AI-Driven Data Analytics and Intent-Based Networking for Orchestration and Control of B5G Consumer Electronics Services

Khizar Abbas, Ali Nauman, Muhammad Bilal, *Senior Member, IEEE,* Jae-Hyung Yoo, James Won-Ki Hong, *Senior Member, IEEE,* and Wang-Cheol Song*

***Abstract*—Network slicing is a critical feature of the beyond fifth-generation (B5G) network that supports a wide range of innovative services from 5.0 industries, next-generation consumer electronics, smart healthcare, etc. Network slicing guarantees the provisioning of quality of service (QoS) aware dedicated resources to each service. However, the orchestration and management of network slicing is very challenging because of the complex configuration process for underlying network resources. Furthermore, the third generation partnership project (3GPP) presented artificial intelligence (AI) based network data analytics function (NWDAF) in 5G for proactive management and intelligence. Therefore, we have developed an intent-based networking (IBN) system for automating network slices and an AI-driven NWDAF for proactive and intelligent resource assurance. The network data analytics function uses a hybrid stacking ensemble learning (STEL) algorithm to predict network resource utilization and a novel automated machine learning (AutoML) and voting ensemble learning-based mechanism to detect and mitigate network anomalies. To validate the performance of the implemented work, real-time datasets were employed, and a comparative analysis was conducted. The experimental result shows that our STEL model enhances the accuracy by 20% and reduces the error rate by 45%. The AutoML and ensemble learning-based optimized model achieved 99.22% accuracy for anomaly detection.**

***Index Terms*—Beyond 5G networks, Next-Generation Consumer Electronics Services, 5G Slicing, Intent-based Networking, AI, Automated Machine Learning, Network Data Analytics Function**

I. Introduction

The traditional networks do not support a wide variety of novel next-generation consumer electronics services, Web 3.0, and industry 5.0 services and ensure just limited services in terms of messaging, voice, and internet access. These novel services have different quality of services (QoS) requirements regarding bandwidth, latency, mobility, reliability, and capacity. Additionally, the consumer electronics industry is growing rapidly with the continuous increase in smart electronic devices. Due to that, many novel consumer electronics applications (gaming and entertainment) have emerged. These smart consumer electronics services have strict QoS requirements, such as ultra-low latency and seamless connectivity [1]. So, it is very challenging for the traditional network to accommodate these diverse consumer electronics services. On the other side, the primary vision of the 5G and beyond networks are to fulfill the diverse service requirements of different consumers, mobile network operators (MNOs), industrial verticals, consumer electronics verticals, and businesses [2]. The future generation mobile networks should accommodate various industrial verticals, including energy, automotive, healthcare, media, entertainment, consumer electronics, and manufacturing. These industrial use cases have diverging QoS requirements, and service-oriented architecture is needed to handle these various use cases. However, the 5G is designed on a service-based pattern that efficiently entertains multi-services with desired bandwidth, reliability, and latency [3], [4].

In recent years, network function virtualization (NFV) and software-defined networking (SDN) have emerged as cutting-edge technologies for constructing virtualized, softwarized, cloudified, highly programmable, and flexible 5G mobile networks [5]. SDN decouples the control plane from the data/user plane. The network functions (NFs) in the control plane run as independent applications under the centralized network controllers. On the other side, NFV enables MNOs to deploy their NFs in a Whitebox or generic hardware instead of expansive dedicated hardware devices. So, NOs can quickly deploy their various virtual network functions (VNFs) over general-purpose servers. Also, mobile edge computing (MEC) emerges as a key technology to overcome the low latency issue, where the storage, computation, and network resources move from the central cloud to the edge. It is also q critical technology of the 5G network to ensure ultra-low latency communication for latency-oriented applications. However, network slicing is the best candidate for MNOs to provide differentiated QoS requirements to each user category [6], [7].

Manuscript received July 10, 2012. This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2021-2017-0-01633) supervised by the IITP(Institute for Information communications Technology Planning Evaluation). This work was partly supported by the Institute of Information Communications Technology Planning Evaluation (IITP) grant funded by the Korean Government (MSIT) (2018-0-00749), Development of Virtual Network Management Technology based on Artificial Intelligence)

Khizar Abbas, Jae-Hyoung Yoo, and James Won-Ki Hong are with the Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang 37673, South Korea (e-mail: engr.khizarabbas14@gmail.com, jhyoo78@postech.ac.kr, jwkhong@postech.ac.kr).

Ali Nauman is with the Department of Information and Communication Engineering Yeungnam University, South Korea (e-mail: anauman@ynu.ac.kr).

Muhammad Bilal is with the Computer Engineering, Hankuk University of Foreign Studies, Yongin-si, Gyeonggido, 17035, South Korea. (e-mail: m.bilal@ieee.org).

Wang-Cheol Song is with the Department of Computer Engineering, Jeju National University, Jeju, South Korea (e-mail: philo@jejunu.ac.kr).

Network slicing is partitioning the physical network into several virtual networks. 5G slicing is possible due to the SDN and NFV computing technologies. Each of the logically isolated networks serves a specific group of consumers. It also enables the MNOs to share their infrastructure among other verticals to ensure service guarantees [8], [9]. So, slicing the network is the best choice to accommodate the distinct tailored service requirements of different businesses, industrial verticals, next-generation consumer electronics service providers, and consumers over the same physical infrastructure. It facilitates infrastructure providers to use generic hardware devices for implementing multiple VNFs for a slice rather than legacy hardware devices. So, multiple VNFs are chained together flexibly to establish an end-to-end (e2e) network slice for a specific user group. The fully cloud-native nature of network slicing makes the 5G networks highly flexible and programmable, supporting various services efficiently [10], [11].

