






# Automatic Data Generation and Optimization for Digital Twin Network

Mei Li , Cheng Zhou , Lu Lu , *Member, IEEE*, Yan Zhang , *Fellow, IEEE*, Tao Sun , *Member, IEEE*, Danyang Chen, Hongwei Yang, and Zhiqiang Li

**Abstract**—With the rise of new applications such as AR/VR, cloud gaming, and vehicular networks, traditional network management solutions are no longer cost-effective. Digital Twin Network (DTN) creates a real-time virtual twin of the physical network, which improves the network's stability, security, and operational efficiency. AI models have been used to model complex network environments in DTN, whose quality mainly depends on the model architecture and data. This paper proposes an automatic data generation and optimization method for DTN called AutoOPT, which focuses on generating and optimizing data for data-driven DTN AI modeling through data-centric AI. The data generation stage generates data in small networks based on scale-independent indicators, which helps DTN AI models generalize to large networks. The data optimization stage automatically filters out high-quality data through seed sample selection and incremental optimization, which helps enhance the accuracy and generalization of DTN AI models. We apply AutoOPT to the DTN performance modeling scenario and evaluate it on simulated and real network data. The experimental results show that AutoOPT is more cost-efficient than state-of-the-art solutions while achieving similar results, and it can automatically select high-quality data for scenarios that require data quality improvement.

**Index Terms**—Data generation, data optimization, data-centric AI, digital twin network (DTN), network performance.

## I. INTRODUCTION

New applications such as AR/VR, cloud gaming, and vehicular networks are emerging as technology advances, which make networks more complex and challenging to manage. Traditional network management solutions are not cost-effective for meeting the requirements of these new applications, such as ultra-low deterministic delay and real-time changes in network topology. Digital twin [2], [3] technology offers a solution to this problem. A digital twin is a virtual representation of a physical object that can sense, present, verify, and simulate its physical counterpart. Digital Twin Network (DTN) [4], [5] utilizes digital twin technology to create real-time mirrors of physical networks, which provides a safe and cost-effective environment

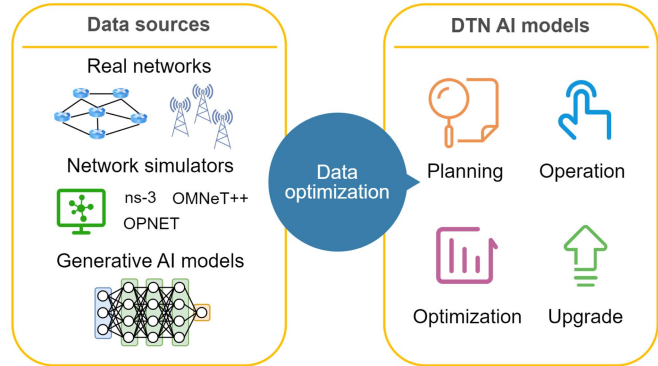


Fig. 1. Data-driven DTN AI modeling.

for network operators to perform operations such as network optimization, what-if analysis, and troubleshooting. DTN can be used for a variety of networks, such as wireless networks [6], [7], optical networks [8], data center networks [9], Internet of Things (IoT) networks [10], and vehicular networks [11], [12].

The use of Artificial Intelligence (AI) models, especially neural networks (NNs), has been demonstrated to be effective in modeling complex network environments for tasks such as performance evaluation [13], [14], traffic prediction [15], resource allocation [16], [17], and service self-healing [18]. AI-based network modeling helps build real-time, lightweight, and highly accurate DTN. The quality of AI models depends mainly on the model architecture and data. Existing studies primarily focused on improving models rather than enhancing data. It is important to note that NN-based AI is a data-driven technique, meaning that data quality directly affects the accuracy and generalization of the model. As shown in Fig. 1, to train high-quality DTN AI models, various data sources such as real networks, network simulators, and generative AI models can be used. Real networks provide high-value real data, but the quantity and type may be limited. Network simulators (such as ns-3 [19] and OMNeT++ [20]) and AI-generated content (AIGC) technology can be utilized to generate simulated network data. In order to obtain high-quality data (especially training data), it is crucial to study how to generate simulated data to address the issue of real data shortage and how to optimize the data from various sources.

The role of data has recently been highlighted by the emerging concept of data-centric AI [21]. This paper proposes an automatic data generation and optimization method for DTN

Received 10 October 2023; revised 19 December 2024; accepted 19 December 2024. Date of publication 26 December 2024; date of current version 6 February 2025. (Corresponding author: Lu Lu.)

Mei Li, Cheng Zhou, Lu Lu, Tao Sun, Danyang Chen, Hongwei Yang, and Zhiqiang Li are with China Mobile Research Institute, Beijing 100120, China (e-mail: limeijy@chinamobile.com; zhouchengy@chinamobile.com; lulu@chinamobile.com; suntao@chinamobile.com; chendanyang@chinamobile.com; yanghongwei@chinamobile.com; lizhiqiangy@chinamobile.com).

Yan Zhang is with the Department of Informatics, University of Oslo, 0313 Oslo, Norway (e-mail: yanzhang@ieee.org).

Digital Object Identifier 10.1109/TSC.2024.3522504

called AutoOPT, which follows the data-centric AI paradigm to improve data quality in a reliable, efficient, and systematic way for NN-based DTN AI modeling. Existing studies on network modeling mainly follow the model-centric AI paradigm, focusing on the model architecture. Differing from existing studies, AutoOPT emphasizes a shift from model improvements to ensuring data quality and reliability. AutoOPT consists of data generation (Stage 1) and data optimization (Stage 2). Stage 1 generates simulated network data based on scale-independent indicators in small networks, which helps the model generalize to large networks with more nodes, higher link capacities, and longer paths. Network configurations (e.g., network topology, routing policy, and traffic matrix) are generated and fed into data generators (e.g., network simulators and generative AI models) to generate the simulated candidate data. Stage 2 automatically selects high-quality data from multi-source candidate data (including real and simulated network data) through seed sample selection and incremental optimization, which helps the model be effectively trained using high-quality data to improve accuracy and generalization. Seed samples are selected using feature extraction, clustering, calculation of cluster centers and nearest neighbors, and expert knowledge verification. The remaining candidate samples are evaluated by the filter model and Out-of-Distribution (OOD) detection. The filter model is used for easy and hard example mining. Easy and hard samples are considered valid and added to the training data. The training data and filter model are optimized iteratively until the high-quality training data meets the requirements.

To show AutoOPT's practical applicability and role in DTN, we introduce three typical use cases: configuration evaluation and optimization in data center networks, performance prediction in IP bearer networks, and task offloading in vehicular networks. We conduct experiments to apply AutoOPT to the DTN performance modeling scenario. We use the state-of-the-art (SOTA) network performance model RouteNet-Fermi (RouteNet-F) [13] as the model architecture. The evaluation metric is the Mean Average Percentage Error (MAPE) / Mean Absolute Error (MAE) of the per-path/flow mean delay, jitter, and loss on networks. The entire process of AutoOPT is evaluated by training the model on the datasets generated by AutoOPT and testing the trained model on the simulated datasets provided by GNet Challenge 2022 [22]. The experimental results show that AutoOPT is more cost-efficient than the winning solutions of the challenge while achieving similar results. We further evaluate the performance of data optimization of AutoOPT with real network datasets [22] for scenarios that only need data quality improvement. The experimental results show that AutoOPT can automatically select high-quality data.

The main contributions of this paper are summarized as follows:

- 1) This paper proposes AutoOPT, an automatic data generation and optimization method that follows the data-centric AI paradigm to improve data quality for NN-based DTN AI modeling.
- 2) The data generation stage generates data based on scale-independent indicators. The data optimization stage filters out high-quality data through seed sample selection and incremental optimization.

- 3) Use cases in data center networks, IP bearer networks, and vehicular networks demonstrate AutoOPT's practical applicability. Extensive experiments on both simulated and real network data validate the effectiveness of AutoOPT.

The rest of the paper is organized as follows. Section II briefly reviews related work. Section III presents the design details of AutoOPT. Section IV presents the role of AutoOPT in DTN and some typical use cases. Section V describes experimental results. Section VI draws some conclusions and discusses directions for future work.

## II. RELATED WORK

DTN [4], [5] utilizes digital twin [2] [3] technology to create real-time mirrors of physical networks, which enables network operators to simulate, optimize, verify, and control networks more efficiently. DTN can improve upon the limitations of physical networks by enhancing their systematic simulation, optimization, verification, and control capabilities. Before deploying new configurations and policies in the physical network, they need to be fully validated. DTN can be used for a variety of networks (such as wireless networks [6], [7], optical networks [8], data center networks [9], IoT networks [10], and vehicular networks [11], [12]) to provide a safe and cost-effective environment for network operators to perform operations including network optimization, what-if analysis, and troubleshooting, etc. Building a real-time, lightweight, and high-precision DTN requires the study of key technologies such as network data collection and storage, digital twin modeling, interactive mapping, intelligent analysis, and decision-making.

Performance modeling is one of the key technologies of DTN, which is critical to DTN and is used in various typical network management scenarios such as planning, operation, optimization, and upgrade. With the advancement of AI technologies, some researchers have utilized data-driven techniques, particularly NNs, to create DTN performance evaluation models [23]. For instance, xWeaver [24] designs an NN to analyze traffic patterns and learn the relationship between traffic and topology. Suzuki et al. [25] propose to combine GNN with semi-supervised learning to estimate end-to-end delay. MimicNet [26] utilizes deep-learning-based internal models and flow-based feeder models to achieve quick and precise performance evaluation for large data center networks. xNet [27] is a data-driven network modeling framework that models the network with graph representations and configurable GNN blocks. RouteNet-F [13] uses GNN to estimate the path mean delay, jitter, and loss by modeling the complex relationship between topology, routing, and traffic. m3 [14] uses machine learning to predict the tail latency performance of data center networks.

