A Graph Neural Networks based Framework for Topology-Aware Proactive SLA Management in a Latency Critical NFV Application Use-case

NIKITA JALODIA\textsuperscript{1,2,3}, (Member, IEEE), MOHIT TANEJA\textsuperscript{1,4}, (Member, IEEE), AND ALAN DAVY\textsuperscript{1,2,3}, (Senior Member, IEEE)

\textsuperscript{1}Walton Institute for Information and Communication Systems Science, South East Technological University, Waterford, X91WR86, Ireland
\textsuperscript{2}Department of Computing and Mathematics, School of Science and Computing, South East Technological University, Waterford, X91K0EK, Ireland
\textsuperscript{3}CONNECT—Centre for Future Networks and Communications, Dublin, Ireland
\textsuperscript{4}Department of Accounting and Economics, School of Business, South East Technological University, Waterford, X91 TX03, Ireland

Corresponding author: Nikita Jalodia (e-mail: nikita.jalodia@waltoninstitute.ie, nikita.jalodia@postgrad.wit.ie).

This work has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

ABSTRACT Recent advancements in the rollout of 5G and 6G have led to the emergence of a new range of latency-critical applications delivered via a Network Function Virtualization (NFV) enabled paradigm of flexible and softwarized communication networks. Evolving verticals like telecommunications, smart grid, virtual reality (VR), industry 4.0, automated vehicles, etc. are driven by the vision of low latency and high reliability, and there is a wide gap to efficiently bridge the Quality of Service (QoS) constraints for both the service providers and the end-user. In this work, we look to tackle the over-provisioning of latency-critical services by proposing a proactive SLA management framework leveraging Graph Neural Networks (GNN) and Deep Reinforcement Learning (DRL) to balance the trade-off between efficiency and reliability. To summarize our key contributions: 1) we compose a graph-based spatio-temporal multivariate time-series forecasting model with multiple time-step predictions in a multi-output scenario, delivering 74.62% improved performance over the established baseline state-of-art model on the use-case; and 2) we leverage realistic SLA definitions for the use-case to achieve a dynamic SLA-aware oversight for scaling policy management with DRL.

INDEX TERMS Network function virtualization (NFV), machine learning (ML), deep learning, graph neural networks (GNN), recurrent neural networks (RNN), convolutional neural networks (CNN), reinforcement learning (RL), quality of service (QoS), service level agreements (SLA)

I. INTRODUCTION

THE next generation of communication networks continue to be driven by a fundamental restructuring in the way that the networks and services are deployed and delivered. 5G’s usage scenario of ultra-reliable low-latency communications (URLLC) is further expected to extend in scope to a high-throughput ubiquitous global connectivity at scale, driving all major verticals towards a change \cite{1}. Network programmability and softwarization are the key drivers of this shift that extends to the vision of 6G \cite{1}, and are delivered via the concepts of Software Defined Networking (SDN) and Network Function Virtualization (NFV) \cite{2}. These continue to play a pivotal role in the vision of 6G, and together form the backbone of flexible and intelligent networks.

As Cloud infrastructure transitions to serve the next-generation requirements of upcoming softwarized application verticals \cite{2}, this is met with evolutionary challenges of incremental reliability guarantees that are expected to outdo legacy systems with specialized infrastructure \cite{1}. To this
end, network slicing clusters applications with similar demands in an appropriate Cloud environment [3], ensuring the placement of latency-critical URLLC applications in a high availability slice where resources are suitably provisioned to ensure reliability.

An SLA often invokes a trade-off between system usage and requirements, i.e., efficiency and reliability [4]. Over-provisioning is a common practice adopted by network and Cloud operators to match the high requirements for these latency-critical applications in a high availability network slice, but this is inefficient in the long run [5]. A transition towards complete softwarization of networks and services brings in the requirement to adopt more complex models to guarantee reliability and Quality of Service (QoS) [6]. A key aspect to driving such a change is in how the Cloud reacts to such a latency-critical demand, and in being precisely proactive over time [5], [7].

In this work, we present a graph-based deep learning framework for dynamic and proactive SLA management in the use case of a latency-critical NFV application. An overview of the system and scope is provided in FIGURE 1. The key contributions are summarised as follows:

- We work with a real-world deployment of a latency critical NFV application with two months’ worth of raw network telemetry data sampled every 30 seconds, and use that as the basis for all our policy formation and learning models.
- We compose a graph-based multivariate time-series forecasting model with multi-step predictions in a multi-output scenario, i.e. the model forecasts a sequential range of future values for multiple features in one go, thereby enabling us to track a realistic deployment setup. Further, we propose the suitability of a topology-aware Graph Neural Network (GNN) based model as opposed to traditional Recurrent Neural Network (RNN) methodologies when applied to such a task. Specifically, our implementation of a Gated Recurrent Unit (GRU) based Graph Convolutional Recurrent Network (GCRN) model demonstrates a 74.62% improvement in performance over the established state-of-art model [8] on the use-case.
- We leverage realistic Service Level Objective (SLO) definitions defined in our previous work [7] to compose a Q-learning based Deep Reinforcement Learning (DRL) model to achieve dynamic SLA-aware policy enforcement for such a latency critical use-case in an operational setting.

To the best of our knowledge, this is the first approach in the area that proposes a GNN and DRL based framework for proactive SLA management, and widely outperforms the previously established residual connections based Long Short-Term Memory (Residual-LSTM) model benchmark [8] in the use-case’s predictive objectives.

The rest of the paper has been structured as follows: §II elaborates on the background and presents the related work,
describes the Clearwater NFV application, and provides an overview of the SLA leveraged in the framework. §IV describes the proposed framework in detail. Thereafter, §V provides details of the experimental setup, §VI discusses the results obtained through the proposed framework, and §VII presents the concluding remarks, with the scope of future work.

II. MOTIVATION AND RELATED WORK
A transition towards softwarized networks demands higher complexity in models to guarantee reliability and QoS [6], and improve policy-based QoS management [9]. This is because of an impending evolution in not just the way networks are composed and managed, but also renewed application architectures [6], [10], corresponding QoS and SLA management techniques [11], and optimization and automation to cope with the added complexity [12].

While service operators come up with new dynamic [13] and predictive [14]–[16] scaling policies to match the demand facing the current generation of Cloud based application services, these are still a long way to go towards supporting latency-critical applications with high availability values [7]. Post appropriate provisioning following anticipated and identified traffic patterns, there is not a lot of work that directly addresses the remaining SLA bottlenecks from an application perspective. Automated SLA management for use-cases deployed on softwarized networks has been highlighted to be a critical requirement for next generation networks [17]–[22].

Much of the work done so far addresses QoS with characterizing and anticipating traffic patterns, anomaly detection [23], and a combination of reactive and proactive scaling policies [24], [25], [26]. Related work in the area explores anticipated network traffic based clustering and forecasting [27], [28], machine learning based network traffic classification in NFV [29], [30], and related resource allocation [4], [5]. Our previous work [7] in the area proposes the use of multi-label classification methodology for a multi-output SLO violation prediction in NFV environments. From a feature forecasting perspective, authors in [31] present an overview of non-linear and linear forecasting methodologies to improve multi-slice resource management in 5G networks. LSTM based approaches [31]–[33] have also been successfully leveraged for resource forecasting within communication networks. Our previous work in the area [8] benchmarks various deep learning based forecasting methodologies, and proposes a Residual LSTM based framework for proactive SLA management in rapid forecasting based resource monitoring of latency sensitive NFV applications.

