A Graph Neural Network-based Digital Twin for Network Slicing Management

Haozhe Wang, Yulei Wu, *Senior Member, IEEE,* Geyong Min, *Member, IEEE,* and Wang Miao

***Abstract*—Network slicing has emerged as a promising networking paradigm to provide resources tailored for Industry 4.0 and diverse services in 5G networks. However, the increased network complexity poses a huge challenge in network management due to virtualised infrastructure and stringent Quality-of-Service (QoS) requirements. Digital twin (DT) technology paves a way for achieving cost-efficient and performance-optimal management, through creating a virtual representation of slicing-enabled networks digitally to simulate its behaviours and predict the time-varying performance. In this paper, a scalable DT of network slicing is developed, aiming to capture the intertwined relationships among slices and monitor the end-to-end (E2E) metrics of slices under diverse network environments. The proposed DT exploits the novel Graph Neural Network model that can learn insights directly from slicing-enabled networks represented by non-Euclidean graph structures. Experimental results show that the DT can accurately mirror the network behaviour and predict E2E latency under various topologies and unseen environments.**

***Index Terms*—Digital twins, network slicing, Graph Neural Networks, end-to-end modelling.**

I. Introduction

With the emergence of Industry 4.0, Industrial Internet-of-Things (IIoT) and smart city, the network infrastructure is envisioned to meet manifold technical demands and diverse Quality-of-Service (QoS) requirements with respect to latency, throughput, capacity and reliability [1]. To efficiently accommodate a wide range of services and ensure that the network can be flexibly tailored for distinct applications, the concept of network slicing has been proposed as a promising approach to fulfil the prominent paradigm shift from one-size-fits-all solution to softwarised and virtualised design [2]. Network slicing offers an effective means to meet the diverse use case requirements, and is a critical enabler for realising industrial applications with stringent latency requirements on a shared infrastructure [3].

Network slices are logically isolated End-to-End (E2E) virtualised networks operating on a shared physical infrastructure and can be controlled and managed independently to support flexible and efficient service provisioning [2]. Such flexibility and efficiency are realised through the combination of two emerging technologies, i.e., Software Defined Networking (SDN) and Network Function Virtualisation (NFV) [4], which decompose traditional monolithic and proprietary network appliances into multiple small and software-based modular network capabilities called Virtual Network Functions (VNFs) running on standard servers.

However, employing network slicing in 5G networks engenders huge challenges in network management, due to softwarised and virtualised infrastructure and significant differentiation in QoS requirements [5]. Especially, it is intractable for ensuring E2E performance to meet the variety of requirements, since slices share the same infrastructure and cross multiple domains [6]. To leverage network slicing properly, it is crucial to efficiently monitor the network and accurately generate E2E metrics. Such measurements can be effectively used to accomplish autonomous management and dynamic network orchestration to guarantee the associated QoS requirements.

The fast development of digital twin (DT) technology in the industry paves a way to cyber-physical integration, through creating virtual replicas of physical objects in a digital way to simulate their behaviours [7]. The management of network slicing can greatly benefit from a network DT. Firstly, a network DT creates a virtual representation of the physical slicing network and can be used to conduct various what-if scenarios and resource allocation approaches without affecting the physical network. Secondly, through interactions with the physical network, a network DT can generate and process its own data and predict the QoS performance after any configuration changes. A DT of network slicing is important to achieve cost-efficient and performance-optimal slicing management and keep monitoring the performance under a diverse set of operating conditions without impacting the physical network. However, different from a DT in the industrial domain, a network DT needs to merge both the physical and virtual components in slicing-enabled networks. Such virtual elements add much more complexity for developing the DT, since they change frequently and can be created or destroyed in real-time.

The recent advances in Artificial Intelligence (AI) provides a promising means to fulfil the demands of network DT development. The emergence of data-driven Machine Learning (ML) techniques especially Deep Learning [8] has gained popularity in networking areas and led to a new breed of models that learn from data, instead of being explicitly programmed. Researchers are using Deep Neural Networks (DNN) to model complex network behaviours [9] and develop decision-making strategies based on Deep Reinforcement Learning [10]. Most of these works are based on acclaimed learning architectures such as Convolutional Neural Networks (CNNs) [11], Recurrent Neural Networks (RNNs) [12] and AutoEncoders [13] and their variants. However, communication networks are

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under Grant No. EP/R030863/1. *Corresponding author: Y. Wu.*

H. Wang, Y. Wu, G. Min and W. Miao are with the Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, UK. (e-mail:{h.wang3, y.l.wu, g.min, wang.miao}@exeter.ac.uk).

fundamentally represented in the form of graphs, and most of the existing learning architectures are not designed to learn such information structured in non-Euclidean domain, due to irregular topology of graphs and interdependency between nodes. As a result, these models are limited in providing accurate results on graph data and hard to achieve generalisation on dynamic topologies and configurations. This challenge hinders the creation of a network DT and its wide applications in slicing-enabled networks.

To solve the above challenge, we design a network DT based on Graph Neural Networks (GNN) to discover the complicated relationships and interdependency among network slicing, resource utilisation and physical infrastructure, and to generate the E2E metrics prediction of each slice under diverse scenarios. GNNs facilitate the learning of rich relation information among elements in a graph structure and capture the dependency of neighbourhood. This is achieved by generating the virtual representation of the shared nodes and links of slicing infrastructure based on the dynamic resource utilisation as state features, and collectively aggregate these states within the source-destination path of a slice to form E2E slice metrics. Moreover, the proposed network DT exploits an inductive graph framework to gain the generalised capability that can produce E2E metrics under different topologies and network configurations.

The network DT model proposed in the paper achieves a vital step towards realising the ambitious vision of autonomous management of network slicing by providing the ability to map the network status to E2E QoS performance for complex slicing-enabled networks. Such an ability ensures that the QoS requirements are constantly met after any changes performed on slices, and the allocation of resources between slices will not exceed the resource capacity of servers, e.g., base stations or edge servers. To the best of our knowledge, this is the first work to build a DT for network slicing management. The main contributions of this paper are threefold:

(i) We model the intricate interactions between multiple slices deployed on a shared physical infrastructure into a graph structure and design a novel GNN-based virtual representation model. Instead of processing features of each node, the model captures the hidden interdependencies among nodes by examining and aggregating the information from multi-hop neighbour nodes.

(ii) We propose a DT of network slicing to investigate the composite traffic generated by slices and produce accurate predictions of E2E slice latency. The DT adopts the inductive graph learning framework to achieve generalisation and perform prediction under various environments.

(iii) We demonstrate the high accuracy of the DT for estimating E2E latency of slices under arbitrary topologies, diverse slice deployment and traffic distributions that have not been seen during the training process. Hypothetical what-if scenarios are performed using the DT to evaluate the E2E performance with QoS constraints.