Instead of focusing on the model architecture, data-centric AI [21] focuses on enhancing data using technologies such as data augmentation, self-supervision, data cleaning, data selection, and data privacy. For instance, data augmentation [28], [29] can generate additional augmented samples, which reduces the expenses of data collection and labeling. The self-supervised models [30], [31] can be learned without manual labels or features, which reduces the effort of labeling and feature engineering. Data selection methods [32], [33] can select the most valuable samples, which reduces the pain caused by Big Data.

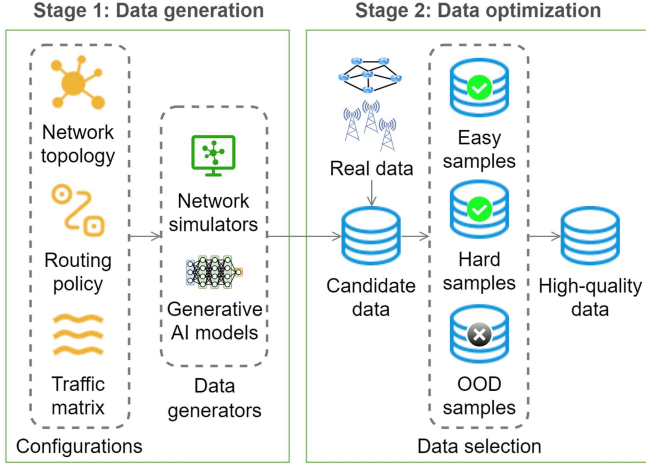


Fig. 2. Framework of AutoOPT. The workflow of AutoOPT consists of data generation and data optimization.

NN-based AI is a data-driven technique. Data quality directly impacts the accuracy and generalization capability of the model. Most existing studies focus on the model architecture (such as the number, size, and type of layers in an NN). In this paper, we focus on improving DTN AI models from the data perspective by utilizing data-centric AI.

### III. METHODS

#### A. Framework

AutoOPT is a method to generate and optimize data for NN-based DTN AI models through data-centric AI. AutoOPT concentrates on designing reliable, efficient, and systematic approaches to improve data. The framework of AutoOPT is shown in Fig. 2, which consists of two stages: data generation (Stage 1) and data optimization (Stage 2).

Stage 1 aims to generate simulated network data to address the shortage of real network data. To achieve this, Stage 1 generates network configurations and feeds them into data generators to generate the candidate data. Network configurations usually consist of network topology, routing policy, and traffic matrix, which need to be diverse to cover as many scenarios as possible based on scale-independent indicators. Data generators can be network simulators (e.g., ns-3 [19] and OMNeT++ [20]) and generative AI models (e.g., GPT [34]-like models for networks). The design details of Stage 1 are presented in Section III-C. Stage 1 can generate data in small networks, enabling DTN AI models to generalize to large networks with more nodes, higher link capacities, and longer paths.

Stage 2 aims to filter out high-quality data from candidate data from various sources. Candidate data includes simulated network data generated in Stage 1 and real data collected from real networks. To be considered high-quality, the data need to be accurate, diverse, and similar to real network data. We can verify this by leveraging expert knowledge, such as the ranges of delay, queue utilization, link utilization, and average port occupancy. AutoOPT investigates the candidate data to filter

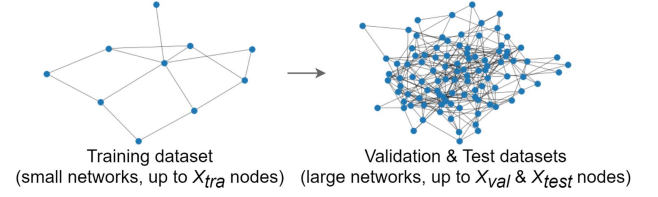


Fig. 3. Problem statement. The fundamental part of the problem is to generate training datasets in small networks that can help the model generalize to large networks.

out easy, hard, and OOD samples. Hard samples refer to samples that are difficult for the model to predict accurately. Exposing the model to more hard samples during training can enhance the model's ability to perform better on such samples. Easy and hard samples are considered valid and added to the high-quality data, while the OOD samples are removed as they are deemed invalid. Section III-D describes the design details of Stage 2. Stage 2 can systematically improve data quality to help DTN AI models be trained effectively using high-quality data, ultimately enhancing the accuracy and generalization of the models.

#### B. Problem Statement

Consider a NN-based DTN AI model  $M$ , a validation dataset  $D_{val}$ , and a test dataset  $D_{test}$ . The architecture of the model  $M$  is fixed. The training dataset  $D_{tra}$  can contain up to  $N_{max}$  samples from small networks with up to  $X_{tra}$  nodes (no isolated nodes). Each sample is a tuple containing network topology, routing policy, and traffic matrix. The link capacity of the training dataset  $D_{tra}$  falls between the range of  $[L_l, L_u]$ . The validation dataset  $D_{val}$  and the test dataset  $D_{test}$  consist of samples from large networks with up to  $X_{val}$  and  $X_{test}$  nodes, which include larger link capacities and longer paths. The test dataset  $D_{test}$  contains samples that have similar distributions to the validation dataset  $D_{val}$  and is not seen in training. These constraints (1) describe the limitations in real-world communication networks. Obtaining data from large networks is often difficult or impractical.

$$|D_{tra}| \leq N_{max}, X_{tra} < X_{val}, X_{tra} < X_{test}, D_{val} \sim D_{test} \quad (1)$$

As shown in Fig. 3, the problem to be solved is how to generate a high-quality training dataset  $D_{tra}$  that meets the given constraints. The model  $M$  trained on the training dataset  $D_{tra}$  from small networks can obtain high accuracy and generalization when tested on the test dataset  $D_{test}$  from large networks with more nodes, higher link capacities, and longer paths.

We take the DTN performance modeling as an example scenario and use the SOTA network performance model RouteNet-F [13] as the model architecture of  $M$ . RouteNet-F is a Graph Neural Networking (GNN) based model for network performance evaluation, which takes the network state as the input and flow-level performance metrics (e.g., delay, jitter, and loss) as the output. RouteNet-F supports network configurations with arbitrary network topology, routing policy, and traffic matrix. This paper focuses on predicting the per-path/flow mean delay,



jitter, and loss on networks. The evaluation objective is to test the accuracy of the model  $M$  on the test dataset  $D_{test}$  after being trained on the generated training dataset  $D_{tra}$ .

The evaluation metric used in this problem to measure the accuracy of the trained model  $M$  is the MAPE/MAE of the per-path/flow mean delay, jitter, and loss on networks. The MAPE (2) represents the mean of the absolute percentage errors of predictions, while the MAE (3) represents the mean of absolute errors of predictions. In these equations,  $y_i$  denotes the actual value,  $\hat{y}_i$  denotes the predicted value, and  $n$  represents the number of samples. A lower MAPE/MAE value indicates a higher model prediction accuracy.

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \right| \quad (2)$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad (3)$$

### C. Data Generation (Stage 1)

This section will introduce how to generate network configurations, including the network topology, routing policy, and traffic matrix. Once these configurations are determined, they can be fed into data generators, such as existing network simulators or generative AI models, to generate the simulated network data.

1) *Network Topology*: Based on the constraints for the training dataset (1), each network topology can contain up to  $X_{tra}$  nodes. Having more nodes in a network topology may result in more flows per sample and diverse path length distributions. To achieve this, we set the number of nodes per topology to  $\lceil 0.8 \times X_{tra} \rceil$  to  $X_{tra}$ . We utilize the Power-Law Out-Degree algorithm [35] to generate network topologies, with parameters based on real-world topologies in the Internet Topology Zoo [36].

If the flow rate exceeds the link bandwidth or the bandwidth assigned for the flow, the packet is temporarily stored in the node buffer. We set the node buffer size to a uniform distribution within the range  $[B_l, B_u]$  bits, where  $B_l$  and  $B_u$  are determined based on the analysis of the validation dataset to have a similar range. A larger node buffer size may result in a larger delay and potentially lower packet loss rate. The node scheduling policy determines the time and order of packet transmission, which is randomly selected from the following policies:

- *First In First Out (FIFO)*: Packets are transmitted in the same order as they arrive in the queue.
- *Strict Priority (SP)*: Packets are mapped to a priority level. The packet with high priority is always transmitted before the packet with low priority.
- *Weighted Fair Queuing (WFQ)*: WFQ assigns a weight to each queue, allocates each flow to a queue, and offers fair output bandwidth sharing based on weights.
- *Deficit Round Robin (DRR)*: DRR is an efficient (with  $O(1)$  complexity) and fair scheduling policy, which uses a deficit accumulator to determine whether a packet can be transmitted within the scheduling cycle.

Having a larger link capacity may result in a smaller delay and less congestion. To cover various link loads and potential scenarios, we set the link capacity to be proportional to the total average bandwidth of all flows that pass through the link. Furthermore, the link capacity falls within the range of  $[L_l, L_u]$  based on training dataset constraints.

2) *Routing Policy*: Routing protocols rely on routing policies to determine the path of packets from source to destination. For a network topology, the default routing configuration is the shortest path policy. In addition, to add randomness, a series of variants of the shortest path policy are also generated as alternatives.

- *Default*: All links in the topology have the same weight (set to 1). We then use the Dijkstra algorithm [37] to generate the shortest path configuration, which relies on Breadth-First Search (BFS) to find the single source shortest path in a weighted digraph.
- *Variants*: We randomly select a few links (a link can be chosen more than once) and add a small weight. We then use the Dijkstra algorithm to create a series of variants of the default shortest path configuration based on the weighted digraph. These variants add some randomness to the routing configuration, which can help to cover longer paths and larger delays.