However, networks can inherently be represented in a graph based format of nodes and edges, and so can the application structure and flow of data in the softwarized domain of SDN and NFV [4], [5]. In traditional Machine Learning (ML) approaches, these spatial inter-dependencies are removed in the pre-processing stage, and so the inherent effect of these topological dependencies are not a part of the learning and prediction stage [34]. Graph neural networks (GNNs) are a family of neural networks that deal with signals defined over graphs. Modern GNNs are categorized into four groups: recurrent GNNs, convolutional GNNs, graph autoencoders, and spatial–temporal GNNs [35]. Further, in contrast to conventional Machine Learning (ML) approaches, a GNN based approach is capable of producing accurate predictions even when the underlying topology is changed from what the model was trained on [36].

Within the wider domain of networks, GNN has been successfully used in problem areas addressing networking performance and generalisation to larger networks [36], radio resource management [37], etc. Such methodologies have also been successfully used in the area of SDN, addressing connection management [38], energy-efficient VNF deployment [39], spatio-temporal link state prediction [40], detecting and mitigating data plane attacks [41], predicting the optimal path for Service Function Chain (SFC) deployment and packet-level traffic steering [42], etc. Within the domain of NFV, GNN based methodologies have proven successful in problem areas addressing network slicing management [43], VNF deployment prediction [44], VNF resource prediction [4], finding optimal SFC path [45], etc.

Since GNN successfully extracts and models spatial features and topological dependencies, there is an intrinsic potential to use this in conjunction with a reinforcement learning (RL) based approach [5]. Authors in [46] have used graph convolutional network (GCN) based GNN with actor-critic based DRL towards network planning. The combination of GNN and DRL has also been successfully leveraged to tackle power grid management [47]. Within the domain of SDN, message passing GNN with deep Q-learning has been used towards connection management [38], and routing optimization [48]. Authors in [39] have used graph convolutional network (GCN) based GNN with double deep Q-network (DDQN) towards energy-efficient VNF deployment. Within the subject area of NFV, graph network based GNN and DRL have been effectively used towards NFV flow migration [49], [50], and optimal VNF placement [51]. Graph convolutional network based GNN, and DRL have also been leveraged to address the problem areas involving VNF forwarding graph placement [52], and for virtual network embedding [53], [54].

This publication is an extension of our previous work in the area—our first publication [7] in the series fulfilled the gap of a lack of realistic SLO definitions used within proposed models in research. We also proposed [7] the use of multi-label classification methodology for a multi-output SLO violation prediction in NFV environments. Next, our work in [8] benchmarked various deep learning based forecasting methodologies, and proposed a Residual LSTM based framework for proactive SLA management in rapid forecasting based resource monitoring of latency sensitive NFV applications. To the best of our knowledge, the proposition in this work is the first approach in the area that...
proposes a spatio-temporal Graph Convolutional Recurrent Network (GCRN) model for the use-case, and benchmarks its performance against conventional deep learning models that have demonstrated favorable performance on the use-case in the past. We also use realistic SLO definitions to propose an SLA-aware deep Q-learning based DRL model, packaged together in a framework that delivers topology-aware proactive SLA management in a latency-critical NFV application.

III. SERVICE LEVEL AGREEMENTS (SLA)

An SLA is a qualitative measure that binds the service provider and facilitator into a formally agreed contract ensuring QoS for the end user. This is realised quantitatively on a set of low level metrics delivered through SLOs and Service Level Indicators (SLIs). The SLIs are quantitative measures that build upon raw system metrics such as those presented in Table 1, which further feed into the SLOs as a definitive target range or threshold towards the deliverance of an SLA.

\[
\text{SLI} \leq \text{target\ threshold} \quad (1)
\]

\[
\text{lowerbound} \leq \text{SLI} \leq \text{upperbound} \quad (2)
\]

The breach of an SLA implies an explicit consequence, often financial; while the SLOs and SLIs are typically measurable indicators that define the policy of tolerance with measurable service characteristics [55].

A. PROJECT CLEARWATER CLOUD IMS

In our work, we use the open-source Project Clearwater as the use-case for a Cloud based virtualized NFV application—it has been widely used in research as a standard test-bed setup for NFV related work [4]–[8], [56], [57]. It consists of 6 main components (VNFCs)— Bono, Sprout, Homestead, Homer, Ellis, and Ralf. A high level view of these VNFCs and their functionalities replicating a standard IP Multimedia Subsystem (IMS) architecture is as shown in FIGURE 2.

B. CLEARWATER SERVICE LEVEL OBJECTIVES (SLO)

We use raw network telemetry data and system monitoring metrics obtained via a standard realization of the Clearwater test-bed setup to define the SLIs and SLOs governing an informal SLA. We utilise these metrics as the foundations for the SLIs, which when matched with a target threshold or range form SLOs. These metrics were collected on a 30 second sampling frequency through Monasca—an open-source Python based monitoring service running on each of the Clearwater VNFCs. The list of these collected metrics is presented in Table 1 and further details regarding the data is elaborated upon in Section V.

In our previous work [7], we fulfill the gap of a lack of realistic SLO definitions used within proposed models in research, and define Clearwater SLOs by using a combination of over two SLIs while drafting each SLO rule [57]. To highlight the varying reason behind the loss of QoS at any time, we define four SLOs for the Clearwater VNF— SLO_1 (load), SLO_2 (computation), SLO_3 (disk), and SLO_4 (input/output, or IO). In-depth elaboration on these SLO definitions has been provided in our previous work [7].

Formally, let \( L \) denote the set of SLOs thus defined:

\[
L = [\text{SLO}_1, \text{SLO}_2, \text{SLO}_3, \text{SLO}_4] \quad (3)
\]

Equivalently:

\[
L = [\text{SLO}^{\text{load}}, \text{SLO}^{\text{computation}}, \text{SLO}^{\text{disk}}, \text{SLO}^{\text{io}}] \quad (4)
\]

The metrics captured by Monasca are at the granularity of the individual VNFCs as shown in FIGURE 2 and an SLO violation at any of the individual VNFCs triggers an SLO violation state for the Clearwater application service. Therefore, we ultimately define the SLOs at the application level, i.e. for the entire VNF as an application service. Thus, each data instance is associated with 4 SLOs as defined by \( L \) above, where \( SLO_j, j \in |L| \) assumes one of two states:

\[
SLO_j = \begin{cases} 
1, & \text{if Violated (at any VNFC)} \\
0, & \text{otherwise} 
\end{cases} \quad (5)
\]

IV. TOPOLOGY-AWARE PROACTIVE SLA MANAGEMENT FRAMEWORK

Given that the aim is to be able to proactively mitigate the potential SLA violations given the time-series of tracked system and application metrics at a set sampling frequency, we decompose the problem as that of continuous feature forecasting, followed by a reinforcement learning methodology that oversees the scaling policy to avoid potential SLA violations based on the forecasts.
TABLE 1: Raw metrics collected through Monasca during Clearwater vIMS application monitoring. This data is available for each of the VNFCs, and is sampled every 30 seconds.