The remainder of this paper is organised as follows. In Section II, the related work is presented. Section III is devoted to the design of the system model. Section IV proposes the GNN-based DT for 5G network slicing. The accuracy of the proposed DT is evaluated in Section V. Finally, Section VI concludes the paper.

II. Related Work

Network slicing has been investigated in the context of industrial communication as a means to manage the increasing complexity of manufacturing networks [1], [3]. The existing works focus on the architecture and framework design rather than slicing of the physical networks due to the lack of virtualisation support in specialised infrastructure. Outside the industrial domain, E2E network slicing has become one of the key enabling forces for fulfilling the full capabilities of 5G and has been recognised as a high-priority technical gap to tackle for application quality.

To bridge the gap, considerable efforts have been dedicated towards the problem of slicing resource allocation. Eichhorn *et al.* [14] proposed an SDN-based slicing solution for 5G core networks, but the problem of RAN slicing among multiple applications was not addressed. Taleb *et al.* [15] presented a multi-domain network slicing architecture, including RAN, edge and core, to achieve E2E resource orchestration. However, the problem of meeting the stringent latency requirements of industrial applications was not fully addressed. Therefore, there is a clear gap on the research of the E2E performance of network slicing across multiple domains. Fulfilling such challenges demands the methods that can monitor and predict E2E slicing metrics to ensure the QoS.

Network modelling that has been used to conduct the evaluation and prediction of network performance is a promising candidate to establish a DT to observe whether current network configurations are able to meet the QoS requirements. The networking community has developed numerous analytical models for decades, in particular based on queuing theory [16], [17] and network calculus [18], [19]. However, to ensure model tractability in complex networks, such models typically rely on simplifying assumptions about the properties of underlying architecture (e.g., traffic with Poisson distribution and static routing), which sacrifice their accuracy. For the DT to deliver its promise, it must have sufficient fidelity so as to accurately reflect the network dynamics that can cause networks to behave unpredictably. The network dynamics are typically created by the interplay among the slices, system configurations, network topology, physical infrastructure, and application traffic. Unfortunately, reflecting the intertwined relationships of the preceding components lies beyond the reach of analytical models.

Machine learning-based approaches have attracted many efforts for network optimisation. Kasgari *et al.* [20] proposed a deep reinforcement learning (DRL)-based resource allocation approach to allocate wireless resources to end-users under the URLLC scenario and leveraged generative adversarial networks (GANs) to generate training data for extreme network circumstance. Although DRL shows promising potential in network management, it still suffers from two challenges. It needs large amounts of samples to learn a comprehensive

experience, and the agent needs to keep interacting with a simulated or real environment, which may slow down the training process. Such challenges can be addressed by integrating DRL with a DT model. A DT model can generate unlimited training samples through assigning various network configurations and can act as an environment to perform model-based DRL to largely reduce the training time.

Extending Deep Learning (DL) [8] operating on data represented in a graph structure has been investigated under the umbrella of GNN [21], and has attracted a significant amount of research efforts in recent years [22]. Recently, a Graph Convolutional Network (GCN) architecture for learning and reasoning over graphs has been proposed in [23]. It performs semi-supervised learning in a transductive setting and requires the information of a whole graph during training. Therefore, GCN has scalability issue and works on fixed graphs, which is not suitable for network slicing with stringent QoS requirements and dynamic topologies.

Achieving continuous slicing performance monitoring under various network configurations and different topologies presents additional challenges, as we have to deal with the intertwined impacts between slices with different resource demands and various QoS requirements. Therefore, it is critical to leverage novel DT technology to achieve efficient and accurate performance monitoring for E2E slices.

III. System Model

In this section, we present the system model of the network slicing DT, including the models of 5G substrate network and network slicing system, and then we propose the E2E network slicing graph representation. Table I summarises the notations used in this paper.

TABLE I
Frequently used notations

Parameter	Meaning
G_p	Physical 5G substrate network
V_p	The node set in the substrate network
L_p	The link set in the substrate network
n_i	A physical node providing multiple resources
$R_r(n_i)$	Capacity of r resource on node n_i
$l_p(i,j)$	A physical link connecting n_i and n_j
$B_p(i,j)$	Network bandwidth of $l_p(i,j)$
s_u	A network slice deployed on the physical network G_p
V_u	The VNF nodes in slice s_u
L_u	The virtual links in slice s_u
$nf_{u,i}$	A VNF of s_u on substrate node n_i
$nf_{u,i}^{Res}$	Resource vector of VNF $nf_{u,i}$
$l_u(i,j)$	A virtual link between two VNFs $nf_{u,i}$ and $nf_{u,j}$
f_i	Superposed resource utilisation on physical node n_i
$f_{l_{i,j}}$	Superposed resource utilisation on physical link $l_p(i,j)$
d_s	Delay of a link between two nodes
h_i	Node state representation
$h_s(u)$	State representation of s_u
d_{s_u}	Current E2E latency of slice s_u
$\Phi(n_i)$	The set of neighbour nodes of n_i
$\mathcal{N}(n_i)$	Sampled neighbour nodes of n_i
K	The neighbour sample depth
E_k	The number of neighbours sampled at layer k

A. Network slicing model

Network slices are deployed on a general 5G substrate network that is equipped with various types of resources. In our model, the substrate network is represented as an undirected graph, denoted as $G_p = (V_p, L_p)$, where V_p is the set of physical nodes that consist of multiple commodity servers and L_p indicates the set of links connecting the nodes. Each physical node of the network $n_i \in V_p$ supplies r different types of resources (e.g., CPU, memory, storage, GPU). The resources provided by a node n_i are expressed as a vector $n_i^{Res} = [R_1(n_i), R_2(n_i), \ldots, R_r(n_i)]$, and the various types of resources will be illustrated in numbers in order to have a uniform input. In our model, we consider that the number of resource types on each node is identical, and thus the length of vector n_i^{Res} is the same for $\forall n_i \in V_p$. If devices from different network domains have various types of resources, each type of resources is set at a specific position in the resource vector. For example, a node in a specific network domain does not have certain types of resources, and those positions can be set to 0. Each link in the substrate network $l_p(i,j) \in L_p$ denotes the physical connection between n_i and n_j with an amount of bandwidth. The total available bandwidth for link $l_p(i,j)$ is denoted by $B_p(i,j)$.