3) *Traffic Matrix*: The traffic matrix plays a crucial role in modeling network performance. It serves as a network map that outlines the traffic entering and leaving the network, including the source, destination, traffic distribution, etc.

To cover low to high loads, we use variable traffic intensity ( $TI$ ) to generate traffic matrix configurations. In each source-destination pair, the average bandwidth ( $AvgBW$ ) is modeled using the following equation:

$$AvgBW(s, d) = \frac{U(0.5, 1) \times TI}{N - 1}, \forall s, d \in \text{nodes}, s \neq d \quad (4)$$

where  $U(0.5, 1)$  denotes a uniform distribution on the interval  $[0.5, 1]$ ,  $N$  denotes the number of nodes in the network topology.

Packet sizes, probabilities of packet sizes, and type of service (ToS) parameters are generated to have similar distributions based on the validation dataset analysis.

We model the arrival of packets for each source-destination pair using one of the following time distributions. Based on the actual situation and validation dataset analysis, the Poisson distribution has a higher proportion than other distributions.

- *Poisson*: Packets are generated at a node according to an independent process, with an average of  $\lambda$  packets per time slot. The packets' arrival intervals follow an exponential distribution. The Poisson model is widely used to simulate traffic in telecommunication networks.
- *Constant Bit Rate (CBR)*: The packets are generated at a constant rate at a node. The CBR traffic model can accurately simulate CBR services, such as voice and video-on-demand.
- *ON-OFF*: The node enters the ON and OFF states alternately. In the ON state, the node sends packets at a constant rate. In the OFF state, the node does not generate packets.

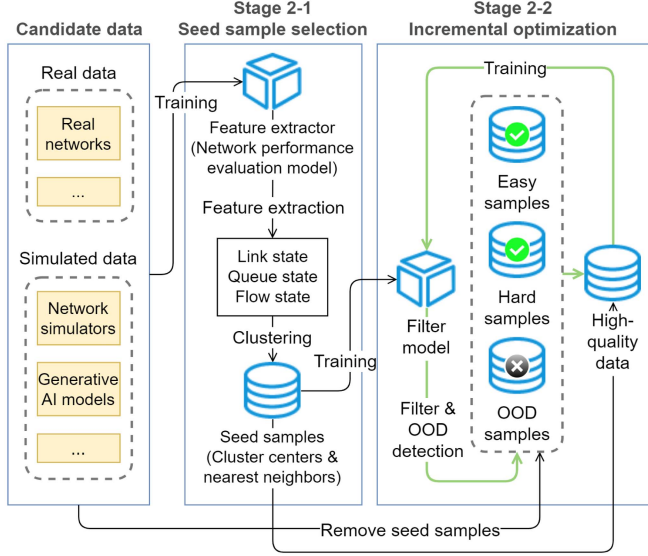


Fig. 4. Workflow of data optimization stage (Stage 2). Stage 2 aims to optimize the data from various sources to filter out high-quality data, including seed sample selection (Stage 2-1) and incremental optimization (Stage 2-2).

The ON-OFF model can be used to simulate self-similar traffic.

#### D. Data Optimization (Stage 2)

As shown in Fig. 4, Stage 2 aims to optimize the candidate data from various sources by filtering out high-quality data, which is divided into two steps: seed sample selection (Stage 2-1) and incremental optimization (Stage 2-2). During Stage 2-1, AutoOPT selects seed samples based on the feature extractor (a network performance evaluation model). The seed samples are then used as the initial training dataset for Stage 2-2. Then, the remaining candidate samples are investigated by the filter model to screen out the easy, hard, and OOD samples. These processes are repeated (as indicated by the green arrows in Fig. 4) until the high-quality data meets the requirements.

1) *Seed Sample Selection (Stage 2-1)*: In Stage 2-1, AutoOPT selects seed samples from the candidate data through feature extraction, clustering, calculation of cluster centers and nearest neighbors, and expert knowledge verification. Algorithm 1 presents the pseudo-code of Stage 2-1 SELECTSEED().

We obtained the candidate dataset  $D_{can} = \{C_i\}_{i=1}^{N_c}$  containing  $N_c$  samples obtained in Stage 1. In Stage 2-1, the feature extractor (the RouteNet-F [13] model with modified output) is trained on the candidate dataset  $D_{can}$ . To prevent the model from over-fitting, we use the validation dataset  $D_{val}$  for validation. The trained feature extractor is denoted as  $M_{ext}$ , which can achieve low MAPE (2) / MAE (3) on the validation dataset  $D_{val}$ . RouteNet-F is a custom GNN model for network performance evaluation. It defines a circular dependency (5) between the states of links  $\mathcal{L}$ , queues  $\mathcal{Q}$  and flows  $\mathcal{F}$ , where  $h_l$ ,  $h_q$  and  $h_f$  denote latent variables about the states of  $\mathcal{L}$ ,  $\mathcal{Q}$  and  $\mathcal{F}$ , respectively.  $G_l$ ,  $G_q$  and  $G_f$  denote some unknown functions.

$$h_f = G_f(h_q, h_l), h_q = G_q(h_f), h_l = G_l(h_q) \quad (5)$$

#### Algorithm 1: Seed Sample Selection SELECTSEED().

**Input:** candidate dataset  $D_{can} = \{C_i\}_{i=1}^{N_c}$  containing  $N_c$  samples; validation dataset  $D_{val}$ ; number of clusters  $n$ ; number of samples selected from each cluster  $k$   
**Output:** seed samples  $D_{seed}$ ; initial filter model  $M_{fil}$

- 1: //feature extraction and clustering
- 2:  $M_{ext} \leftarrow \text{TRAIN}(D_{can}, D_{val})$
- 3:  $\{[S_{\mathcal{L}}, S_{\mathcal{Q}}, S_{\mathcal{F}}]_i\}_{i=1}^{N_c} \leftarrow \text{EXTRACT}(M_{ext}, D_{can})$
- 4:  $\{CLS_i\}_{i=1}^n \leftarrow \text{CLUSTER}(\{[S_{\mathcal{L}}, S_{\mathcal{Q}}, S_{\mathcal{F}}]_i\}_{i=1}^{N_c})$
- 5: **for**  $CLS_i$  in  $\{CLS_i\}_{i=1}^n$  **do**
- 6: //calculate the cluster center and nearest neighbors
- 7:  $center \leftarrow \text{CENTER}(CLS_i)$
- 8:  $nn \leftarrow \text{NN}(center, k-1)$
- 9:  $seed \leftarrow \text{CONCAT}(center, nn)$
- 10:  $D_{seed} \leftarrow \text{APPEND}(D_{seed}, seed)$
- 11:  $pass \leftarrow \text{False}$
- 12: **while**  $!pass$  **do**
- 13:  $pass \leftarrow \text{VERIFY}(D_{seed})$
- 14: //replace  $N_x$  samples with their nearest neighbors
- 15:  $tmp \leftarrow \text{RANDOM}(seed, N_x)$
- 16:  $\text{REMOVE}(D_{seed}, tmp)$
- 17:  $D_{seed} \leftarrow \text{APPEND}(D_{seed}, \text{NN}(tmp, 1))$
- 18: **end while**
- 19: **end for**
- 20:  $M_{fil} \leftarrow \text{TRAIN}(D_{seed})$
- 21: **return**  $D_{seed}, M_{fil}$

We use the output tensors  $h_l$ ,  $h_q$  and  $h_f$  of RouteNet-F as features. The feature extractor  $M_{ext}$  is used to extract features for all samples  $\{C_i\}_{i=1}^{N_c}$  in the candidate dataset  $D_{can}$ , which can convert a candidate sample  $C_i$  into a feature vector as follows:

$$[S_{\mathcal{L}}, S_{\mathcal{Q}}, S_{\mathcal{F}}]_i \quad (6)$$

where  $S_{\mathcal{L}}$  denotes the state of the physical links of network topology (e.g., utilization),  $S_{\mathcal{Q}}$  describes the state of queues at the output port of nodes (e.g., occupation), and  $S_{\mathcal{F}}$  denotes the state of the active flows in source-destination pairs in the network (e.g., delay, throughput, and loss).