| Metric Name             | Semantics                                                                 |
|-------------------------|---------------------------------------------------------------------------|
| CPU                     |                                                                           |
| cpu.idle_perc           | Percentage of time the CPU is idle when no IO requests are in progress    |
| cpu.system_perc         | Percentage of time the CPU is used at the system level                    |
| cpu.wait_perc           | Percentage of time the CPU is idle AND there is at least one IO request in progress |
| Disk                    |                                                                           |
| disk.inode_used_perc    | The percentage of inodes that are used on a device                        |
| disk.space_used_perc    | The percentage of disk space that is being used on a device               |
| IO Read                 |                                                                           |
| io.read_kbytes_sec      | Kbytes/sec written by an IO device                                        |
| io.read_req_sec         | Number of read requests/sec to an IO device                               |
| io.read_time_sec        | Amount of read time in seconds to an IO device                             |
| IO Write                |                                                                           |
| io.write_kbytes_sec     | Kbytes/sec written by an IO device                                        |
| io.write_req_sec        | Number of write requests/sec to an IO device                              |
| io.write_time_sec       | Amount of write time in seconds to an IO device                            |
| Load                    |                                                                           |
| load.avg_1_min          | The normalized (by number of logical cores) average system load over a 1 minute period |
| load.avg_15_min         | The normalized (by number of logical cores) average system load over a 15 minute period |
| load.avg_5_min          | The normalized (by number of logical cores) average system load over a 5 minute period |
| Memory                  |                                                                           |
| mem.free_mb             | Mbytes of free memory                                                     |
| mem.usable_mb           | Total Mbytes of usable memory                                             |
| mem.usable_perc         | Percentage of total memory that is usable                                  |
| Network                 |                                                                           |
| In                      |                                                                           |
| net.in_bytes_sec        | Number of network bytes received per second                              |
| net.in_packets_sec      | Number of network packets received per second                             |
| Out                     |                                                                           |
| net.out_bytes_sec       | Number of network bytes sent per second                                   |
| net.out_packets_sec     | Number of network packets sent per second                                 |

A. GRAPH NEURAL NETWORK (GNN) BASED CLEARWATER FEATURE FORECASTING

Recurrent neural networks (RNN) are a class of neural networks that are powerful for modeling sequential data such as time-series, and are especially crafted towards such use-cases [7]. An RNN layer maintains an internal state that encodes information about the time-steps it has seen so far. However, traditional forecasting by way of evolved RNN models like GRU and LSTM consider only temporal information of features for sequence modeling.

When considering use-cases with additional spatial dependencies between features, relying solely on temporal variation likely imposes a cap on performance no matter how complex the RNN model in use. When input data can inherently be represented in a graph-based format of nodes and edges, the temporal flow of information between nodes also involves spatial dependencies. GNNs are a family of neural networks that deal with signals defined over graphs.

As such, Graph Convolutional Recurrent Network (GCRN) is an evolved version of GNN that is a generalisation of classical RNN to data structured in a graph format, and can be defined as a deep learning model capable of predicting structured sequences of data. For computational efficiency, GCRN combines Convolutional Neural Networks (CNN) on graphs to identify spatial structures, and RNN to find dynamic patterns [58].

Consider $x_t: \mathcal{V} \rightarrow \mathbb{R}^{d_x}$ defined on the nodes of the Clearwater graph $\mathcal{G} = (V, \mathcal{E}, A)$ as shown in FIGURE 3 where $\mathcal{V}$ is a finite set of $|\mathcal{V}| = n = 6$ vertices denoting the 6 Clearwater VNFCs. $\mathcal{E}$ as the set of edges, and $A \in \mathbb{R}_r^{n \times n}$ as the adjacency matrix denoting the (optional) weight of connection between two vertices. Therefore, a signal $x_t: \mathcal{V} \rightarrow \mathbb{R}^{d_x}$ defined on the nodes of the Clearwater graph may be regarded as a matrix $x_t \in \mathbb{R}_r^{n \times d_x}$, whose column $i$ is the $d_x$-dimensional value of $x_t$ at the $i^{th}$ node.

FIGURE 3 provides an overview of the GCRN-based Clearwater feature forecasting model.
Algorithm 1: A GNN-based Framework for Topology-Aware Proactive SLA Management in a Latency Critical NFV Application Use-case

• PRE-PROCESSING
  Input: Data: $D \in \mathbb{R}^d$
  procedure DATA SPLITS
  | Split the data in train, validation, and test sets
  end
  procedure DATA TRANSFORMATION
  | Data standardization and normalization
  end
  procedure DATA WINDOWING AND BATCHING
  | Split data into sliding windows of features and targets
  end

• GCRN FORECASTING MODEL
  Input: Train and validation data windows from DATA WINDOWING
  Define Clearwater VNFC graph nodes
  Initialize Clearwater graph edge indices (connections)
  repeat
  | $X =$ Input data features sliding tensor slice
  | $Y =$ Expected data features sliding tensor slice
  | $\hat{Y}$ = Forecasted data features sliding tensor slice
  | GCRN Model Layers ($(in, out, K)$):
    | GConvGRU layer (Clearwater edge indices)$^7$
    | Linear layer
  | $\theta =$ Return: MSE loss between ($\hat{Y}$, $Y$)
  until Convergence;
  Output: Forecasted features $\hat{Y}$

• DEEP REINFORCEMENT LEARNING MODEL
  Input: $x_t \in D$
  Initialize $Q$ network function with random weights $\theta$
  Initialize target $Q$ function with random weights $\theta$
  Initialize experience replay memory $\mathcal{E}$ to set capacity
  repeat
    | TRAINING DEEP Q-NETWORK
      | Observe state $s_t$ (i.e. $x_t$) of Clearwater forecasting environment at time $t$
      | Policy $\pi$: $\epsilon$-greedy
        | if Exploration (with probability $\epsilon$) then
          | Randomly select action $a_t$
        | else
          | Select action $a_t = \arg\max_a Q(s_t; a; \theta)$
        end
      | Record reward $r_t$ for action $a_t$
      | Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{E}$
      | Sample random minibatch of transitions $(s_i, a_i, r_i, s_{i+1})$ from $\mathcal{E}$
      | Perform a gradient descent step on $(r_i - Q(s_i, a_i; \theta))^2$ with respect to $\theta$
      | Reset $\hat{Q} = Q$ periodically
  until Training Episode Counter Complete;
  Output: Trained reinforcement learning model

\[
\hat{Y}_t, ..., \hat{Y}_{t+F} = \arg\max_{Y_t, ..., Y_{t+F}} P(Y_t, ..., Y_{t+F} | X_{t-H}, ..., X_{t-1}) \quad (6)
\]

$X_t \in D$ here is the matrix of the numeric state of each of the 21 features for the 6 VNFCs as observed at time $t$, $Y_t \in D$ denotes the corresponding matrix of expected data features, and $\hat{Y}_t \in D$ denotes the corresponding matrix of forecasted data features. $P$ models the probability of expected features $Y$ in a window of size $F$ to appear conditioned on the past $X$ features in a window of size $H$. Since we have spatial dependencies, the features of observations within $X, Y$, and $\hat{Y}$ are linked by pairwise relationships, modelled by $G$.

