. 2018.

[27] Y. Ohara, H. Shirokura, A. D. Banik, Y. Yamagishi, and K. Kyunghwan, "Kamuee: An IP packet forwarding engine for multi-hundred-gigabit software-based networks," in *Proc. Internet Conf.*, 2018, pp. 1–10.

[28] Rahul, A. Gupta, A. Bansal, and K. Roy, "Solar energy prediction using decision tree regressor," in *Proc. 5th Int. Conf. Intell. Comput. Control Syst.*, 2021, pp. 489–495.

[29] S. R. Polamuri*, D. K. Srinivasi, and D. A. K. Mohan, "Stock market prices prediction using random forest and extra tree regression," *Int. J. Recent Technol. Eng.*, vol. 8, no. 3, pp. 1224–1228, Sep. 2019.

[30] Z. El Mrabet, N. Sugunaraj, P. Ranganathan, and S. Abhyankar, "Random forest regressor-based approach for detecting fault location and duration in power systems," *Sensors*, vol. 22, no. 2, p. 458, Jan. 2022.

[31] G. A. Busari and D. H. Lim, "Crude oil price prediction: A comparison between AdaBoost-LSTM and AdaBoost-GRU for improving forecasting performance," *Comput. Chem. Eng.*, vol. 155, Dec. 2021, Art. no. 107513.

[32] A. Keprate and R. M. C. Ratnayake, "Using gradient boosting regressor to predict stress intensity factor of a crack propagating in small bore piping," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage. (IEEM)*, Dec. 2017, pp. 1331–1336.

[33] J. Fan, X. Ma, L. Wu, F. Zhang, X. Yu, and W. Zeng, "Light gradient boosting machine: An efficient soft computing model for estimating daily reference evapotranspiration with local and external meteorological data," *Agricult. Water Manage.*, vol. 225, Nov. 2019, Art. no. 105758.

[34] F. Amini and G. Hu, "A two-layer feature selection method using genetic algorithm and elastic net," *Expert Syst. Appl.*, vol. 166, Mar. 2021, Art. no. 114072.

[35] H. Nikaein, A. Sheikhi, and S. Gazor, "Target detection in passive radar sensors using least angle regression," *IEEE Sensors J.*, vol. 21, no. 4, pp. 4533–4542, Feb. 2021.

[36] D. Maulud and A. M. Abdulazeez, "A review on linear regression comprehensive in machine learning," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 4, pp. 140–147, Dec. 2020.

[37] J. Tropp and A. C. Gilbert, "Signal recovery from partial information via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[38] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, Feb. 2000.

[39] Z. Zhao, S. Xu, B. H. Kang, M. M. J. Kabir, Y. Liu, and R. Wasinger, "Investigation and improvement of multi-layer perceptron neural networks for credit scoring," *Expert Syst. Appl.*, vol. 42, no. 7, pp. 3508–3516, May 2015.

[40] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.

[41] *PyCaret: An Open Source, Low-Code Machine Learning Library in Python*. Accessed: Dec. 10, 2023. [Online]. Available: https://www.pycaret.org

[42] M. Sebbanü, R. NockO, J. Chauchat, and R. Rakotomalala, "Impact of learning set quality and size on decision tree performances," *Int. J. Comput. Sci. Secur.*, vol. 1, no. 1, p. 85, 2000.

[43] *OPNET*. Accessed: Dec. 10, 2023. [Online]. Available: https://www.riverbed.com

[44] P. Bruhn, M. Kuehlewind, and M. Muehleisen, "Performance and improvements of TCP CUBIC in low-delay cellular networks," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, Jun. 2022, pp. 1–9.

[45] D. Shan, P. Zhang, W. Jiang, H. Li, and F. Ren, "Towards the fairness of traffic policer," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2021, pp. 1–10.

[46] 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," 2020, *arXiv:2009.11848*.

[47] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G networks: Use cases and technologies," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 55–61, Mar. 2020.

[48] J. Zhang, Y. Li, W. Xiao, and Z. Zhang, "Non-iterative and fast deep learning: Multilayer extreme learning machines," *J. Franklin Inst.*, vol. 357, no. 13, pp. 8925–8955, Sep. 2020.

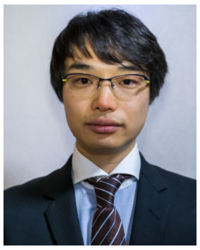
[49] J. Zhang, Y. Zhao, F. Shone, Z. Li, A. F. Frangi, S. Q. Xie, and Z.-Q. Zhang, "Physics-informed deep learning for musculoskeletal modeling: Predicting muscle forces and joint kinematics from surface EMG," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 31, pp. 484–493, 2023.

[50] A. H. Arani, P. Hu, and Y. Zhu, "HAPS-UAV-enabled heterogeneous networks: A deep reinforcement learning approach," 2023, *arXiv:2303.12883*.

[51] X. Chen, J. Wang, H. Li, Y. T. Xu, D. Wu, X. Liu, G. Dudek, T. Lee, and I. Park, "One for all: Traffic prediction at heterogeneous 5G edge with data-efficient transfer learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6.

[52] E. Mahdavi, A. Fanian, A. Mirzaei, and Z. Taghiyarrenani, "ITL-IDS: Incremental transfer learning for intrusion detection systems," *Knowl.-Based Syst.*, vol. 253, Oct. 2022, Art. no. 109542.

KYOTA HATTORI received the B.E. degree in applied physics and the M.E. degree in computational science and engineering from Nagoya University, in 2004 and 2006, respectively, and the Ph.D. degree in information science and technology from Hokkaido University, in 2019. In 2006, he joined the NTT Network Service Systems Laboratories, NTT Corporation, Tokyo, Japan, where he has been engaged in research on traffic flow control and optical network architecture. He is currently a Senior Research Engineer with the NTT Network Service Systems Laboratories, NTT Corporation. He received the Award for SC4 Best Paper from the 2012 Photonics in Switching (PS2012) and the Young Researcher's Award from IEICE, in 2013. He holds the CISSP Certification.



TOMOHIRO KORIKAWA (Member, IEEE) received the B.S. and M.S. degrees in physics from Waseda University, Tokyo, Japan, in 2012 and 2014, respectively, and the Ph.D. degree in informatics from Kyoto University, Kyoto, Japan, in 2021. In 2014, he joined NTT Corporation, where he is currently a Researcher with the NTT Network Service Systems Laboratories. His research interests include network architecture, network design, and network digital twins.

CHIKAKO TAKASAKI received the B.S. and M.S. degrees in informatics from Ochanomizu University, Tokyo, Japan, in 2019 and 2021, respectively. In 2021, she joined NTT Corporation, where she is currently a Researcher with the NTT Network Service Systems Laboratories. Her research interests include network architecture, network digital twins, and machine learning for networks.

HIDENARI OOWADA received the M.E. degree from Tohoku University, Japan, in 2002. He is currently a Senior Research Engineer with the NTT Network Service Systems Laboratories, NTT Corporation, Tokyo, Japan.