The network slices support diverse services, such as multimedia streaming, vehicular communication and Internet-of-Things (IoT), which have different requirements on latency. To achieve the diversity, the 5G networks adopt the SDN and NFV network techniques, in which the nodes and links support virtualisation and can be divided into multiple isolated virtual containers and virtual links to accommodate VNFs. In this way, each network slice consists of multiple associated VNFs deployed on different physical nodes, and these VNFs are chained together through multiple virtual links. Thus, a network slice s_u can be represented as a directed graph denoted by $s_u = (V_u, L_u)$, where V_u and L_u represent the sets of VNFs and virtual links respectively. Each VNF is characterised by a three-tuple, namely the slice to which it belongs, the physical hosting node and the demands of each type of resources. From a modelling point of view, a VNF of slice s_u deployed on node n_i can be indicated by $nf_{u,i} = (s_u, n_i, nf_{u,i}^{Res}), \forall nf_{u,i} \in V_u$, where $nf_{u,i}^{Res} = [R_1(nf_{u,i}), \ldots, R_r(nf_{u,i})]$, $R_r(nf_{u,i}) \leq R_r(n_i)$. Similarly, a virtual link can be characterised by the slice it belongs to, the two VNFs it connects with, and the bandwidth requirement, which can be expressed as $l_u(nf_{u,i}, nf_{u,j}) = (s_u, L'(i,j), B_u(i,j))$, where $L'(i,j)$ is a set of physical links between n_i and n_j that depends on the routing scheme of G_p. For the sake of simplicity, we substitute $l_u(nf_{u,i}, nf_{u,j})$ with $l_u(i,j)$. Therefore, a network slice s_u in a shared infrastructure G_p can be depicted as

$$s_u = \left(\{nf_{u,i}\}_{n_i \in V_p}, \{l_u(i,j)\}_{n_i \neq n_j \in V_p}\right) \tag{1}$$

An E2E slicing model is crucial for developing a DT to

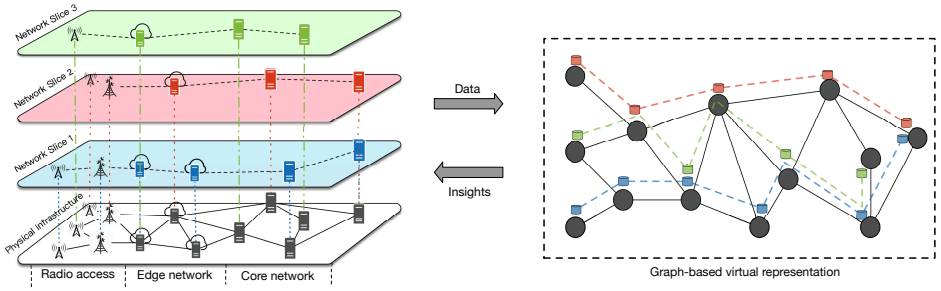


Fig. 1. Network shared by multiple network slices and associated graph-based virtual representation.

capture the whole picture of a network and to evaluate the E2E QoS metrics of slices. In this paper, we consider the E2E latency as the main metric, since latency is one of the most important factors of industrial applications and always has a strict requirement. Delivering an E2E network slice on the substrate network G_p involves managing the resources of nodes allocated for each VNF and selecting a set of proper links connecting them to meet the requirements and ensure optimal utilisation of resources.

B. Graph-based virtual representation

We formalise the problem of producing E2E latencies of multiple slices that share the same infrastructure, as shown in Fig. 1. The left part of the figure shows a network that contains three slices sharing the same substrate network, including a radio access network (RAN), an edge network and a core network. Each slice is dedicated with respect to a service that requires specific functionalities in a particular order. Each network function requires resources from different domains of the substrate network.

The right part of the figure presents the graph-based representation corresponding to the left network slicing infrastructure. Each physical node in any domain may host multiple VNFs that belong to different slices. Physical links are shared by several virtual links, and a virtual link may also be deployed through many physical links. For example, the slices in blue and green colours all have virtual links based on two or more physical links. Therefore, an E2E slicing latency depends on the traffic flows that traverse an ordered set of VNFs, which is further determined by the state of every link connecting these VNFs.

A graph-based virtual representation is the foundation of a GNN model. The sliced network with different types of resources as illustrated in the right part of Fig. 1 is often represented as a high-dimensional array, neglecting the interaction between slices and the relationship between resources and network topology. To address this challenge, we propose a graph-based virtual representation which retains those complex relations. To establish the graph-based virtual representation, we denote the state of a node in the graph as h_i which embodies the information of the slicing traffic traversing this node and the utilisation of links connected by the node. The state of a slice h_s is subject to the states of all the nodes within it, which contains the key information to obtain the E2E latency. Next, we develop a graph-based DT to produce accurate E2E slicing latency and have enough flexibility to adapt to the dynamic slice deployment, including the change of number of slices, different physical topologies and variations of slice utilisation. Furthermore, since the traffic has to travel through predefined ordered VNFs in a network slice, the packet losses and delay on any link would affect the overall latency. We will also propose a nonlinear method to aggregate the states of nodes of a slice.

IV. GNN-based Digital Twin of E2E Network Slicing

In this section, we provide the detailed mathematical description of the proposed DT of network slicing, which performs on the graph representing the physical network and slices.

A. GNN-based digital twin model

To establish the DT, we need to model the interdependent VNFs with predefined orders and different resource requirements, and the QoS requirements for isolated slices. In this regard, we propose a topology-free and generalised graph model as the core component for predicting the E2E latency of slices based on GNNs. The model is motivated by the fact that in network slicing, the physical infrastructure is shared by multiple slices, and the flow traffic in the network is the composition of these slices. In addition, the flow traverses multiple VNFs in each slice sequentially. This implies that the latency of traffic running on one slice is highly related to the traffic behaviours of the other slices that share the same physical nodes and links. Such a dependency of slices makes the GNN model as an appealing approach for solving the intertwined relationship between slices, since most of the existing learning frameworks are not designed to learn from data in non-Euclidean structure and are limited in providing accurate results on graph data and hard to achieve generalisation on dynamic topologies and configurations. In our proposed DT, instead of treating each slice sharing the same infrastructure individually as vectorised number of resources, we propose a GNN-based model that turns the whole substrate network including the deployed network slices into a directed graph leveraging the graph-based virtual representation developed in Section III-B. Then, we adopt the GraphSAGE framework that has been used in chemistry and biology [24], to develop an efficient and scalable GNN-based model to enable latency inferring under dynamic network slicing scenarios.

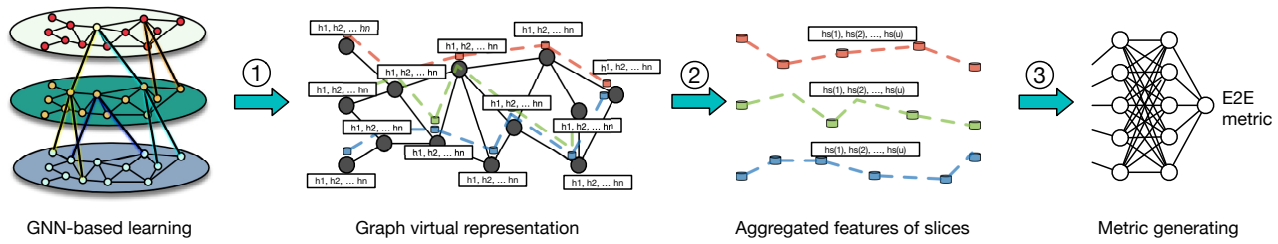


Fig. 2. Working mechanism of the proposed DT.