Since we use a GRU based GCRN, the model (GConvGRU) can be represented as:

\[
y = \sigma(W_{xz} \cdot G \cdot X_t + W_{hz} \cdot G \cdot h_{t-1}),
\]

\[
r = \sigma(W_{xr} \cdot G \cdot X_t + W_{hr} \cdot G \cdot h_{t-1}),
\]

\[
\tilde{h} = \tanh(W_{zh} \cdot G \cdot X_t + W_{hh} \cdot G \cdot (r \odot h_{t-1}))
\]

\[
h_t = z \odot h_{t-1} + (1-z) \odot \tilde{h}
\]

$z, r, \tilde{h}$, and $h_t$ are GRU parameters—$z$ represents the update gate, $r$ represents the reset gate, $h_t$ represents the hidden state at time $t$, and $\tilde{h}$ represents the new hidden state. Further, $\odot$ represents the Hadamard product, and $\sigma$ represents the logistic sigmoid function. $^*$G then represents the graph convolution operator, and the support $K$ of the graph convolutional kernels defined by the the Chebyshev coefficients $W_x$ and $W_h$ determines the number of parameters independent of the number of nodes $n$.

Algorithm 1 presents the procedural workflow adopted towards achieving such a model that learns spatio-temporal structures from graph-structured and time-varying data. We evaluate this methodology in the following sections, benchmarking it against the current state-of-art on the use-case.

B. DEEP REINFORCEMENT LEARNING (DRL)

When it comes to dynamic application scenarios, RL is an effective machine learning methodology. A RL agent directly interacts with the environment to form a policy for decision-making based on a reward mechanism, that is customizable to achieve the desired outcomes. Specifically, RL follows the Markov Decision Process (MDP) model to train an agent that iteratively observes the state $s_t$ of the environment at a discrete time step $t$ to prescribe action $a_t$ that maximizes the reward $r_t$. Cumulating and maximizing the expectation of rewards over time, RL thereby attains an efficient policy $\pi$ for stochastic scenarios. Q-Learning in this regard is a typical off-policy model-free RL algorithm that calculates the value of each state-action pair as a Q-value function $Q(s, a)$:

\[
Q(s_{t+1}, a_{t+1}) = \mathbb{E}[r_t + \gamma \max_a Q(s', a') | s_t, a_t]
\]

Then, based on the policy $\pi = P(a | s)$ (e.g. $\epsilon$-greedy), it chooses the action with the largest Q-value ($Q^*$), and follows the gradient towards higher rewards:
When building blocks are violated, i.e., \( \text{SLA violated} \) if any of the 4 SLOs as defined in \( \mathcal{L} \) are violated, i.e.,

\[
\text{SLA} = \begin{cases} 
1, & \text{if any } \text{SLO}_i \text{ in } \mathcal{L} \text{ is Violated} \\
0, & \text{otherwise} 
\end{cases} 
\] (11)

Considering the information collected in the data we’re working with, the action space is defined as:

\[
a_t = \begin{cases} 
1, & \text{i.e. Scale up} \\
0, & \text{i.e. No requirement to scale up} 
\end{cases} 
\] (12)

Correspondingly, the reward function is defined as:

\[
r_t = \begin{cases} 
+20, & \text{if } a_t = 1 \text{ and } \text{SLA} = 1 \\
-10, & \text{if } a_t = 0 \text{ and } \text{SLA} = 1 \\
-5, & \text{if } a_t = 1 \text{ and } \text{SLA} = 0 \\
0, & \text{otherwise} 
\end{cases} 
\] (13)

Algorithm I details the workflow of the reinforcement learning model within the overall framework deployed, and FIGURE 4 presents its corresponding graphical overview.

FIGURE 5 provides another view at how GCRN and GRL work together within the framework, and what they deliver.

For the deep Q-learning problem formulation, we consider the SLA violated (state 1) if any of the 4 SLOs as defined in Section III are violated, i.e.,

\[
Q(s, a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdot \cdot \cdot | s_t = s, a_t = a, \pi] 
\] (9)

\( \gamma \) here is the discount factor applied to the rewards at each time step, accounting for the diminishing value of the reward at time \( t \) as we iterate forward. DRL uses deep neural networks as non-linear function approximators to estimate the action-value function and deal with a large range of outcomes and their impact over time:

\[
Q(s, a; \theta) \approx Q^*(s, a) 
\] (10)

\( \theta \) here refers to the weights of the neural network function approximator. Further, with the use of experience replay and network cloning, DRL is adept to deal with a complex environment with a big range of outcomes [59].

As stated previously, there is an active trade-off between efficiency and reliability in the existing scaling policies for latency-critical applications by service providers [4]. The aim of the reinforcement learning module here is to oversee and interface between an existing scaling policy and the defined SLA, and proactively generate warning alerts for anticipated SLA violations based on the GNN-based feature forecasting module and the defined SLOs. Building upon our existing work on granular SLA and SLO violation prediction [7], this serves as a proof-of-concept for dynamic SLA-aware policy adjustment.

For the deep Q-learning problem formulation, we consider the SLA violated (state 1) if any of the 4 SLOs as defined in Section III are violated, i.e.,

\[Q^*(s, a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdot \cdot \cdot | s_t = s, a_t = a, \pi] \] (9)

\( \gamma \) here is the discount factor applied to the rewards at each time step, accounting for the diminishing value of the reward at time \( t \) as we iterate forward. DRL uses deep neural networks as non-linear function approximators to estimate the action-value function and deal with a large range of outcomes and their impact over time:

\[Q(s, a; \theta) \approx Q^*(s, a) \] (10)

\( \theta \) here refers to the weights of the neural network function approximator. Further, with the use of experience replay and network cloning, DRL is adept to deal with a complex environment with a big range of outcomes [59].

As stated previously, there is an active trade-off between efficiency and reliability in the existing scaling policies for latency-critical applications by service providers [4]. The aim of the reinforcement learning module here is to oversee and interface between an existing scaling policy and the defined SLA, and proactively generate warning alerts for anticipated SLA violations based on the GNN-based feature forecasting module and the defined SLOs. Building upon our existing work on granular SLA and SLO violation prediction [7], this serves as a proof-of-concept for dynamic SLA-aware policy adjustment.

For the deep Q-learning problem formulation, we consider the SLA violated (state 1) if any of the 4 SLOs as defined in Section III are violated, i.e.,

\[Q^*(s, a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdot \cdot \cdot | s_t = s, a_t = a, \pi] \] (9)

\( \gamma \) here is the discount factor applied to the rewards at each time step, accounting for the diminishing value of the reward at time \( t \) as we iterate forward. DRL uses deep neural networks as non-linear function approximators to estimate the action-value function and deal with a large range of outcomes and their impact over time:

\[Q(s, a; \theta) \approx Q^*(s, a) \] (10)

\( \theta \) here refers to the weights of the neural network function approximator. Further, with the use of experience replay and network cloning, DRL is adept to deal with a complex environment with a big range of outcomes [59].