Then, we cluster the feature vectors  $\{[S_{\mathcal{L}}, S_{\mathcal{Q}}, S_{\mathcal{F}}]_i\}_{i=1}^{N_c}$  of the candidate dataset  $D_{can}$ . The clustering algorithm groups samples into  $n$  clusters, where  $n$  is estimated based on prior knowledge. The clusters are denoted as  $\{CLS_i\}_{i=1}^n$ , where samples within the same cluster are more similar to each other than those in different clusters. For each of the  $n$  clusters, we calculate the cluster center and the  $k-1$  nearest neighbors of each cluster center by calculating the euclidean distance (7) between a sample  $x$  and the cluster center  $c$ , where  $m$  denotes the dimension of the feature vector.

$$d(x, c) = \sqrt{\sum_{i=1}^m (x_i - c_i)^2} \quad (7)$$

These  $nk$  samples, including cluster centers and nearest neighbors, are taken as seed samples and added to the seed dataset

**Algorithm 2:** Incremental Optimization INCOPT().

---

**Input:** candidate dataset  $D_{can}$ ; validation dataset  $D_{val}$ ; seed dataset  $D_{seed}$ ; initial filter model  $M_{fil}$ ; maximum number of training samples  $N_{max}$   
**Output:** training dataset  $D_{tra}$

```

1:  $D_{tra} \leftarrow D_{seed}$ 
2: while  $|D_{tra}| < N_{max}$  do
3:   //investigate remaining candidate dataset
4:    $D_{rem} \leftarrow D_{can} - D_{tra}$ ,  $N_r \leftarrow |D_{rem}|$ 
5:    $\{loss\}_{i=1}^{N_r} \leftarrow \text{INFERENCE}(M_{fil}, D_{rem})$ 
6:    $D_{rem} \leftarrow \text{SORT}(D_{rem}, \{loss\}_{i=1}^{N_r})$ 
7:   //filter out easy, hard, and OOD samples
8:    $D_{low} \leftarrow \text{TOP}(D_{rem})$ ,  $D_{high} \leftarrow \text{LAST}(D_{rem})$ 
9:    $D_{ood} \leftarrow \text{OOD}(D_{high})$ 
10:   $D_{hard} \leftarrow D_{high} - D_{ood}$ 
11:   $D_{tra} \leftarrow \text{CONCAT}(D_{low}, D_{hard})$ 
12:   $pass \leftarrow \text{False}$ 
13:  while  $!pass$  do
14:     $pass \leftarrow \text{VERIFY}(D_{tra})$ 
15:    //replace  $N_x$  samples with their nearest
    neighbors
16:     $tmp \leftarrow \text{RANDOM}(D_{tra}, N_x)$ 
17:     $\text{REMOVE}(D_{tra}, tmp)$ 
18:     $D_{tra} \leftarrow \text{APPEND}(D_{tra}, \text{NN}(tmp, 1))$ 
19:  end while
20:   $M_{fil} \leftarrow \text{TRAIN}(D_{tra}, D_{val})$ 
21: end while
22: return  $D_{tra}$ 

```

---

$D_{seed}$ . In order to ensure the validity of the seed dataset  $D_{seed}$ , we verify these samples using expert knowledge. This includes analyzing factors such as ranges of delay, queue utilization, link utilization, and average port occupancy. If any of the samples fail this verification process, we replace them with their nearest neighbors until all of the seed samples pass the verification.

After obtaining the seed dataset  $D_{seed}$  consisting of  $nk$  samples, we train the RouteNet-F model using  $D_{seed}$ . To ensure that the model does not over-fit, we use the validation dataset  $D_{val}$  for validation during training. The trained model will be the initial filter model  $M_{fil}$  in Stage 2-2.

2) *Incremental Optimization (Stage 2-2):* We obtain the seed dataset  $D_{seed}$  and the initial filter model  $M_{fil}$  in Stage 2-1.  $D_{seed}$  serves as the initial training dataset  $D_{tra}$ . In Stage 2-2, we iteratively optimized the filter model  $M_{fil}$  to investigate the candidate dataset  $D_{can}$  until we obtain the high-quality training dataset  $D_{tra}$  that satisfies the requirements. Algorithm 2 presents the pseudo-code of Stage 2-2 INCOPT().

At first, we obtain the remaining candidate dataset  $D_{rem}$  containing  $N_r$  samples by subtracting the training dataset  $D_{tra}$  from the candidate dataset  $D_{can}$ . We use the filter model  $M_{fil}$  to perform inference on the candidate samples in the remaining candidate dataset  $D_{rem}$  and obtain respective loss values  $\{loss\}_{i=1}^{N_r}$  of these samples. We propose an order-based selection strategy. We sort the candidate samples in ascending order based on their loss values and process the top  $N_t$  samples

with relatively low loss values using (8). At iteration  $iter$ , the  $i$ -th sample is selected with probability  $P_i$ , where  $R_{iter}$  represents the selection ratio and  $P_i$  exponentially decays with its order  $i$ . The selected  $low$  samples are added to the dataset  $D_{low}$ .

$$P_{N_t} = \frac{P_1}{R_{iter}}, P_i = \frac{1/\exp(\log(R_{iter})/N_t)^i}{\sum_{i=1}^{N_t} 1/\exp(\log(R_{iter})/N_t)^i} \quad (8)$$

We process the last  $N_l$  candidate samples with relatively high loss values in a similar way. The samples with higher loss values have higher probabilities of being selected. The selected  $high$  samples are added to the dataset  $D_{high}$ .

Then, since lower loss values indicate higher prediction accuracy, the samples in the dataset  $D_{low}$  are deemed valid based on the evaluation of the current filter model  $M_{fil}$ . Subsequently, we examine samples in the dataset  $D_{high}$ , which could be hard or OOD samples. Including hard samples in the training dataset can benefit model training, while OOD samples should be detected and removed. To identify OOD samples, we use two methods - rule-based and influence-based. The rule-based method defines a series of rules based on prior knowledge and removes invalid samples based on these preset rules, such as those with either too low or too high average port occupancy and those without long paths. The influence-based method measures the impact of a sample on the model. According to (9), if the MAPE (or MAE) of the model decreases beyond the threshold  $thr_{iter}$  on iteration  $iter$  after adding the sample, it is considered to be an OOD sample and is added to the dataset  $D_{ood}$ . The remaining samples in the dataset  $D_{high}$  not identified as OOD samples are considered hard samples and are placed in the dataset  $D_{hard}$ .

$$\frac{\text{MAPE}(M_w) - \text{MAPE}(M_{w/o})}{\text{MAPE}(M_{w/o})} \geq thr_{iter} \quad (9)$$

Next, we put the easy samples from dataset  $D_{low}$  and the hard samples from dataset  $D_{hard}$  to the training dataset  $D_{tra}$ . We then examine these samples using expert knowledge, such as analyzing the ranges of delay, queue utilization, link utilization, average port occupancy, and the balance of easy and hard samples. When some samples do not pass the expert knowledge verification, we replace them with their nearest neighbors calculated in Stage 2-1 until all samples pass the verification. Subsequently, we train a new filter model  $M_{fil}$  on the training dataset  $D_{tra}$  (with the validation dataset  $D_{val}$  for validation to prevent over-fitting).

We repeat the above processes iteratively to optimize the filter model  $M_{fil}$  and the training dataset  $D_{tra}$  until  $D_{tra}$  meets the constraints described in Section III-B.

#### IV. CASE STUDY

##### A. AutoOPT in DTN

Fig. 5 illustrates the framework of DTN and the role of AutoOPT in DTN. DTN uses digital twin technology to create a real-time virtual twin of the physical network. The virtual twin maps the actual state of the physical network, enables efficient management and intelligent decision-making, and improves the network's stability, security, and operational efficiency.



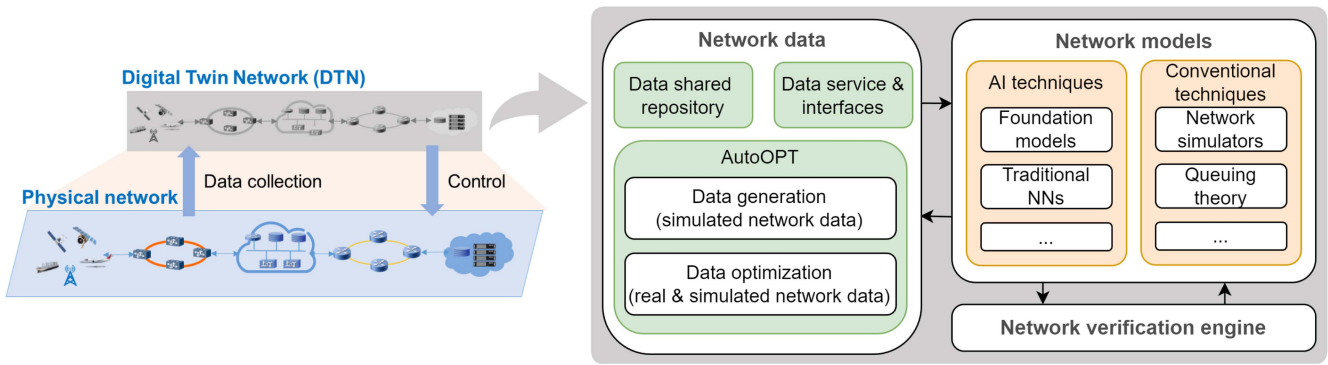


Fig. 5. AutoOPT in DTN. DTN uses digital twin technology to create a real-time virtual twin of the physical network. AutoOPT can be used to generate simulated network data and optimize real and simulated network data in DTN.

In DTN, network mechanisms can be modeled using conventional and AI techniques. Conventional techniques include network simulators, network calculus, queuing theory, etc. AI techniques include foundation models (e.g., GPT-like models for networks), traditional NNs, etc. The network verification engine provides simulation and emulation capabilities, which can be used to verify network models.

Network data is the foundation for building DTN. The quantity and quality of data directly affect the accuracy of DTN modeling. Due to issues such as high cost, privacy, security, and lack of scenario diversity, real data collected from real networks and network management systems often struggles to meet the requirements of DTN modeling. AutoOPT offers a solution with its advanced generation and optimization capabilities. By generating realistic simulated data, improving data quality, diversifying data, and removing noise and outliers, AutoOPT can provide richer, more accurate, and high-quality network data for DTN modeling. The distribution of real network data is constantly changing [38] [39]. In AutoOPT, the network configurations are designed to be diverse to cover as many scenarios as possible, which is beneficial for adapting to real data with various distributions. Additionally, data is continuously generated and optimized. The real data is a part of the candidate data, and the feature extractor and the filter model are trained using the candidate data. If the distribution of the real data changes, the candidate data, the feature extractor, and the filter model will also change to generate beneficial data for the real data.

## B. Use Cases

DTN can be used for a variety of networks, such as data center networks, IP bearer networks, vehicular networks, wireless networks, optical networks, and IoT networks. This section demonstrates the role of AutoOPT in DTN through some typical use cases.