To build the DT representing the multi-network slicing scenario as a graph, we combine all individual graphs of slice $s_u = (V_u, L_u)$ into a synthetic network graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$. The topology of the synthetic graph is the same as the physical graph, in which the features of each node are derived from the metrics of VNFs deployed on the node and the link features are the composition of virtual links. We indicate the set of slices in the substrate network as a set $S = \{s_1, s_2, \ldots, s_u \ \ldots\}$, and thus the resources taken by the VNFs of each slice at node n_i and link $l_{i,j}$ are superposed by

$$f_i = \sum_{s_u \in S} nf_{u,i}^{Res} \tag{2}$$

$$f_{l_{i,j}} = \sum_{s_u \in S} B_u(i,j) \tag{3}$$

where f_i denotes the aggregated resources allocated to the VNFs running on a node n_i, and each VNF $nf_{u,i}$ deployed on node n_i belongs to a specific slice s_u. Similarly, $f_{l_{i,j}}$ is defined as the superposed physical link resources taken by the slices. $B_u(i,j)$ denotes the allocated link resource to slice s_u. Since a virtual link in a network slicing may cross multiple physical links that are known in advance when the slice was deployed, the same feature of $B_u(i,j)$ will be applied to all physical links.

Next, we design the DT based on the above graph model to produce E2E latency for each slice. The main principle is to obtain the state of each node and link in $\mathcal{G}$ based on their own features and the observed states of their neighbours. Since the traffic travels through a sequential of VNFs in a slice, the order of nodes and links remain the same in a slice, which means the resource utilisation of a directed link explicitly depends on the nodes it is connected. To enhance the efficiency of the model, we integrate the link features into node features by adding new features of the port number that sends out the traffic to a specific neighbour with the utilised bandwidth, unifying the node state h_i with link state h_l.

$$h_{l(i,j)} = h_i = \mathcal{P}\Big(\sum_{j \in \mathcal{N}(n_i)} f_i \sqcup f_{l_{i,j}} \Big), n_i, n_j \in \mathcal{V} \tag{4}$$

where $\sqcup$ denotes the concatenation of the two features, and $\mathcal{P}$ is the state generation method based on the observed features of the node and its neighbours.

The working mechanism of the proposed DT is shown in Fig. 2 and described in Algorithm 1. The model accepts the network graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, the node feature vectors $\{x_i\}, v_i \in V$ and the model depth K that indicates how many hops of neighbours' information to be observed as the inputs, and will return the predicted E2E latency of each slice d_{s_u}. Several components are designed in the model, including the sample function (line 4), aggregation method (line 5 and 6) and normalisation (line 7), which will be detailed in Section IV-B. The forward computational process of the model works iteratively layers by layers and nodes by nodes. Formally, the states of nodes at each layer can be represented as vectors $\{h_i^{(k)}\}, n_i \in \mathcal{V}, k \in \{1, 2, \ldots, K\}$, where $h_i^{(k)}$ denotes the states of n_i at the kth layer. For all the nodes, their states at layer 0 are initialized with their feature vectors, i.e., $h_i^{(0)} = \{x_i\}, v_i \in \mathcal{V}$. Given a node v_i, its neighbours can be represented as set $\Phi(n_i) = \{n_j | n_j \in \mathcal{V} \wedge (n_i, n_j) \in \mathcal{L}\}$. The designed DT will effectively aggregate the neighbours' states to learn a node embedding representation. Instead of directly working on the complete neighbourhood set $\Phi(n_i)$, the model uses a subset of the neighbours prior to information aggregation to improve the generalisation and efficiency. After we get the states $\{z_i\}$ of nodes in the graph $\mathcal{G}$ (line 10), we can further derive the slice state $h_s(u)$ (line 12). To integrate the states of the nodes a slice contains into a slice state, we define a parametric function that expresses the dependence of the state of a slice on the aggregation of the nodes of it, which can be written as $h_s(u) = \mathcal{I}(z_0, \ldots, z_{|V_u|}), s_u \in S$. In

Algorithm 1: GNN-based DT for E2E slicing

Input : Network $\mathcal{G} = (\mathcal{V}, \mathcal{L})$; input features of physical nodes $\{f_i\}_{n_i \in \mathcal{V}}$; depth K; initial weight for aggregation $\mathbf{W}$;
Output: E2E latency of slices $\{s_1, \ldots, s_u\}$

1 Initialise node states $h_i^{(0)} = f_i, \forall n_i \in \mathcal{V}$;
2 **for** $k \in \{1, 2, \ldots, K\}$ **do**
3 **for** $n_i \in \mathcal{V}$ **do**
4 $\mathcal{N}(n_i) = \text{Sample}\left(\Phi(n_i)\right)$;
5 $h_{\mathcal{N}(n_i)}^{(k)} = \mathcal{A}\big(\{h_j^{(k-1)} | v_j \in \mathcal{N}(n_i)\}\big)$;
6 $h_i^{(k)} = \sigma\big(\mathbf{W}^{(k)} \cdot (h_{\mathcal{N}(n_i)}^{(k)} \sqcup h_i^{(k-1)})\big)$;
7 $h_i^{(k)} = \text{Normalise}(h_i^{(k)})$;
8 **end**
9 **end**
10 $z_i = h_i^{(k)}, \forall n_i \in \mathcal{V}$;
11 **for** *each* $s_u \in S$ **do**
12 $h_s(u) = \mathcal{I}(z_0, \ldots, z_{|V_u|}), s_u \in S$;
13 $d_{s_u} = \mathcal{D}(h_s(u)), s_u \in S$;
14 **end**

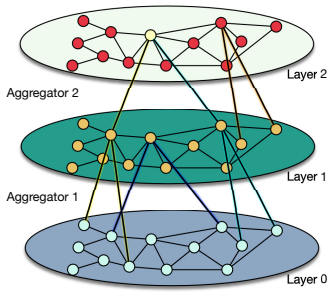


Fig. 3. An illustration of the sample and aggregation process of the GNN model.

this work, the function is implemented as the inner product iteratively for all nodes in the slice. Finally, with the slice states as input, we can generate the E2E latency (line 13). We define a function $\mathcal{D}(h_s(u)), s_u \in S$ to generate the E2E latency d_{s_u}, where $\mathcal{D}$ represents the dependence of the slice E2E latency on its state. This dependence is learned as a fully connected neural network with a linear activation. Next we will detail the proposed sample and aggregation method.

B. Samples and Aggregation of states

We design a sample and aggregation method to perform the local neighbourhood sampling and generate the states for the sampled neighbours. To make the model efficient and scalable, the designed neighborhood sampling method uniformly samples a fixed size of neighbours, instead of using a full-neighbourhood set, aiming to keep computing and memory complexity consistent when inferring a batch of target nodes with diverse degrees in parallel.