As stated previously, there is an active trade-off between efficiency and reliability in the existing scaling policies for latency-critical applications by service providers [4]. The aim of the reinforcement learning module here is to oversee and interface between an existing scaling policy and the defined SLA, and proactively generate warning alerts for anticipated SLA violations based on the GNN-based feature forecasting module and the defined SLOs. Building upon our existing work on granular SLA and SLO violation prediction [7], this serves as a proof-of-concept for dynamic SLA-aware policy adjustment.

For the deep Q-learning problem formulation, we consider the SLA violated (state 1) if any of the 4 SLOs as defined in Section III are violated, i.e.,

\[Q^*(s, a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdot \cdot \cdot | s_t = s, a_t = a, \pi] \] (9)

\( \gamma \) here is the discount factor applied to the rewards at each time step, accounting for the diminishing value of the reward at time \( t \) as we iterate forward. DRL uses deep neural networks as non-linear function approximators to estimate the action-value function and deal with a large range of outcomes and their impact over time:

\[Q(s, a; \theta) \approx Q^*(s, a) \] (10)

\( \theta \) here refers to the weights of the neural network function approximator. Further, with the use of experience replay and network cloning, DRL is adept to deal with a complex environment with a big range of outcomes [59].

As stated previously, there is an active trade-off between efficiency and reliability in the existing scaling policies for latency-critical applications by service providers [4]. The aim of the reinforcement learning module here is to oversee and interface between an existing scaling policy and the defined SLA, and proactively generate warning alerts for anticipated SLA violations based on the GNN-based feature forecasting module and the defined SLOs. Building upon our existing work on granular SLA and SLO violation prediction [7], this serves as a proof-of-concept for dynamic SLA-aware policy adjustment.

For the deep Q-learning problem formulation, we consider the SLA violated (state 1) if any of the 4 SLOs as defined in Section III are violated, i.e.,

\[Q^*(s, a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdot \cdot \cdot | s_t = s, a_t = a, \pi] \] (9)

\( \gamma \) here is the discount factor applied to the rewards at each time step, accounting for the diminishing value of the reward at time \( t \) as we iterate forward. DRL uses deep neural networks as non-linear function approximators to estimate the action-value function and deal with a large range of outcomes and their impact over time:

\[Q(s, a; \theta) \approx Q^*(s, a) \] (10)

\( \theta \) here refers to the weights of the neural network function approximator. Further, with the use of experience replay and network cloning, DRL is adept to deal with a complex environment with a big range of outcomes [59].

As stated previously, there is an active trade-off between efficiency and reliability in the existing scaling policies for latency-critical applications by service providers [4]. The aim of the reinforcement learning module here is to oversee and interface between an existing scaling policy and the defined SLA, and proactively generate warning alerts for anticipated SLA violations based on the GNN-based feature forecasting module and the defined SLOs. Building upon our existing work on granular SLA and SLO violation prediction [7], this serves as a proof-of-concept for dynamic SLA-aware policy adjustment.

For the deep Q-learning problem formulation, we consider the SLA violated (state 1) if any of the 4 SLOs as defined in Section III are violated, i.e.,

\[Q^*(s, a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdot \cdot \cdot | s_t = s, a_t = a, \pi] \] (9)

\( \gamma \) here is the discount factor applied to the rewards at each time step, accounting for the diminishing value of the reward at time \( t \) as we iterate forward. DRL uses deep neural networks as non-linear function approximators to estimate the action-value function and deal with a large range of outcomes and their impact over time:

\[Q(s, a; \theta) \approx Q^*(s, a) \] (10)

\( \theta \) here refers to the weights of the neural network function approximator. Further, with the use of experience replay and network cloning, DRL is adept to deal with a complex environment with a big range of outcomes [59].

As stated previously, there is an active trade-off between efficiency and reliability in the existing scaling policies for latency-critical applications by service providers [4]. The aim of the reinforcement learning module here is to oversee and interface between an existing scaling policy and the defined SLA, and proactively generate warning alerts for anticipated SLA violations based on the GNN-based feature forecasting module and the defined SLOs. Building upon our existing work on granular SLA and SLO violation prediction [7], this serves as a proof-of-concept for dynamic SLA-aware policy adjustment.

For the deep Q-learning problem formulation, we consider the SLA violated (state 1) if any of the 4 SLOs as defined in Section III are violated, i.e.,
V. EXPERIMENTAL SETUP
The experiments were all set up using Python (version 3.9.7) and its associated data-science libraries. We use PyTorch® and PyTorch Lightning to program all the deep learning based implementations in the paper. For the GCRN implementation, we use PyTorch Geometric Temporal, a temporal GNN extension library for PyTorch Geometric that supports the processing of spatio-temporal signals. We program a custom Gym environment that supports Clearwater data for the DRL implementation, and use Stable Baselines3 to implement deep Q-learning in PyTorch.

A. DATASET
We use a publicly hosted dataset obtained via a standard Clearwater testbed, a visualization for which is presented in FIGURE. The dataset comprises of raw telemetric data files that track 26 metrics for each of the 6 monitored VNFs that compose the Clearwater ecosystem, and includes bursts of abnormal behaviour through its integrated stress testing tools to simulate VNF congestions and QoS degradations. The data is sampled every 30 seconds, and spans an overall period of 2 months. This corresponds to 156 features overall, and 177,098 rows of raw temporal data. A brief description of the captured metrics is summarised in Table 1.

B. SYSTEM CONFIGURATION
The experiments were performed in a Docker containerized environment running atop a bare-metal Linux server with 2 NVIDIA® Tesla K20m GPUs. The Docker image runs an Ubuntu 20.04 LTS operating system, and CUDA version 10.2 for the GPUs.

C. LEARNING AND ADAPTATION
We adopt a sliding window methodology for time-series forecasting, which is suitable for rapid forecasting. The models were tested on different window sizes, and considering the short-term dynamics of the use-case, the input window size was optimally set to 4 time-steps (i.e. data from the past 2 minutes, with a 30 second sampling frequency), and the models forecast the possibilities of SLO violations over each of the next 4 time-steps (i.e. the future 2 minute horizon). Thus, we use the data for all the features over the last 2 minutes to forecast the specific SLOs that may be violated at each step over the next 2 minutes.

The available data consisting of 177,098 rows was split into training and test sets in the ratio of 80 : 20, with the training set then further split into training and validation set in same 80 : 20 ratio split. Further, given the high degree of extreme outliers in data due to the incorporated stress tests, and the widely varied scales for each of the features depending on the category of raw metric, we use a Quantile Transformer as a non-linear transformation method to transform the very skewed nature of this dataset and map it to a uniform distribution in [0, 1].