1) *Configuration Evaluation and Optimization in Data Center Networks*: Data centers are essential for powering the exponential growth of Internet services. They contain numerous computing and storage nodes connected by a specially designed data center network (DCN). DCN serves as a communication backbone and plays a critical role in optimizing data center

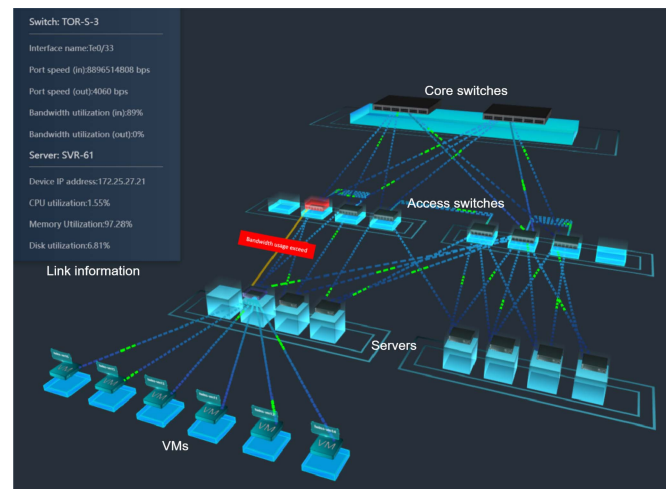


Fig. 6. The practice of using DTN for a small DCN in China Mobile.

operations. However, the unique requirements of the DCN, such as large-scale, diverse applications, high power density, and high reliability, pose significant challenges to its infrastructure and operations.

DTN can be applied to DCN to test network configurations and new technologies, thereby reducing risk and eliminating network failures caused by incorrect configurations. DTN needs to be able to model DCN traffic accurately. One of the challenges is generating realistic network traffic. By understanding traffic patterns, AutoOPT can help generate simulated network data and optimize the collected real data and simulated network data. Various factors (e.g., business type, network size, traffic quantity, and load) influence traffic patterns in large-scale DCN. Additionally, traffic patterns change over time. For example, latency-sensitive online transaction processing workloads tend to appear during the day, while online analytical processing workloads appear at night. Moreover, as DCN's bandwidth increases, the time scale of network events decreases. For example, micro-burst traffic can cause packet dropping, leading to severe performance degradation, such as the Incast problem.

The practice of using DTN for a small DCN in China Mobile is shown in Fig. 6, which includes 2 switches in the core layer, 8 Top of Rack (ToR) access switches, and 8 servers. Without

DTN, fault detection and recovery primarily depend on manual experience, which can impact accuracy and efficiency. DTN can actively prevent inappropriate network configurations and improve operational and maintenance efficiency by over 30%.

2) *Performance Prediction in IP Bearer Networks*: Internet service providers face challenges in delivering high-bandwidth, low-latency, and reliable network services. For instance, in large networks like metropolitan area networks (MANs), network devices such as routers and switches typically reach tens of thousands, and various services are mixed and deployed in the same environment. The IP protocol, which is widely adopted, follows the best-effort principle and does not aim to provide predictable performance. Therefore, ensuring long-term stability and availability of network services is challenging in the event of potential network failures.

DTN can serve as a high-fidelity simulation environment for IP bearer network performance prediction. Obtaining accurate network status information is crucial for fine-tuning network protocols and identifying network faults. In recent years, in-band network telemetry (INT) technology [40] has enabled the development of innovative ultra-large-scale network architectures by carrying network performance data in metadata within data packet headers on the data plane. Based on real network performance data obtained through INT, AutoOPT can generate fine-grained and high-fidelity simulated data and optimize the real and simulated data to be more suitable for model training.

3) *Task Offloading in Vehicular Networks*: The evolution of vehicular networks has led to various delay-sensitive applications, such as autonomous driving and navigation. Vehicles with limited resources face challenges in meeting these requirements with low/ultra-low latency. Vehicles can offload some computationally intensive tasks to other resource-rich computing platforms, such as nearby vehicles, edge servers, and remote cloud servers. However, the dynamic network topology, strict low-delay requirements, and large task data in vehicular networks pose significant challenges for efficient offloading.

DTN is an emerging approach that enables real-time monitoring of the status of vehicular networks and helps make effective offload decisions by connecting the physical and digital worlds. Machine learning algorithms are gaining popularity for task offloading to improve accuracy and efficiency. Vehicular networks are significantly more dynamic and heterogeneous than traditional communication networks, resulting in more serious real data shortages and quality issues. On the one hand, AutoOPT can generate simulated data using scale-independent indicators, which helps adapt to dynamic network topologies. On the other hand, AutoOPT can automatically filter high-quality data from multiple sources, improving the effectiveness of model training.

## V. EVALUATION

### A. Experimental Setup on Simulated Network Data

We evaluate the entire process of AutoOPT with the validation dataset and the test dataset provided by GNNet Challenge 2022 [22], involving data generation (Stage 1) and data optimization (Stage 2). These datasets are generated by simulating various network scenarios via the OMNet++ network simulator.

Experiments are conducted on a server with the following configuration:

- *CPU*: Intel® Xeon® Gold 5118 with 4 cores, 2.30GHz;
- *Memory*: 32 GB;
- *OS*: Ubuntu 18.04.3 LTS (Bionic Beaver);
- *Network simulator*: an accurate packet-level network simulator based on OMNet++ (Docker image<sup>1</sup>).

Details regarding the datasets are as follows:

- *Constraints of training dataset*:  $D_{tra}$  can contain up to  $N_{max} = 100$  samples (network topology, routing policy, and traffic matrix tuple) from small networks with up to  $X_{tra} = 10$  nodes (no isolated nodes). The link capacity of  $D_{tra}$  falls between the range of  $[L_l = 8000, L_u = 64000]$ .
- *Validation dataset*:  $D_{val}$  consists of samples from large networks with up to  $X_{val} = 300$  nodes, which include larger link capacities and longer paths.
- *Test dataset*:  $D_{test}$  consists of samples from large networks with up to  $X_{test} = 300$  nodes, which include larger link capacities and longer paths.  $D_{test}$  has similar distributions with  $D_{val}$  and is not seen during training.

The evaluation metrics used to measure the accuracy of the trained RouteNet-F model are the MAPE (2) / MAE (3) of the per-path mean delay, jitter, and loss on networks.

### B. Results on Simulated Network Data

1) *Data Visualization*: The distributions of the generated training dataset through AutoOPT are visualized in Fig. 7. Fig. 7(a) displays the link load's Cumulative Distribution Function (CDF). The CDF of a random variable is a method to describe the distribution of random variables. The CDF of a random variable  $X$  is defined as  $F_X(x) = P(X \leq x)$ , for all  $x \in \mathbb{R}$ . The CDF in Fig. 7(a) demonstrates that the generated training dataset is evenly distributed across light to heavy link load. Fig. 7(b) presents the Histogram of the delay per path across all samples, which shows that the delay of 89% of paths is less than 1 s, and the delay of 93% of paths is less than 3 seconds. Fig. 7(c) shows the Histogram of losses for all samples, while Fig. 7(d) displays the Histogram of generated bandwidth per path for all samples. Through data visualization, we find that the scale-independent indicators of the generated training dataset have a comparable distribution to the validation dataset, indicating the effectiveness of AutoOPT to some extent.

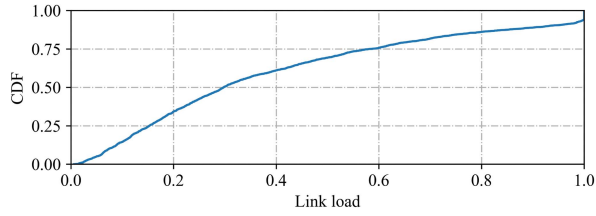
2) *Generalization to Large Networks*: Generalization is an open challenge in AI modeling. We evaluate the generalization of the training dataset generated by AutoOPT. We train a RouteNet-F model on the generated training dataset from small networks (8 to 10 nodes) and test the model on a test dataset from large networks (50 to 300 nodes) that is not seen during training. The experimental results are presented in Table I and Fig. 8, which show the MAPE/MAE of the per-path mean delay, jitter, and loss on networks with different topology sizes (number of nodes) obtained on the validation and test datasets. For the test dataset, the results in Fig. 8 show that the MAPE of delay

<sup>1</sup> docker pull bnnpuc/netsim:v0.1

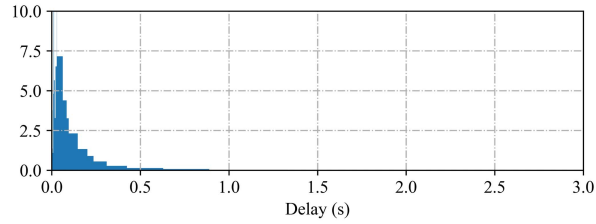


TABLE I  
GENERALIZATION TO LARGE NETWORKS

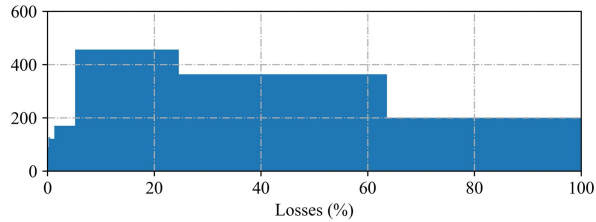
Topology Size	Delay		Jitter		Loss	
	Validation MAPE	Test MAPE	Validation MAPE	Test MAPE	Validation MAE	Test MAE
ALL	6.63%	8.55%	14.59%	15.05%	6.06%	6.83%
50 nodes	5.38%	5.80%	13.64%	14.49%	5.55%	5.29%
75 nodes	5.44%	5.24%	12.98%	13.71%	5.13%	5.18%
100 nodes	5.27%	5.32%	14.04%	14.07%	5.66%	5.20%
130 nodes	6.62%	7.34%	13.29%	14.51%	6.62%	5.91%
170 nodes	6.83%	8.21%	14.41%	14.52%	6.70%	6.16%
200 nodes	6.31%	8.35%	14.16%	14.81%	6.67%	6.73%
240 nodes	7.64%	8.15%	15.18%	15.19%	7.01%	7.24%
260 nodes	7.56%	8.76%	15.36%	15.30%	6.90%	7.39%
280 nodes	7.66%	9.33%	15.13%	15.47%	6.87%	7.33%
300 nodes	7.75%	9.03%	14.60%	15.36%	6.68%	7.24%



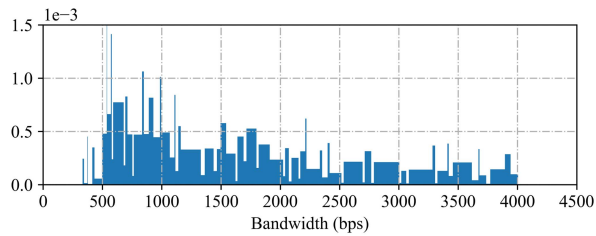
(a) CDF of the link load.