We show an example of neighbour sampling and aggregation in Fig. 3. In the figure, the sampling depth is set as $K = 2$, and $E_1 = E_2 = 2$ neighbours are sampled for each layer. The figure depicts that the states of nodes in the current layer only depend on the previous sampled layer. For instance, states of nodes in Layer 2 only need information from Layer 1. Between the layers, the aggregators accept a neighbourhood as input and combine each neighbour's states with weights to create a neighbourhood embedding. In other words, the state of the current node is derived from its own features and information from the node's neighbourhood. Each aggregator has its own weights that are learned during the training. The sampling process (Line 4 in Algorithm 1) is defined by

$$\mathcal{N}(n_i) = \begin{cases} \{n_i\}, \ k = 0 \\ \cup_{n_i \in \mathcal{N}^{k-1}_{(n_i)}} \mathcal{M}\big(\Phi(n_i), E_k\big), \ k = 1, 2, \ldots, K \end{cases} \tag{5}$$

where $\Phi(n_i)$ is a set of neighbouring nodes of n_i, E_k is the sampling size at depth k, and $\mathcal{M}$ is a defined function for sampling the neighbouring nodes $\Phi(n_i)$. The proposed DT model does not visit all the neighbours, therefore, $\mathcal{M}\big(\Phi(n_i), E_k\big)$ follows a uniform distribution $U\big(1, deg(n_i)\big)$ to randomly choose E_k nodes from $\Phi(n_i)$. In this way, the number of sampled neighbours of a single node grows with respect to the number of layers K, i.e., the total sampling size of $\cup_{k=1}^{K} \mathcal{N}^k(n_i)$ is $\prod_{k=1}^{K} E_k$.

After the sampling, we aggregate the states of nodes in the sampled set toward the original node n_i. The process is shown between Line 5 and 7 in Algorithm 1. The initial node states in layer 0, $h_i^{(0)}$ for a sampled set, are the input node features. Unlike traditional machine learning, the sampled neighbour node set has an orderless characteristic, since in an arbitrary topology of network each node has different number of neighbours. To achieve symmetric and orderless, we adopt the mean aggregator proposed in [24] in our model, which derives an average state from the states of the neighbouring nodes in the previous layer. The aggregation of states from the neighbours of node n_i (Line 5) can be depicted by

$$h^{(k)}_{\mathcal{N}(n_i)} = \frac{1}{|\mathcal{N}(n_i)|} \sum_{v_j \in \mathcal{N}(n_i)} \mathcal{P}^{\rho}_{drop}\left(h_j^{(k-1)}\right) \tag{6}$$

where $\mathcal{P}^{\rho}_{drop}$ is a random dropout with probability ρ to state vector of $h_j^{(k-1)}$. So the aggregation process of Lines 5 and 6 in Algorithm 1 can be written as

$$h_i^{(k)} = \sigma\Big(\mathbf{W}^{(k)} \cdot \big(\mathcal{P}^{\rho}_{drop}(h_i^{(k-1)}) \sqcup h^{(k)}_{\mathcal{N}(n_i)}\big) + b^k\Big) \tag{7}$$

where $h_i^{(k)}$ is the state presentation for node n_i at layer k, $W^{(k)}$ is the trainable weight matrix of Layer K aggregator for all nodes and b^k is the bias.

At last, the aggregated states are normalised by dividing with their modulus, and thus the $\mathrm{Normalise}$ function in Line 7 can be written as

$$h_i^{(k)} = \frac{h_i^{(k)}}{\parallel h_i^{(k)} \parallel_2}, \forall n_i \in \mathcal{V} \tag{8}$$

C. Training Algorithm

The GNN model in the DT is a general neural architecture capable of modeling various network performance metrics. In order to apply it to particular problems, we define the size of the hidden states for each node of slice (h_i), the size of hidden states for each slice (h_s), the number of aggregation layers K, and the number of sampled neighbours E_k as hyperparameters. The input layers of the DT depend on the size of K and E_k. Since the number of sampled neighbours $\mathcal{N}$ increase exponentially with these two parameters, a large sample depth can lead to a huge input set. We need to carefully choose their sizes. Normally, $K = 2$ and $E_k = 3$ are enough for most of the scenarios, which means we consider the effects of nodes that are 4 hops away. Each aggregator and state embedding layer is followed by a dropout layer. Typically, when a neural network is optimised to minimize the error for a particular output, the solution may be too optimistic. Thus, repeating an inference multiple times with random dropout can provide a probabilistic distribution of results, and this distribution can be used to generalise the predictions.

We train the model and optimise its performance in a stepping manner. In each step, the model will receive a matrix of network traffic generated by a set of slices and output the E2E latency for each slice under such scenario. In each training iteration, the aggregators of GNN are trained to generate similar states for the nodes that are near to each other, while

enforcing that the states of nodes far from each other are highly discrepant. After the processing of K layers, the final states of slices $h_s(u)$ are generated. They are fed into a dense layer to predict the E2E latency. The DT model is trained to minimise the Log-Cosh loss, $\sum_{i=1}^{n} \log(\cosh(y_i^p - y_i))$ between the predicted latency and the ground truth. The Log-Cosh loss function is chosen for its ability to smooth the impact of abnormal samples that are caused by network device failure, due to the logarithm of the hyperbolic cosine of the prediction error. It can work like Mean Squared Error (MSE) for normal training samples, but will not be significantly drifted by abnormal samples. Adam [25], a gradient-based optimisation technique is employed to tune the parameters of the aggregators, weight matrices $W^{(k)}$ and the prediction layer.

From a computational perspective, the loops over neighbours of different layers are the most time-consuming process of the model. An upper bound estimate of time complexity can be obtained as $O(\mathcal{N}(n_i)^K|\mathcal{V}||f_i|^2)$, where $\mathcal{N}(n_i)$ is the number of sampled neighbours per node, K is the number of layers, $|\mathcal{V}|$ is the number of nodes in the network, and $|f_i|$ is the size of features. To improve the training efficiency, the model is trained in a mini-batch manner, i.e., a fixed-size of neighbours are sampled. For each batch, the time complexity is determinate at $O(\prod_{k=1}^{K} E_k)$. The space complexity is $O(b\mathcal{N}(n_i)^K|f_i| + |\mathcal{V}||f_i|^2)$, where b is the training batch size.

V. Performance Evaluation

In this section, the effectiveness and accuracy of the developed DT are validated. We first describe the setting of our experiments under three different topologies, and then present the experimental results to demonstrate the accuracy of predicted E2E latency from the DT. At last, we show three use cases of the DT in resource allocation, link failure mitigation and latency violation monitoring.