Table 2 presents the results for the performance of two GCRN model architectures on forecasting metrics. For the deep Q-learning model, the observation space is defined as a discrete snapshot of the forecasted features at time \( t \), i.e., the episode length is defined as 1. We use a multi-layer perceptron based deep-learning policy for the Q-value approximation and the target network. We train the model for 5,000,000 timesteps, set the learning rate to 0.0001, and stick to default values from original algorithm propositions in [59] and [62] for most of the hyperparameters. To elaborate, the size of the replay buffer was set to 1,000,000, the minibatch size for each gradient update was set at 32, the update coefficient \( \tau \) was set to hard update at 1, the discount factor \( \gamma \) was set to 0.99. The model collects transitions for 50,000 timesteps before learning starts, and the model is set to update every 4 steps, with 1 gradient step after each rollout. The replay buffer class was set to automatic selection, and the target network \( \hat{Q} \) was set to update every 10,000 environment steps, with the maximum value for the gradient clipping was set to 10. The initial value of the random action probability \( \epsilon \) was set to 1, while the final value of \( \epsilon \) was set to 0.05, with the exploration fraction set to 10% of the training period.

VI. RESULTS AND DISCUSSION
Table 2 presents the results for the performance of two architecture variants of the GCRN model on forecasting metrics. To set this in perspective, FIGURE 6 compares the mean absolute error (MAE) of various models that have been previously applied to the use-case data [8] with the same forecasting horizon. Our proposed GCRN-based spatio-temporal forecasting model achieves a 74.62% improvement over the

| Model | In Channels | Out Channels | Fitter Size | MSE | MAE | RMSE |
|-------|-------------|--------------|-------------|-----|-----|------|
| GCRN  | 4           | 128          | K = 3       | 0.036 | 0.132 | 0.188 |
| GCRN  | 4           | 2048         | K = 3       | 0.034 | 0.128 | 0.185 |

For training the forecasting model, we use Mean Squared Error (MSE) as the loss function to be minimized. Further, we use Adam as the optimizer for its adaptive nature, with a learning rate of \( 10^{-2} \). ReLu (Rectified Linear Unit) is used as the activation function between layers due to its computational simplicity and high optimization performance [61]. For the general layers, we use the default PyTorch initializations for weights, activation, and bias. To control overfitting as the model gains complexity, we deploy an early stopping criteria that tracks the validation MSE with a patience of 10 epochs to ensure that the training is not stopped at a local optimization minima. Model weights are restored from the best epoch at the end of training, i.e. the best performance achieved before the model began to overfit on the training set.
Residual-LSTM \cite{8} model that was proposed as the best-in-class in previous work. This quantifies the significant leap in performance when considering spatio-temporal GCRN as opposed to a general feed-forward deep learning model (Dense), and specialized RNN-based deep learning models like LSTM and its Residual variants which have been covered in detail in \cite{8}. GCRN merges CNN for graph-structured data and RNN to simultaneously identify meaningful spatial structures and dynamic patterns, and this achieves a significant leap in performance on the use-case as opposed to regular and temporal deep learning models.

Further, FIGURES 7 and 8 demonstrate good convergence of the proposed deep Q-learning model while training, with incremental improvements in learned policy. FIGURE 9 shows the results when the trained model policy was tested on a random test sample of 100 episodes, delivering a positive reward 98\% of the time.

VII. CONCLUSION, CHALLENGES, AND FUTURE WORK

In this work, we propose a GNN and DRL based framework for proactive SLA management in the use-case of Clearwater, a latency-critical NFV application. We compose a Graph Convolutional Recurrent Network (GCRN) based spatio-temporal multivariate time-series forecasting model that forecasts the evolution of system monitoring features for the Clearwater VNF over the next 4 time steps, delivering 74.62\% improved performance over the established baseline state-of-art model on the use-case. The wide leap in forecasting performance with our proposed framework quantifies the benefits of incorporating spatial metadata in use-cases that require multi-dimensional feature forecasting in high-frequency temporal data flows. Further, we leverage realistic Clearwater SLA and SLO definitions to develop a deep Q-learning based DRL model, and train it to act as an effective SLA-aware dynamic oversight of the scaling policy at play.

As methodologies and principles evolve across systems and domains, and new principles like site reliability engineering take over the legacy operational practices, machine learning has a key role to play. The next generation of systems contain requirements that move beyond traditional monitoring, now venturing into the domain of observability. Proactive problem and event management is the need of the hour to support Cloud-based infrastructure and services for next generation verticals. To this end, one of the key challenges in production systems is to tackle data drift—models
trained offline and then deployed online are effective only as long as the data distribution remains equivalent to what the model has been trained on, and the features’ behaviour lies within the sample subset of scenarios that the training set contains. To tackle this, it is important to configure retraining feedback loops that periodically expose the model to a new subset of incoming data offline, allow a readjustment of hyperparameters inline with the new observations if any, and then a redeployment of the updated model in production. While this is beyond the scope of the current manuscript, the step has been highlighted in FIGURE 1. Moreover, an enhanced ability to react to the unknown is another reason why reinforcement learning is an important layer of oversight over the supervised learning models.

In future work, we plan to set up a custom testbed deployment with the aim to better mimic production systems and test the model overheads in retraining feedback loops, and improve the reinforcement learning model for extended online sequence modeling with longer episode lengths. Retraining overheads are a key consideration in production deployments, and something that could become a potential demerit of an otherwise well-performing model. Further, while the data we worked with includes bursts of abnormal behaviour through its integrated stress testing tools to simulate VNF congestions and QoS degradations, the proposed approaches from this dissertation could also be integrated with a traffic and workload forecasting methodology for a higher degree of detail in proactive violations’ prediction, and then combined with a dynamic policy enforcement for a wider end-to-end management control loop. Moreover, since the methodology is transferable to other latency-critical verticals within the high-availability network slice, we aim to validate this on a different use-case while also collecting and incorporating external network features.