(b) Histogram of the delay per path across all samples.



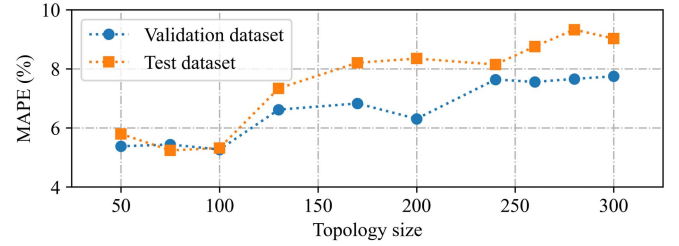
(c) Histogram of losses for all samples.



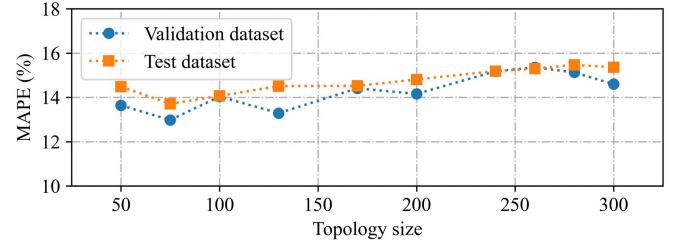
(d) Histogram of generated bandwidth per path for all samples.

Fig. 7. Distribution visualization of the generated training dataset through AutoOPT.

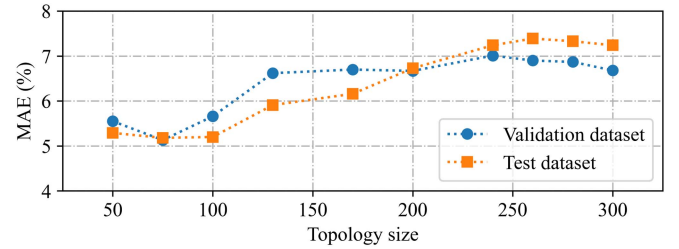
with the topology sizes closer to the training dataset is less than 6%. The MAPE of delay remains stable at about 9% as the topology size increases. Similar results can be observed for MAPE of jitter and MAE of loss as shown in Fig. 8(b) and (c). The experimental results demonstrate that the training dataset generated by AutoOPT has a strong generalization capability.



(a) Delay.



(b) Jitter.



(c) Loss.

Fig. 8. Network performance evaluation (delay, jitter, and loss) - MAPE/MAE with different topology sizes (number of nodes) obtained on the validation and test datasets.

3) *Performance of Data Optimization:* We evaluate the seed sample selection (Stage 2-1) and incremental optimization (Stage 2-2) in the data optimization stage (Stage 2). The performance of Stage 2 is shown in Table II and Fig. 9, which includes the number of samples in the training dataset, MAPE of the per-path mean delay on networks obtained on the validation dataset and test dataset of each iteration. During the first iteration, 20 seed samples are screened out in Stage 2-1, which are used as initial training samples. The model trained on these seed samples can obtain a relatively low MAPE on both the validation and test datasets, indicating the effectiveness of Stage 2-1. Filter models

TABLE II  
PERFORMANCE OF DATA OPTIMIZATION

Iteration	Training Samples	Delay	
		Validation MAPE	Test MAPE
1	20 (seed samples)	10.96%	12.38%
2	38	9.46%	11.34%
3	57	8.43%	10.76%
4	79	7.13%	9.43%
5	100	6.63%	8.55%

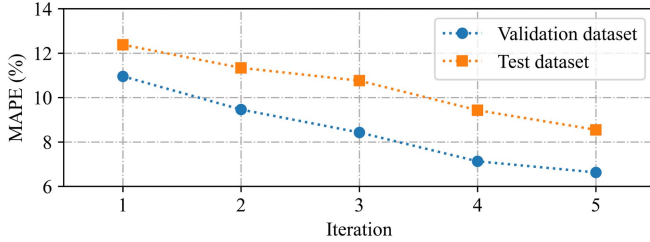


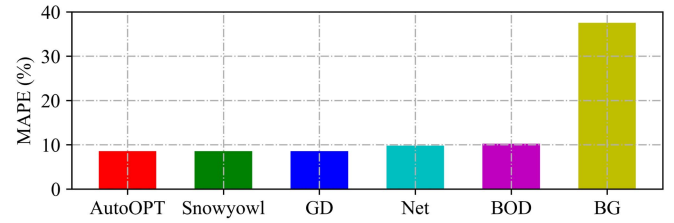
Fig. 9. Network performance evaluation (delay) - MAPE obtained on the validation dataset and test dataset of each iteration.

are RouteNet-F models trained on the training dataset used in each iteration of Stage 2-2. The MAPE obtained in Iterations 2-5 shows that these filter models help optimize the dataset, resulting in 100 high-quality training samples.

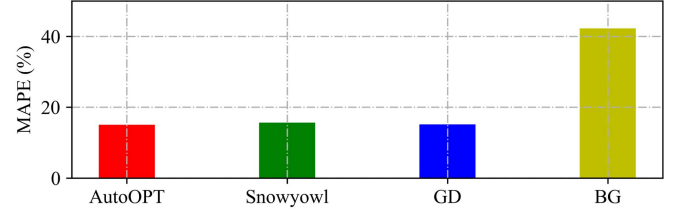
4) *Comparison With Baseline Methods*: We compare AutoOPT with the following baseline methods:

- *Baseline solution provided by GNNet Challenge (BG)* [22]: *quickstart.ipynb* in official GitHub repository;
- *Snowyowl* [41]: a three-step method including initial data generation, data refactoring, and data cleaning;
- *Ghost Ducks (GD)* [42]: an oracle (a well-trained model on a large dataset) based sampling method;
- *Net* [43]: a two-step method including beta distribution data generation and leave-one-out sample ranking;
- *Bayesian Optimization on Detours (BOD)* [44]: BOD formulates data generation as a black-box Bayesian optimization problem.

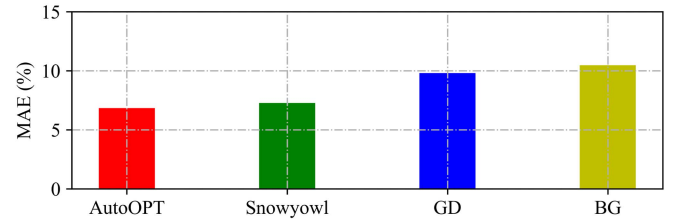
Snowyowl, GD, Net, and BOD are winning solutions of GNNet Challenge 2022 [22]. For each solution (AutoOPT and the baseline methods), a RouteNet-F model in TensorFlow [45] is trained with fixed training parameters on the generated training dataset from networks with small topology sizes (up to 100 samples and up to 10 nodes). The trained model is tested on the test dataset from networks with large topology sizes (50 to 300 nodes) not seen in training. Fig. 10 compares the MAPE/MAE of the per-path mean delay, jitter, and loss on networks. The jitter and loss of Net and BOD are not evaluated because they are not open source. As shown in Fig. 10(a), we observe that the MAPE of delay of AutoOPT, Snowyowl, and Ghost Ducks is similar, at about 8.5%. Additionally, as shown in Fig. 10(b) and (c), the MAPE of jitter (15.05%) and the MAE of loss (6.83%) of AutoOPT are better than other baseline methods. The winning solutions of the challenge rely on either fine-grained analysis of the validation dataset (e.g., Snowyowl) or the massive candidate dataset (e.g., around 270,000 samples in GD). AutoOPT only requires some coarse-grained features of the



(a) Delay.



(b) Jitter.



(c) Loss.

Fig. 10. Comparison of AutoOPT and baseline methods on the test dataset.

TABLE III  
STATISTICAL INFORMATION OF REAL NETWORK DATA

Data	Statistical Information
Number of Nodes	5 ~ 8
Number of Edges	10 ~ 26
Link Capacity	10 ~ 80 Gbps
Traffic Rate	8.58E+05 ~ 3.24E+08 bits/sec
Packet Rate	222 ~ 43856 packets/sec
Packet Size	824 ~ 11552 bits
Scenarios	CBR + ON-OFF / ON-OFF

validation dataset to generate a smaller candidate dataset (around 500 samples in experiments) and can then automatically select high-quality samples from the candidate dataset. In comparison to the winning solutions of the challenge, AutoOPT achieves similar results with a more cost-effective approach.

### C. Experimental Setup on Real Network Data

In some scenarios, a small amount of high-quality training data needs to be selected from the large amount of data collected, which can significantly reduce the time and computing power required for subsequent model training. Therefore, we further evaluate the performance of data optimization (Stage 2) of AutoOPT with the real network dataset provided by GNNet Challenge 2023 [22].