A. Experiment environments

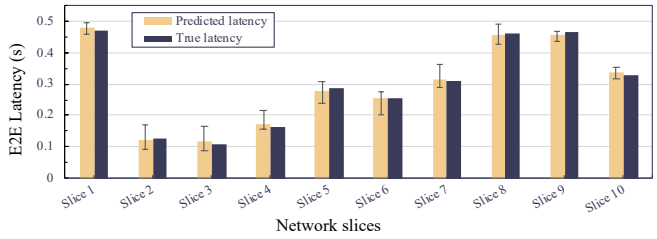
As there is no large-scale dataset that contains network slicing data that we could take advantage of in the literature, we convert the datasets generated in [26], which represents the pairwise source-destination traffic matrices in different topologies, routing scheme and traffic patterns to the slicing-style data. The datasets are chosen not only because of its generalised configurations, but also due to the per-source/destination measurements, which can mimic the E2E network slicing metrics that are needed for the performance evaluation. We transform the traffic matrices under three topologies into three network slicing scenarios to fit the E2E slicing model. The first dataset is based on the NSFNET network [27] including a topology of 14 nodes and 42 links with 10 slices deployed. The second dataset is collected from the GEANT2 [28] network that contains 24 nodes and 74 links with 30 slices deployed. The third dataset is a synthetically-generated network that embrace 50 nodes and 276 links with 50 slices deployed. The features of each VNF in a slice include the volume of traffic generated and the number of packets sent with different resource utilisation rates. The ground truth is the mean E2E latency of each slice under a given resource utilisation. Note that we keep the labels in the original dataset that were measured from the networks with predefined propagation delay, and thus the latency may be higher than the requirement of network slicing applications in 5G. The labelled latencies are only to evaluate the accuracy in predicting E2E latencies of the proposed model.

The proposed DT is implemented in TensorFlow and Stellargraph [29], a graph machine learning library. For each slice scenario, the graph model of the DT is trained for 2000 steps with a minibatch forward propagation manner with size 64. The parameters for the DT are set as follows: the rectified linear units are used as the activation function; $K = 2$ aggregators are trained to explore neighbours within 4 hops; and the state sizes are set as 256 for each VNF. The size of sampled neighbours at each layer is set as $E_1 = 3, E_2 = 3$ for NSFNET, $E_1 = 6, E_2 = 4$ for GEANT2, and $E_1 = 9, E_2 = 6$ for the synthetic network, respectively. To avoid over fitting, we set the dropout rate as 0.3. During the training, we minimise the Log-Cosh loss between the prediction and the ground truth, and use an Adam optimiser with an initial learning rate of 0.0005. To evaluate the scalability of the DT model, we convert the time series data into hundreds of graph screenshots, i.e., we construct a single graph using the slicing traffic matrices and the resources allocated to the slices at time slot t. We then combine 500 single graphs with different configurations into a huge graph, e.g., a combined GEANT2 graph containing 12000 nodes and 37000 links, and use it as the input graph for the developed DT model. To improve the generalisation of the DT, we remove 30 percent of the nodes including the links connecting them from the graph. The DT is trained on the reduced graph with the remaining 70 percent of the nodes from the original graph. Then, during the evaluation step, we re-integrate the removed nodes and manage to predict the E2E latency of slices without re-training the DT.

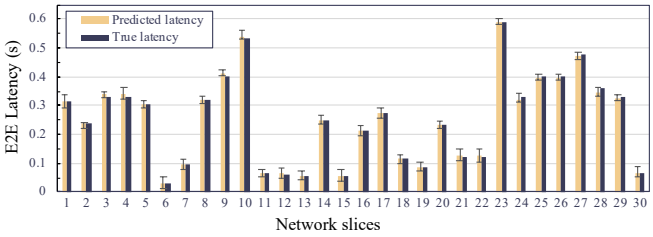
B. Performance evaluation of the DT

To demonstrate the accuracy of the proposed DT, we compare the predicted latencies of slices from the DT with the measured true latency. Fig. 4 depicts the results generated by the trained DT that are applied to three different scenarios with various number of slices. Each bar in the figure is the median values of the generated E2E latency predictions after running the DT for 30 times for each scenario. The error bars illustrate the range of 95% of the predictions. The results and predicted ranges show that the latencies predicted by the DT match well with the true latencies under all three scenarios, which have less than 5% error rate and small variance, validating the accuracy of the proposed DT.

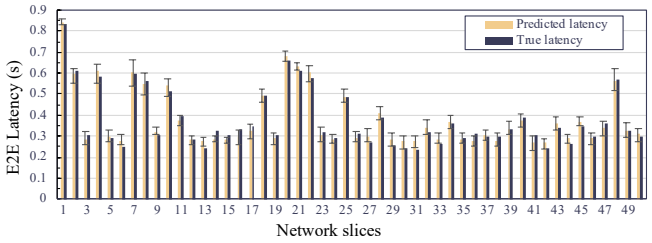
To evaluate the generalisation capability of the DT, we apply it on the GEANT2 topology but with different slice deployment strategies that were not seen during the training. In each experiment, we deploy different quantities of slices ranging from 10 to 100 with interval 10, and each slice has its own resource utilisation. The slices are deployed in the network using various strategies that are derived from 10 different routing algorithms. Besides the slice traffic, we add



(a) NSFNet with 10 slices deployed



(b) GEANT2 with 30 slices deployed



(c) Synthetic network with 50 slices deployed

Fig. 4. The predicted E2E latency of each slice vs. true latency under three network configurations.

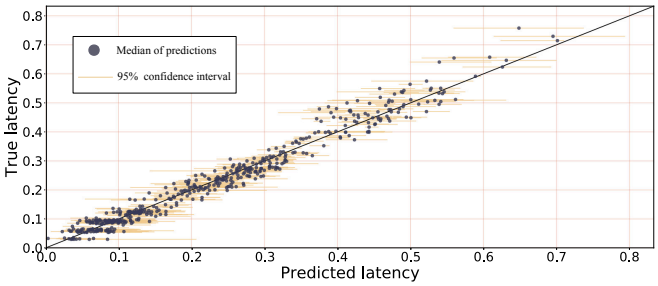


Fig. 5. Evaluating the generalisation capability of the model on GEANT2 with diverse configurations.

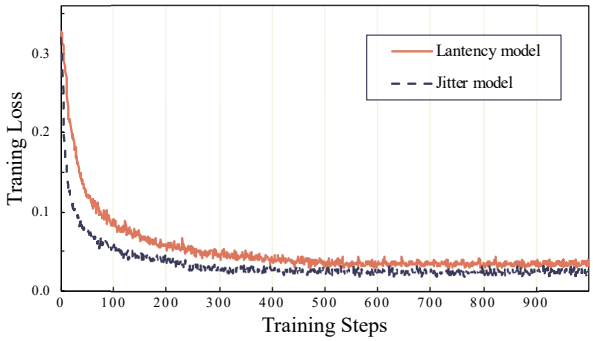


Fig. 6. Evaluating the adaptive capability of the DT for predicting new jitter metric on GEANT2 Topology.

background traffic between the nodes to represent the traffic generated by non-slicing applications and control signals. 550 pairs of predicted latencies and true latencies are collected through 10 experiments. In Fig. 5, we show the regression plot of the results on GEANT2 with 500 randomly selected latencies of slices. The dots represent the median value of the predictions and yellow lines show the 95% confidence level. The results show that the overall prediction error is considerably low, but the errors of high latency is more noticeable than low ones. This is caused by the fact that high latency slices are rare during the training, and they may be treated as abnormal points of the data during the training as they generate large gradients. The loss function Log-Cosh is capable of alleviating such effects so the information of such data points will be fully absorbed. However, this problem is relatively easy to solve by using an inclusive training set that contains a wider range of latency distributions.