REFERENCES

[1] W. Jiang, B. Han, M. A. Habibi, and H. D. Schotten, “The Road Towards 6G: A Comprehensive Survey,” IEEE Open Journal of the Communications Society, vol. 2, pp. 334–366, 2021.
[2] R. Mijumbi, J. Serrat, J.-L. Gorrisch, N. Bouten, F. De Turck, and R. Boutaba, “Network Function Virtualization: State-of-the-Art and Research Challenges,” IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 236–262, 2016.
[3] T. Zhang, H. Qu, L. Linguaglossa, W. Cerroni, and P. Giaccone, “NFV Platforms: Taxonomy, Design Choices and Future Challenges,” IEEE Transactions on Network and Service Management, vol. 18, pp. 30–48, Mar. 2021.
[4] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, “Topology-Aware Prediction of Virtual Network Function Resource Requirements,” IEEE Transactions on Network and Service Management, vol. 14, pp. 106–120, Mar. 2017.
[5] N. Jalodia, S. Henna, and A. Davy, “Deep Reinforcement Learning for Topology-Aware VNF Resource Prediction in NFV Environments,” in 2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), (Dallas, TX, USA), pp. 1–5, IEEE, Nov. 2019.
[6] S. Cherrared, S. Imadali, E. Fabre, and G. Goessler, “LUMEN: A global fault management framework for network virtualization environments,” in 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), (Paris), pp. 1–8, IEEE, Feb. 2018.
[7] N. Jalodia, M. Taneja, and A. Davy, “A Deep Neural Network-Based Multi-Label Classifier for SLA Violation Prediction in a Latency Sensitive NFV Application,” IEEE Open Journal of the Communications Societies, vol. 2, pp. 2469–2493, 2021.
[8] N. Jalodia, M. Taneja, A. Davy, and B. Dezfouli, “A Residual LSTM based Multi-Label Classification Framework for Proactive SLA Management in a Latency Critical NFV Application Use-Case,” in 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), (Las Vegas, NV, USA), pp. 782–789, IEEE, Jan. 2022.
[9] A. Binsahaj, T. R. Sheltami, and K. Salah, “A Survey on Autonomic Provisioning and Management of QoS in SDN Networks,” IEEE Access, vol. 7, pp. 73384–73435, 2019.
[10] S. Cherrared, S. Imadali, E. Fabre, G. Gossler, and I. G. B. Yahia, “A Survey of Fault Management in Network Virtualization Environments: Challenges and Solutions,” IEEE Transactions on Network and Service Management, vol. 16, pp. 1537–1551, Dec. 2019.
[11] X. Zheng, N. Huang, S. Yin, G. Wen, and X. Zhang, “A Service Deployment Method Considering Application Reliability of Networks,” IEEE Access, vol. 9, pp. 28505–28513, 2021.
[12] J. Suomalainen, A. Juhola, S. Shahabuddin, A. Mammela, and I. Ahmad, “Machine Learning Threatens 5G Security,” IEEE Access, vol. 8, pp. 190822–190842, 2020.
[13] “Dynamic scaling for Amazon EC2 Auto Scaling - Amazon Web Services (AWS),” https://amzn.to/3yPU963, accessed May 2021.
[14] “Predictive scaling for Amazon EC2 Auto Scaling - Amazon Web Services (AWS),” https://amzn.to/3c5pQPc, accessed May 2021.
[15] “Predictive Scaling for EC2 with Machine Learning - Amazon Web Services (AWS),” https://amzn.to/3vBHlyg, accessed May 2021.
[16] “Using Predictive Autoscaling - Google Cloud.” https://bit.ly/3yV55H, accessed May 2021.
[17] C. Sun, J. Bi, Z. Zheng, and H. Hu, “SLA-NFV: an SLA-aware High Performance Framework for Network Function Virtualization,” in Proceedings of the 2016 ACM SIGCOMM Conference, (Floridaopolis Brazil), pp. 581–582, ACM, Aug. 2016.
[18] B. Yi, X. Wang, K. Li, S. k. Das, and M. Huang, “A comprehensive survey of Network Function Virtualization,” Computer Networks, vol. 133, pp. 212–262, Mar. 2018.
[19] E. Kapassa, M. Touloupou, and D. Kyriazis, “SLAs in 5G: A Complete Framework Facilitating VNF- and NS- Tailored SLAs Management,” in 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), (Krakow), pp. 469–474, IEEE, May 2018.
[20] I. G. Ben Yahia, J. Bendriss, A. Samba, and P. Dooze, “CogNitive 5G networks: Comprehensive operator use cases with machine learning for management operations,” in 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), (Paris), pp. 252–259, IEEE, Mar. 2017.
[21] J. Bendriss, I. G. Ben Yahia, P. Chemouil, and D. Zeghlache, “AI for SLA Management in Programmable Networks,” in DRCN 2017 - Design of
Reliable Communication Networks; 13th International Conference, pp. 1–8, 2017.

O. Orhan, C. F. Chiasserini, J. Mangues-Bafalluy, J. Baranda, G. Landi, B. Martini, X. Costa-Perez, C. Puligheeda, and L. Vlacareghi, “Automated Service Provisioning and Hierarchical SLA Management in 5G Systems,” IEEE Transactions on Network and Service Management, pp. 1–20, 2021.

J. Hong, S. Park, J.-H. Yoo, and J.-W.-K. Hong, “Machine Learning based SLA-Aware VNF Anomaly Detection for Virtual Network Management,” in 2020 16th International Conference on Network and Service Management (CNSM), (Izmir, Turkey), pp. 1–7, IEEE, Nov. 2020.

D. de Vleeschauwer, J. Baranda, J. Mangues-Bafalluy, F. C. Chiasserini, M. Malinverno, C. Puligheeda, L. Magoula, J. Martin-Perez, S. Barmounakis, K. Kondepu, L. Vlacareghi, X. Li, C. Papagianni, and A. Garcia-Saadveda, “Growth Data-Driven AI-Based Scaling,” in 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), (Porto, Portugal), pp. 383–388, IEEE, June 2021.

D. Lee, J.-H. Yoo, and J.-W.-K. Hong, “Deep Q-Networks based Auto-scaling for Service Function Chaining,” in 2020 16th International Conference on Network and Service Management (CNSM), (Izmir, Turkey), pp. 1–9, IEEE, Nov. 2020.

Y. Jiang, M. Kodialam, T. V. Lakshman, S. Mukherjee, and L. Tassiulas, “Fast relaying in coded networks,” in 2006 Conference on Computer Communications (INFOCOM), (Miami, FL, USA), vol. 3, pp. 1399–1408, IEEE, March 2006.

M. Ferriol-Galmés, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfarini, “Graph Neural Networks for routing optimization in Software Defined Networks,” IEEE Communications Letters, vol. 25, pp. 176–180, Jan. 2021.

J. V. B. Beyer, C. Jones, J. Petoff, and N. Murphy, Site Reliability Engineering: How Google Runs Production Systems. O’Reilly Media, Incorporated, 2016.

D. A. Gebremarian, M. Usman, and M. Qaraqe, “Applications of Artificial Intelligence and Machine Learning in the Area of SDN and NFV: A Survey,” in 2019 16th International Multi-Conference on Systems, Signals & Devices (SSD), (Istanbul, Turkey), pp. 545–549, IEEE, Mar. 2019.

J. V. B. Beyer, C. Jones, J. Petoff, and N. Murphy, Site Reliability Engineering: How Google Runs Production Systems. O’Reilly Media, Incorporated, 2016.

S. Yeom, C. Choi, S. S. Kolekar, and K. Kim, “Graph Convolutional Network based Link State Prediction,” in 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), (Tainan, Taiwan), pp. 486–490, IEEE, Sept. 2021.

Y. Cao, H. Jiang, Y. Deng, J. Wu, P. Zhou, and W. Luo, “Detecting and Mitigating DDoS Attacks in SDN Using Spatial-Temporal Graph Convolutional Network,” IEEE Transactions on Dependable and Secure Computing, pp. 1–20, 2021.

A. Rafiq, T. A. Khan, M. Afaf, and W.-C. Song, “Service Function Chaining and Traffic Steering in SDN using Graph Neural Network,” in 2020 International Conference on Information and Communication Technology Convergence (ICTC), (Jeju, Korea (South)), pp. 500–505, IEEE, Oct. 2020.

H. Wang, Y. Wu, G. Min, and W. Miao, “A Graph Neural Network-based Digital Twin for Network Slicing Management,” IEEE Transactions on Industrial Informatics, pp. 1–20, 2020.

H.-G. Kim, S. Park, D. Heo, S. Lange, H. Choi, J.-H. Yoo, and J. W.-K. Hong, “Graph Neural Network-based Virtual Network Function Deployment Prediction,” in 2020 16th International Conference on Network and Service Management (CNSM), (Izmir, Turkey), pp. 1–7, IEEE, Nov. 2020.