The real network dataset is collected from a network testbed. The network testbed includes 8 Huawei NetEngine 8000 M1A hardware routers, each using 6 ports connected via two hardware switches with 48 ports at 1 Gbps. The T-REX [46] traffic

TABLE IV  
PERFORMANCE OF DATA OPTIMIZATION ON REAL NETWORK DATA

Model	Size( $D_{tra}$ )/ Size( $D_{can}$ )	CBR + ON-OFF			ON-OFF		
		Size( $D_{tra}$ )	Test MAPE	Performance loss	Size( $D_{tra}$ )	Test MAPE	Performance loss
RouteNet-F	10%	47 GB	38.65%	4.13%	45 GB	38.89%	3.61%
	25%	118 GB	36.28%	1.76%	112 GB	36.32%	1.04%
	50%	236 GB	35.25%	0.73%	225 GB	35.63%	0.35%
	100%	471 GB	34.52%	-	449 GB	35.28%	-
m0b1us	10%	47 GB	21.34%	1.67%	45 GB	22.79%	1.36%
	25%	118 GB	20.56%	0.89%	112 GB	22.01%	0.58%
	50%	236 GB	20.06%	0.39%	225 GB	21.66%	0.23%
	100%	471 GB	19.67%	-	449 GB	21.43%	-

generator is connected to the testbed through 10 Gbps ports with the switches, generating 1 or 2 flows per path that can be CBR or ON-OFF. An optical splitter at the fibers connected to the traffic generator creates a copy of the traffic. Mellanox ConnectX-5 cards with hardware timestamping and DPDK are used to process the traffic and compute the delay. The dataset includes 11 topologies ranging from 5 to 8 nodes (10 different routings per topology). The dataset contains two datasets with no intersection: CBR + ON-OFF (471 GB) and ON-OFF (449 GB). The main limitation of the dataset is the richness of scenarios. The statistical information of the dataset is shown in Table III.

For the CBR + ON-OFF and ON-OFF datasets, we split each dataset into two parts: 20% for the test dataset  $D_{test}$  and the remaining 80% for the candidate dataset  $D_{can}$ . We use AutoOPT's Stage 2 to optimize  $D_{can}$  to select a small amount of high-quality data to be the training dataset  $D_{tra}$ . The dataset contains some noisy data and incomplete data. Noisy and incomplete data caused by the data collection are removed or completed during data preprocessing. For noisy data that affects training accuracy, AutoOPT's OOD detection mechanism can identify and remove them.

The evaluation metric used to measure the accuracy of the trained RouteNet-F model is the MAPE (2) of the per-flow mean delay on networks.

#### D. Results on Real Network Data

1) *Adaption to Models and Scenarios*: For the CBR + ON-OFF and ON-OFF datasets, we train the following models on the training dataset  $D_{tra}$  and test them on the test dataset  $D_{test}$ .

- *RouteNet-F* [13]: we use the implementation version [47] which used as the baseline for GNNet Challenge 2023;
- *m0b1us* [48]: a method uses the attention mechanism to enhance GNN generalization, which is the 1st place solution of GNNet Challenge 2023.

Table IV presents Size( $D_{tra}$ )/Size( $D_{can}$ ), Size( $D_{tra}$ ), MAPE of the mean per-flow delay, and performance loss (MAPE differences compared to 100% candidate data) obtained on the test dataset. Table IV shows that the models (RouteNet-F and m0b1us) still perform well on the test dataset using only a small amount of data selected by AutoOPT for training. For example, for the CBR + ON-OFF dataset, 10% of the data yields the MAPE of 38.65% (RouteNet-F) and 21.34% (m0b1us), with the performance loss of only 4.13% (RouteNet-F) and 1.67% (m0b1us); for the ON-OFF dataset, 10% of the data obtained

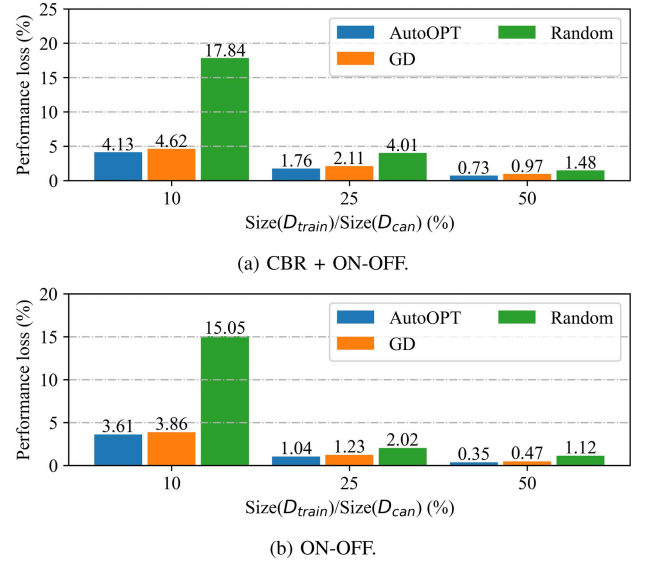


Fig. 11. Comparison of AutoOPT and baseline methods.

the MAPE of 38.89% (RouteNet-F) and 22.79% (m0b1us), with the performance loss of only 3.61% (RouteNet-F) and 1.36% (m0b1us). The experimental results indicate that AutoOPT can adapt to various models and scenarios, effectively selecting high-quality data for training and significantly reducing the time and computing power needed for model training.

2) *Comparison With Baseline Methods*: We compare AutoOPT with random selection and GD [42]. GD is an oracle (a well-trained model on a large dataset) based sampling method, one of the winning solutions of GNNet Challenge 2022. For the CBR + ON-OFF and ON-OFF datasets, we use each solution (AutoOPT and the baseline methods) to optimize  $D_{can}$  to select a small amount of high-quality data to be the training dataset  $D_{tra}$ . We train the RouteNet-F model on the training dataset  $D_{tra}$  generated by each solution and test them on the test dataset  $D_{test}$ . As shown in Fig. 11, AutoOPT can significantly reduce performance loss compared to random selection and performs better than GD. GD is one of the baseline methods for simulated network data experiments. Compared to delay prediction on simulated network data, AutoOPT has more apparent advantages over GD in real network data. The results indicate that AutoOPT has more robust adaptability to real network data with noisy and incomplete data.



## VI. CONCLUSION AND FUTURE WORK

DTN offers a solution for managing complex networks with emerging applications such as cloud gaming, AR/VR, and vehicular networks. AI models help build real-time, lightweight, and highly accurate DTNs. To train high-quality DTN AI models, various data sources such as real networks, network simulators, and generative AI models can be utilized. In this paper, an automatic data generation and optimization method for DTN called AutoOPT has been proposed. AutoOPT generates simulated network data based on scale-independent indicators and selects high-quality data from multi-source candidate data through seed sample selection and incremental optimization. With this approach, AutoOPT can improve data quality in a reliable, efficient, and systematic way, thus improving the accuracy and generalization of the model. We introduce typical use cases in data center networks, IP bearer networks, and vehicular networks to show the role of AutoOPT in DTN. We conduct experiments to apply AutoOPT to the DTN performance modeling scenario. AutoOPT's entire process is evaluated by training the model on datasets generated by AutoOPT and testing on simulated datasets provided by GNNet Challenge 2022. The experimental results show that AutoOPT is more cost-efficient than the winning solutions of the challenge while achieving similar results. We also evaluate AutoOPT's data optimization performance on real network datasets that require quality improvement and demonstrate that AutoOPT can automatically select high-quality data.

Several topics related to generating and optimizing data for DTN still need to be studied. Our future research will focus on the following aspects: 1) selecting data generators based on factors such as accuracy, speed, and fidelity; 2) utilizing data augmentation techniques in Stage 1 to reduce costs further; 3) studying the explainability of the data generation and optimization methods; 4) studying how to provide feedback for data collection to form a closed loop.