The DT model is designed with generalisation and transferability in mind. Fig. 6 depicts the transfer learning capability of the DT, where we adapt the trained E2E latency model for predicting an unseen jitter metric of network slicing, using the same aggregation and sampling parameters and the features of nodes. By aggregating information from a few neighbouring nodes, the model can achieve good performance in just a few training steps. The results show that the original latency model is converged around 600 steps, but the training speed for the jitter metric is much faster (around 250 steps). The results indicate that the developed GNN-based DT has good adaptive capability and can be easily used to predict new metrics, which is important for working on the dynamic slicing network, where new services emerge everyday.

C. Potential exploitation and application

The developed DT can be used as an efficient tool to investigate the impact of key metrics on the E2E performance of slices, which can be used in network slicing orchestration and operation. We present three potential applications of the DT to network slicing operation tasks.

The resource utilisation of network slices is dynamic and time-varying. To guarantee the performance of slices, network operators need to know E2E slicing latencies under different resource utilisation situations in near real-time before making any decisions. With this regard, we use the DT to monitor the latency of certain network slices, and explore whether the Service Level Agreement (SLA) requirements are violated as the utilisation of the resources increases. We deploy 3 slices with E2E latency requirements on the GEANT2 network starting with low overall resource utilisation (30%). To illustrate high network loads situations, we increase both the slice traffic and the background traffic (which is not generated by the applications of slices) by 30% in each experiment, through multiplying the traffic volume by 1.3. We repeat the experiment 4 times and the results are shown in Fig. 7 that presents the changes of slice latencies against different resource utilisations. It demonstrates that the DT can be used to monitor the SLA violation through comparing the predicted

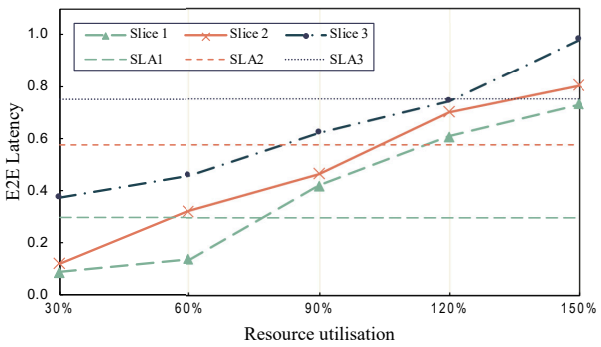


Fig. 7. Monitored latency of slices with different SLA requirements vs. dynamic resource utilisation.

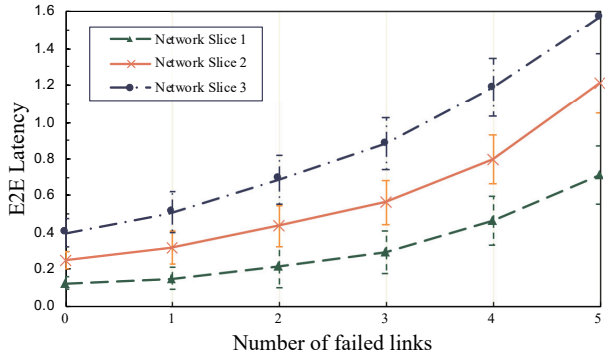


Fig. 8. Evaluating the adaptive ability of the DT under link failure event for finding new paths for slices and predicting the latencies after the changes on GEANT2 topology.

performance of slices with the predefined SLA, and can generate alert when the latency has been violated. For example, when the network utilisation increases to 90%, the latency of Slice 1 violates the SLA and needs to either increase the allocated resources or to migrate some VNFs to other nodes and links with low resource utilisation, which can still satisfy the QoS requirement.

Next, we evaluate the adaptiveness of the proposed DT model under the presence of a rare event, such as link failures. When a certain link with slices running on fails, it is necessary to find a new path that avoids this link and migrate the affected slices. In the experiment, three slices with low resource utilisation are deployed in GEANT2 network and any two slices have at least one shared physical link. We randomly remove links that have slices deployed on them, and use the shortest path routing to find an alternative path for the affected slices. We remove 1 to 5 links each time and exploit the DT to predict new E2E latency after migrating the three slices. Fig. 8 depicts the adaptive ability of the DT model. Each point in the figure is the mean latency obtained from 5 experiments. As the number of failed links increases, the latencies of all three slices become higher. Due to less paths available, the slices are sharing more links and occupy more resources. As shown in the results, the proposed DT can work with rare events and generate latencies for slice migrations, which provide valuable insights for resource optimisation.

At last, we investigate that how to leverage the DT for

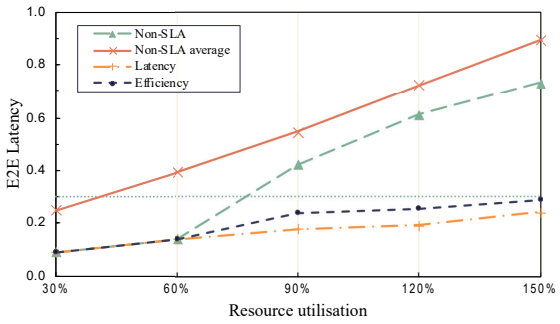


Fig. 9. Deployment optimisation for Slice 1 to achieve SLA guarantee.

optimising the decision-making when the SLA of a certain slice is not met. Fig. 9 depicts the optimisation of Slice 1 to satisfy the latency requirement under the same environment as Fig. 7. A simple greedy-based optimiser is developed to search for deployment solutions when the SLA is violated. We design two objectives, which aim to minimise the latency and maximise the network efficiency respectively, while ensuring the SLA. The green line shows the latency changes of Slice 1 and the red solid line denotes the average latency of Slice 2 and 3 without any optimisation. The line labelled as "Latency" represents the results of the solution with minimum latency among all possible deployment cases that have enough resources for Slice 1. The "Efficiency" optimisation balances the resource utilisation of all physical nodes and links, which keeps high scalability of slices and remains high possibility to embed new slices. The results demonstrate that the DT can be easily exploited for various network operation and optimisation tasks and can also be integrated with other machine learning frameworks, for example using Reinforcement Learning to replace the greedy optimiser for autonomous network management.