D. Heo, S. Lange, H.-G. Kim, and H. Choi, “Graph Neural Network based Service Function Chaining for Automatic Network Control,” arXiv:2009.05240 [cs], Sept. 2020. arXiv:2009.05240.

H. Wu, Y. Tan, Y. Zhang, and X. Jin, “Network planning with deep reinforcement learning,” in Proceedings of the 2021 ACM SIGCOMM 2021 Conference, (Virtual Event USA), pp. 258–271, ACM, Aug. 2021.

D. Yoon, S. Hong, B.-J. Lee, and K.-E. Kim, “Winning the L2(RPN) challenge: Power grid management via semi-markov afterstate actor-critic,” in International conference on learning representations, 2021.

A. Swaminathan, M. Chaba, D. K. Sharma, and U. Ghosh, “GraphNET: Graph Neural Networks for routing optimization in Software Defined Networks,” Computer Communications, vol. 178, pp. 169–182, Oct. 2021.

P. Sun, J. Lan, Z. Guo, D. Zhang, X. Chen, Y. Hu, and Z. Liu, “DeepMigration: Flow Migration for NFV with Graph-based Deep Reinforcement Learning,” in ICC 2020 – 2020 IEEE International Conference on Communications (ICC), (Dublin, Ireland), pp. 1–6, IEEE, June 2020.

P. Sun, J. Lan, J. Li, Z. Guo, Y. Hu, and T. Hu, “Efficient flow migration for NFV with Graph-aware deep reinforcement learning,” Computer Networks, vol. 183, p. 107575, Dec. 2020.

P. Sun, J. Lan, J. Li, Z. Guo, and Y. Hu, “Combining Deep Reinforcement Learning With Graph Neural Networks for Optimal VNF Placement,” IEEE Communications Letters, vol. 25, pp. 176–180, Jan. 2021.

Y. Xie, L. Huang, Y. Kong, S. Wang, S. Xu, X. Wang, and J. Ren, “Virtualized Network Function Forwarding Graph Placing in sdn and nfv-Enabled iot Networks: A Graph Neural Network Assisted Deep Reinforcement Learning Method,” IEEE Transactions on Network and Service Management, pp. 1–20, 2021.

A. Rkhami, T. A. Quang Pham, Y. Hadjadj-Aoul, A. Outtargat, and G. Rubino, “On the Use of Graph Neural Networks for Virtual Network Embedding,” in 2016 19th International Symposium on Networks, Computers and Communications (ISNCC), (Montreal, QC, Canada), pp. 1–6, IEEE, Oct. 2020.

Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, “Automatic Virtual Network Embedding: A Deep Reinforcement Learning Approach With Graph Convolutional Networks,” IEEE Journal on Selected Areas in Communications, vol. 38, pp. 1040–1057, June 2020.

B. Beyer, C. Jones, J. Petoff, and N. Murphy, Site Reliability Engineering: How Google Runs Production Systems. O’Reilly Media, Incorporated, 2016.

M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, “IP Traffic Monitoring and Analysis (NTMA): A Survey,” Computer Communications (NetSoft), (Paris, France), pp. 42–47, IEEE, June 2019.

F. Schiavoni, S. Perini, and A. Pierangeli, “Graph Neural Networks for routing optimization in Software Defined Networks,” in 2019 3rd International Conference on Science and Technology (ICST), (Istanbul, Turkey), pp. 1–6, IEEE, Sept. 2019.

M. Abbasi, A. Shahzadi, and A. Taherkordi, “Deep Learning for Network based Link State Prediction,” in 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), (Taiwan, Taiwan), pp. 348–353, IEEE, Sept. 2021.
D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529–533, Feb. 2015.

[60] B. Rozemberczki, P. Scherer, Y. He, G. Panagopoulos, A. Riedel, M. Astefanoi, O. Kiss, F. Beres, a. G. Lopez, N. Collignon, and R. Sarkar, “PyTorch geometric temporal: Spatiotemporal signal processing with neural machine learning models,” in Proceedings of the 30th ACM international conference on information and knowledge management, pp. 4564–4573, 2021.

[61] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[62] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” arXiv:1312.5602 [cs], Dec. 2013. arXiv: 1312.5602.

NIKITA JALODIA recently completed her Ph.D. with the Department of Computing and Mathematics at the Emerging Networks Lab Research Division in Walton Institute of Information and Communication Systems Science, South East Technological University, Ireland. She joined in July 2017, and has since been working as a part of the Science Foundation Ireland funded CONNECT Research Centre for Future Networks and Communications. Her current research interests include Data Science, Network Function Virtualization (NFV), Machine and Deep Learning, Knowledge-Defined Networks, Internet of Things (IoT), and Fog and Cloud Computing. She received her Bachelor’s Degree in Computer Science and Engineering from The LNM Institute of Information Technology, Jaipur, India in 2017, with a specialization in Big Data and Analytics with IBM. She has also previously worked as a Software Developer at Publicis (Sapient) Global Markets, India.

MOHIT TANEJA is a Tenured Lecturer in Business Information Systems, Data Science and Computing with the Department of Accountancy and Economics, School of Business at South East Technological University. Previously, he was a Postdoctoral Research Fellow with the Programmable Autonomous Systems Division of Walton Institute; working on EU, National, International, and Industry funded projects. He worked as the Project and Tech Lead on EU-H2020 funded Smart Cities 2030 project. In Walton, he also worked within the Strategic Division at Walton, liaising with all the Research Divisions within the Institute. He has also been associated with the SFI funded VistaMilk Research Centre and CONNECT Centre. He has previously worked as an Experienced Software Research Engineer with the Emerging Networks Lab (ENL) division. He earned his Ph.D. from South East Technological University, Ireland in 2020. The title of his dissertation was ‘Fog Computing Support for Internet of Things Applications’. He was also a visiting research fellow at IBM Research Labs, Ireland from 2017-2018. His current research interests include Fog and Cloud Computing, IoT, Distributed Systems, and Distributed Data Analytics. He received his Bachelor’s Degree in Computer Science and Engineering from The LNM Institute of Information Technology, Jaipur, India in 2015.

ALAN DAVY is currently the Head of the Department of Computing and Mathematics, School of Science and Computing, South East Technological University (SETU), Waterford, Ireland. Previously, he was the Research Division Manager of the Emerging Networks Laboratory in WIT’s Walton Institute of Information and Communication Systems Science. He has been the Coordinator and Principal Investigator on a number of National, International, and EU projects such as TERAPOD, SFI-TIDA, 5GinFIRE’s C2G-RAN, etc. He was awarded B.Sc. (with Hons.) in Applied Computing and Ph.D. degrees from the South East Technological University, Waterford, Ireland, in 2002 and 2008 respectively. He has been a recipient of the Marie Curie International Mobility Fellowship in 2010, and has worked at the Universitat Politècnica de Catalunya, and IIT Madras in India. His current research interests include Virtualised Telecom Networks, Fixed and Wireless Network Management, Software Defined Infrastructure, Internet of Things, Fog and Edge Computing, Bio-inspired Systems, Molecular/Nano-scale Communications, and TeraHertz Communications.