## REFERENCES

- [1] M. Li, C. Zhou, L. Lu, Y. Zhang, and T. Sun, "AutoOPT: Data generation and optimization for digital twin network (DTN)," in *Proc. IEEE 16th Int. Conf. Cloud Comput.*, 2023, pp. 376–382.
- [2] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the digital twin: A systematic literature review," *CIRP J. Manuf. Sci. Technol.*, vol. 29, pp. 36–52, 2020.
- [3] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji, "Digital twin modeling," *J. Manuf. Syst.*, vol. 64, pp. 372–389, 2022.
- [4] T. Sun et al., "Digital twin network (DTN): Concepts, architecture, and key technologies," *Acta Automatica Sinica*, vol. 47, no. 3, pp. 569–582, 2021.
- [5] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13789–13804, Sep. 2021.
- [6] L. Bariah, H. Sari, and M. Debbah, "Digital twin-empowered communications: A new frontier of wireless networks," *IEEE Commun. Mag.*, vol. 61, no. 12, pp. 24–36, Dec. 2023.
- [7] Z. Tao, W. Xu, Y. Huang, X. Wang, and X. You, "Wireless network digital twin for 6G: Generative AI as a key enabler," *IEEE Wireless Commun.*, vol. 31, no. 4, pp. 24–31, Aug. 2024.
- [8] D. Wang et al., "Digital twin of optical networks: A review of recent advances and future trends," *J. Lightw. Technol.*, vol. 42, no. 12, pp. 4233–4259, 2024.
- [9] H. Hong et al., "NetGraph: An intelligent operated digital twin platform for data center networks," in *Proc. ACM SIGCOMM Workshop Netw.-Appl. Integration*, 2021, pp. 26–32.
- [10] L. Tang, Q. Hou, W. Wen, D. Fang, and Q. Chen, "Digital twin assisted VNF migration through resource prediction in SDN/NVF-enabled IoT networks," *IEEE Internet Things J.*, vol. 11, no. 21, pp. 35445–35464, Nov. 2024.
- [11] L. Zhao, G. Han, Z. Li, and L. Shu, "Intelligent digital twin-based software-defined vehicular networks," *IEEE Netw.*, vol. 34, no. 5, pp. 178–184, Sep./Oct. 2020.
- [12] L. Zhao et al., "A digital twin-assisted intelligent partial offloading approach for vehicular edge computing," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3386–3400, Nov. 2023.
- [13] M. Ferriol-Galmés et al., "RouteNet-fermi: Network modeling with graph neural networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 3080–3095, Dec. 2023.
- [14] C. Li et al., "m3: Accurate flow-level performance estimation using machine learning," in *Proc. ACM SIGCOMM Conf.*, 2024, pp. 813–827.
- [15] Y. Yoo, G. Yang, C. Shin, J. Lee, and C. Yoo, "Machine learning-based prediction models for control traffic in SDN systems," *IEEE Trans. Serv. Comput.*, vol. 16, no. 6, pp. 4389–4403, Nov./Dec. 2023.
- [16] H. Zhang, X. Ma, X. Liu, L. Li, and K. Sun, "GNN-based power allocation and user association in digital twin network for the terahertz band," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3111–3121, Oct. 2023.
- [17] R. Wang, V. Friderikos, and A. H. Aghvami, "Energy-aware design policy for network slicing using deep reinforcement learning," *IEEE Trans. Serv. Comput.*, vol. 17, no. 5, pp. 2378–2391, Sep./Oct. 2024.
- [18] P. Yu et al., "Digital twin driven service self-healing with graph neural networks in 6G edge networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3607–3623, Nov. 2023.
- [19] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*. Berlin, Germany: Springer, 2010, pp. 15–34.
- [20] A. Varga, "Omnet," in *Modeling and Tools for Network Simulation*. Berlin, Germany: Springer, 2010, pp. 35–59.
- [21] S. E. Whang, Y. Roh, H. Song, and J.-G. Lee, "Data collection and quality challenges in deep learning: A data-centric AI perspective," *The VLDB J.*, vol. 32, pp. 791–813, 2023.
- [22] J. Suárez-Varela et al., "The graph neural networking challenge: A worldwide competition for education in AI/ML for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 51, no. 3, pp. 9–16, 2021.
- [23] L. Hui, M. Wang, L. Zhang, L. Lu, and Y. Cui, "Digital twin for networking: A data-driven performance modeling perspective," *IEEE Netw.*, vol. 37, no. 3, pp. 202–209, 2022.
- [24] M. Wang et al., "Neural network meets DCN: Traffic-driven topology adaptation with deep learning," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 2, no. 2, pp. 1–25, 2018.
- [25] T. Suzuki, Y. Yasuda, R. Nakamura, and H. Ohsaki, "On estimating communication delays using graph convolutional networks with semi-supervised learning," in *Proc. Int. Conf. Inf. Netw.*, 2020, pp. 481–486.
- [26] Q. Zhang, K. K. Ng, C. Kazer, S. Yan, J. Sedoc, and V. Liu, "Mimicnet: Fast performance estimates for data center networks with machine learning," in *Proc. ACM SIGCOMM Conf.*, 2021, pp. 287–304.
- [27] M. Wang, L. Hui, Y. Cui, R. Liang, and Z. Liu, "xNet: Improving expressiveness and granularity for network modeling with graph neural networks," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 2028–2037.
- [28] C. Zheng, G. Wu, and C. Li, "Toward understanding generative data augmentation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 54046–54060.
- [29] N. Schneider, S. Goshtasbpour, and F. Perez-Cruz, "Anchor data augmentation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 74890–74902.
- [30] J. Gui et al., "A survey on self-supervised learning: Algorithms, applications, and future trends," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 9052–9071, Dec. 2024.
- [31] Y. You, T. Chen, Z. Wang, and Y. Shen, "When does self-supervision help graph convolutional networks?," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 10871–10880.
- [32] J. Yoon, S. Arik, and T. Pfister, "Data valuation using reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 10842–10851.
- [33] S. M. Xie, S. Santurkar, T. Ma, and P. Liang, "Data selection for language models via importance resampling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 34201–34227.
- [34] T. B. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.
- [35] C. R. Palmer and J. G. Steffan, "Generating network topologies that obey power laws," in *Proc. IEEE Glob. Telecommun. Conf.*, 2000, pp. 434–438.
- [36] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

- [37] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*. New York, NY, USA: ACM, 2022, pp. 287–290.
- [38] C. Mao, L. Zhao, G. Min, A. Hawbani, A. Y. Al-Dubai, and A. Y. Zomaya, "Informative causality-based vehicle trajectory prediction architecture for domain generalization," in *Proc. IEEE Glob. Commun. Conf.*, 2023, pp. 1–6.
- [39] X. Zhou et al., "Digital twin enhanced federated reinforcement learning with lightweight knowledge distillation in mobile networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3191–3211, Oct. 2023.
- [40] L. Tan et al., "In-band network telemetry: A survey," *Comput. Netw.*, vol. 186, 2021, Art. no. 107763.
- [41] J. M. Ziazet, C. Boudreau, O. Delgado, and B. Jaumard, "Designing graph neural networks training data with limited samples and small network sizes," *ITU J. Future Evolving Technol.*, vol. 4, no. 3, pp. 492–502, 2023.
- [42] E. Sason, Y. Lubarsky, A. Gaissinski, E. Kravchik, and P. Kisilev, "Oracle-based data generation for highly efficient digital twin network training," *ITU J. Future Evolving Technol.*, vol. 4, no. 3, pp. 472–484, 2023.
- [43] M. Helm, B. Jaeger, and G. Carle, "Data-efficient GNN models of communication networks using beta-distribution-based sample ranking," *ITU J. Future Evolving Technol.*, vol. 4, no. 3, pp. 485–491, 2023.
- [44] Q. Xiang, Y. Lee, and Y. Lin, and Z. Shao, "Bayesian optimization on detours," 2022. Accessed: Oct. 10, 2024. [Online]. Available: <https://github.com/ITU-AI-ML-in-5G-Challenge/iDMG-Improving-Network-Digital-Twins-through-Data-centric-AI>
- [45] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Des. Implementation*, Savannah, GA, USA, 2016, pp. 265–283.
- [46] O. A. Adeleke, N. Bastin, and D. Gurkan, "Network traffic generation: A survey and methodology," *ACM Comput. Surv.*, vol. 55, no. 2, pp. 1–23, 2022.
- [47] B. N. N. Center, "Routenet baseline for the graph neural networking challenge," 2023. Accessed: Oct. 10, 2024. [Online]. Available: [https://github.com/BNN-UPC/GNNNetworkingChallenge/tree/2023\\_RealNetworkDT](https://github.com/BNN-UPC/GNNNetworkingChallenge/tree/2023_RealNetworkDT)
- [48] C. Modesto, R. Aben-Athar, A. Silva, and S. Lins, "Enhanced GNN generalization to real network dataset by attention mechanism," 2023. Accessed: Oct. 10, 2024. [Online]. Available: <https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-007-GNN-m0b1us>



**Mei Li** received the BS degree from the Beijing University of Posts and Telecommunications, Beijing, China, and the PhD degree from the Institute of Software, Chinese Academy of Sciences. She is a researcher with China Mobile Research Institute. Her research interests include digital twin network and AI-enabled network.



**Cheng Zhou** received the master's degree in signal and information processing from Beijing Jiaotong University, Beijing, China. He is a project manager with China Mobile Research Institute. His research interests include IP network technology and AI-enabled network.



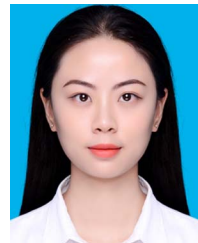
**Lu Lu** (Member, IEEE) received the master's degree from the Beijing University of Posts and Telecommunications, Beijing, China. She is a deputy director with the Department of Network and IT Technology, China Mobile Research Institute; leader of the Core Network Group of CCSA TC5; vice chairman of ITU-T SG13. Her research interests include mobile core network, future network architecture, and edge computing.



**Yan Zhang** (Fellow, IEEE) received the BS degree from Beihang University, Beijing, China, the MS degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, and the PhD degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He is currently a full professor with the Department of Informatics, University of Oslo, Oslo, Norway. His research interests include next-generation wireless networks leading to 6G, and green and secure cyber-physical systems, such as smart grid and transport. He is a fellow of IET, an elected member of the Academia Europaea (MAE), the Royal Norwegian Society of Sciences and Letters (DKNVS), and the Norwegian Academy of Technological Sciences (NTVA).



**Tao Sun** (Member, IEEE) received the BS and PhD degrees in control science and engineering from Tsinghua University, Beijing, China. He is a chief expert of China Mobile Research Institute. His research interests include new mobile network architecture, AI-enabled network, etc.



**Danyang Chen** received the master's degree in communication and information system from Xidian University, in 2020. She is a researcher of China Mobile Research Institute. Her research interests include digital twin network, intent-based network, and other future network technologies.



**Hongwei Yang** received the master's degree in power electronics and power transmission from the Wuhan University of Technology. He is a project manager with China Mobile Research Institute. His research interests include network intelligence and network performance measurement.



**Zhiqiang Li** received the bachelor's degree in electronic science and technology from the Harbin University of Science and Technology. He is a technical manager with China Mobile Research Institute. His research interests include IP network technology and AI-enabled network.