VI. Conclusions

In this paper, we investigated the key challenges in ensuring the E2E network slicing performance in Industry 4.0 and various 5G applications, and developed a network DT for network slicing. We exploited the state-of-the-art GNN model to solve the E2E slicing challenges, by developing a virtual representation to construct a graph from the network consisting of slices and discovering insights directly on the graph, instead of converting the network into matrices. We proposed a GNN-based DT based on an inductive graph framework to generate feature embeddings of the network slices represented as graphs, which were then used to predict the E2E metrics. The experimental results showed that the proposed DT is able to monitor the E2E performance of slices through generating the accurate predictions of slice latencies. We also leveraged the DT as a cost-effective tool to monitor SLA violations, mitigate the impact of link failures and find optimal deployment solutions for slices.

References

[1] A. Banchs, D. M. Gutierrez-Estevez, M. Fuentes, M. Boldi, and S. Provvedi, "A 5G Mobile Network Architecture to Support Vertical Industries," *IEEE Commun. Mag.*, vol. 57, no. 12, pp. 38–44, 2019.

[2] S. Zhang, "An Overview of Network Slicing for 5G," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 111–117, 2019.

[3] A. E. Kalor, R. Guillaume, J. J. Nielsen, A. Mueller, and P. Popovski, "Network Slicing in Industry 4.0 Applications: Abstraction Methods and End-to-End Analysis," *IEEE Trans. Ind. Informat.*, vol. 14, no. 12, pp. 5419–5427, 2018.

[4] D. Lopez, J. Ordonez-Lucena, P. Ameigeiras *et al.*, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80 – 87, 2017.

[5] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429 – 2453, 2018.

[6] H.-T. Chien, Y.-D. Lin, C.-L. Lai, and C.-T. Wang, "End-to-End Slicing as a Service with Computing and Communication Resource Allocation for Multi-Tenant 5G Systems," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 104–112, 2019.

[7] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2405–2415, 2019.

[8] D. Silver, A. Huang, C. J. Maddison *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484 – 489, 2016.

[9] Y. Zuo, Y. Wu, G. Min, and L. Cui, "Learning-based network path planning for traffic engineering," *Future Generation Computer Systems*, vol. 92, pp. 59 – 67, 2019.

[10] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource Management with Deep Reinforcement Learning," in *HotNets-XV*. ACM, nov 2016, pp. 50 – 56.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, vol. 60, no. 6, pp. 84 – 90, 2017.

[12] U. Paul, J. Liu, S. Troia, O. Falowo, and G. Maier, "Traffic-profile and machine learning based regional data center design and operation for 5G network," *J. Commun. Networks*, vol. 21, no. 6, pp. 569–583, 2019.

[13] S. Rathore, J. H. Ryu, P. K. Sharma, and J. H. Park, "DeepCachNet: A Proactive Caching Framework Based on Deep Learning in Cellular Networks," *IEEE Netw.*, vol. 33, no. 3, pp. 130–138, 2019.

[14] F. Eichhorn, M.-I. Corici, T. Magedanz, P. Du, Y. Kiriha, and A. Nakao, "SDN enhancements for the sliced, deep programmable 5g core," in *CNSM*. IEEE, nov 2017, pp. 1–4.

[15] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, "On Multi-Domain Network Slicing Orchestration Architecture and Federated Resource Control," *IEEE Netw.*, vol. 33, no. 5, pp. 242–252, 2019.

[16] J. Nightingale, P. Salva-Garcia, J. M. A. Calero, and Q. Wang, "5G-QoE: QoE Modelling for Ultra-HD Video Streaming in 5G Networks," *IEEE Trans. Broadcast.*, vol. 64, no. 2, pp. 621–634, 2018.

[17] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, 2000.

[18] J. Liebeherr, A. Burchard, and F. Ciucu, "Delay Bounds in Communication Networks With Heavy-Tailed and Self-Similar Traffic," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 1010–1024, 2012.

[19] W. Miao, G. Min, Y. Wu *et al.*, "Stochastic Performance Analysis of Network Function Virtualization in Future Internet," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 613 – 626, 2019.

[20] A. T. Z. Kasgari, W. Saad, M. Mozaffari, and H. V. Poor, "Experienced Deep Reinforcement Learning with Generative Adversarial Networks (GANs) for Model-Free Ultra Reliable Low Latency Communication," *IEEE Trans. Commun.*, pp. 1–1, 2020.

[21] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61 – 80, 2008.

[22] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *ArXiv*, 2019.

[23] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv*, 2016.

[24] W. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *NIPS*, 2017, pp. 1024–1034.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015.

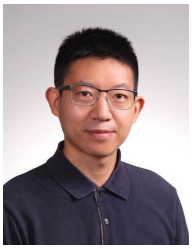
[26] K. Rusek, J. Suárez-Varela, A. Mestres *et al.*, "Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN," in *SOSR '19*, 2019, pp. 140 – 151.

[27] X. Hei, J. Zhang *et al.*, "Wavelength converter placement in least-load-routing-based optical networks using genetic algorithms," *Journal of Optical Networking*, vol. 3, no. 5, pp. 363–378, 2004.

[28] F. Barreto, "Fast Emergency Paths Schema to Overcome Transient Link Failures in OSPF Routing," *Int. J. Comput. Net. Commun.*, vol. 4, no. 2, pp. 17–34, 2012.

[29] C. Data61, "StellarGraph Machine Learning Library," *GitHub Repository*, 2018.

Haozhe Wang is currently a Research Associate with the Computer Science Department within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, UK. He received his B.S. and M.S. degrees from Dalian University of Technology, Dalian, China in 2005 and 2012, respectively, and the Ph.D. degree from University of Exeter, Exeter, UK, in 2017. His research interest includes network slicing resource management, learning-based network optimisation, future Internet architecture, Information-Centric Networking and network analytical modelling.



Yulei Wu is a Senior Lecturer with the Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, United Kingdom. He received the B.Sc. degree (First Class Honours) in Computer Science and the Ph.D. degree in Computing and Mathematics from the University of Bradford, United Kingdom, in 2006 and 2010, respectively. His expertise is on intelligent networking, and his main research interests include computer networks, networked systems, software defined networks and systems, network management, and network security and privacy. He is an Editor of IEEE Transactions on Network and Service Management, IEEE Transactions on Network Science and Engineering, IEEE Access, and Computer Networks (Elsevier) and. He is a Senior Member of the IEEE, and a Fellow of the HEA (Higher Education Academy).

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Computer Networks, Wireless Communications, Parallel and Distributed Computing, Ubiquitous Computing, Multimedia Systems, Modelling and Performance Engineering.

Wang Miao received his Ph.D. degree in Computer Science from the University of Exeter, United Kingdom in 2017. He is currently a Postdoctoral Research Associate at the College of Engineering, Mathematics, and Physical Sciences of the University of Exeter. His research interests focus on Network Function Virtualization, Software Defined Networking, Unmanned Aerial Networks, Wireless Communication Networks, Wireless Sensor Networks, Edge Artifical Intelligence, and Performance Modelling and Analysis.