The International Telecommunication Union (ITU) and 3GPP have divided these diverse services into ultra-reliable low latency (URLLC), enhanced mobile broadband (eMBB) service, and massive machine-type communication (mMTC). The eMBB slice category contains ultra-high definition (UHD) communication services, e.g., video streaming. Also, the URLLC slice type includes low-latency applications, autonomous driving, and healthcare industry communication. Besides, the mMTC contains smart agriculture, smart factory, metering, billing, and logistics services type communication [12]. So, the automatic management and orchestration of these innovative services is required.

Moreover, the Internet Engineering Task Force (IETF) introduced an innovative technology, namely Intent-based Networking (IBN), for automating the network [13]. IBN is an intelligent system that enables the future network to be self-configured, self-planned, self-assured, and self-healing. It provides closed-loop assurance for handling multi-domain network resources. Conversely, In the 5G control plane, 3GPP has also introduced an innovative NWDAF function to provide intelligence to the network [14]. It collects the data from control and user plane NFs: user plane function (UPF), access and mobility management function (AMF), network slice selection function (NSSF), etc., and performs analytics by using various machine learning (ML) techniques. These ML algorithms are trained on historical network data and perform recommendation, prediction, and detection tasks [15]–[17]. So, AI and ML technologies are very popular for the proactive automation and management of cloud resources. Due to this, NWDAF is the best choice for performing proactive management and assurance of multi-domain resources.

A. Research Motivation

The primary aim of the 5G network is to accommodate a wide variety of innovative services with differentiated QoS requirements. So, it is a highly error-prone and time-consuming process to generate manual configurations for every service. It also needed adequate human intervention, manual work, and expertise. Besides, the manual resource allocation in multidomain infrastructure for establishing an e2e network slice is not optimal. The automatic deployment of network resources over the RAN and core domain is still challenging because each domain requires specific configurations. So, a well-designed solution is required to generate multi-domain network configurations for activating e2e slices as per QoS requirements. Conversely, AI and ML approaches are needed to manage network resources proactively. In this aspect, dynamic scaling of the cloud resources is another critical issue while managing the cloud resources. It causes degradation of the QoS in resource overloading and wastes the cloud resources in underloading cases. Hence, autoscaling cloud resources is vital for reducing costs and guaranteeing the customers' QoS requirements. So, an AI and ML-based accurate estimation of future network resource utilization to perform autoscaling is needed. Predicting accurate network usage can enhance network efficiency and reduce operational costs. The prediction results can also be used to consolidate resources, proactive management, host mode detection, and QoS assurance. Moreover, ensuring network security is a vital consideration for the future of mobile networks. Nowadays, AI approaches seem very promising to enhance network security. So, a computationally efficient ML-based mechanism is needed to detect network anomalies accurately.

B. Research Contributions

To overcome the challenges associated with the above-mentioned research problems, these are the major objectives and contributions of this research:

- An IBN system to automate the slice configuration generation process and orchestrate the slice resources over the multi-domain infrastructure. IBN eliminates manual policy generation and automatically performs instantiation, activation, and deletion of slice resources.
- IBN follows a one-touch approach where users need to insert abstract QoS requirements from the GUI portal, and the system converts these service QoS into slice templates or policies. After that, generated slice configurations are implemented with the Open Source Mano (OSM) and a RAN controller.
- AI techniques are an important aspect of the IBN platform for proactive updates and assurance. So, we have developed an NWDAF-based mechanism for IBN to enable network intelligence.
- A hybrid stacking ensemble learning (STEL) model has been implemented inside the NWDAF for network resource workload prediction. These results can be used for autoscaling resources, QoS assurance, detecting overloaded and underloaded hosts, and improving resource usage efficiency.
- A novel autoML and voting ensemble learning-based optimized method has been implemented to detect and mitigate network anomalies and ensure network security.
- The NWDAF enhances the intelligence of the IBN mechanism by enabling proactive management and updates

of network slice resources, as well as detecting and mitigating network anomalies.

C. Paper Structure

The rest of the paper is structured as follows. Related literature about network slicing automation and management, IBN, NWDAF, AI approaches for resource usage forecasting, and network attack detection and mitigation is presented in Section II. Section III presents the details of our system with its components, such as the IBN framework for network slicing and NWDAF for AI intelligence. Section IV explains the results and performance analysis of the implemented system. The concluding remarks on the implemented mechanism have been summarized in Section V.

II. Related Work

This section explains the detail of previous works related to network slicing, 5G networks, orchestration and management of novel consumer electronics and B5G services, slice LCM, and AI /ML approaches for the automation of future networks. It also includes industrial solutions for automating the networks.

A. E2E Network Slice Orchestration and Management

To provide differentiated services to 5G consumers, the management and automation of network slicing is an essential activity for MNOs. Several mobile network standardization bodies have defined the specifications for e2e network slice automation and management, such as 3GPP, IETF, European Telecommunication Standards Institute (ETSI), ITU, Next Generation Mobile Network (NGMN), Fifth Generation Partnership Project (5GPP), etc. As aforementioned, network slicing is divided into three major types: eMBB, URLLC, and mMTC. The eMBB type of service needs reliable broadband connectivity and high speed. On the other side, URLLC slice requires low-latency communication, and mMTC requires seamless connectivity for many devices and smart industries [12], [18], [19].

Several open-source orchestration platforms were implemented in the literature. These orchestration platform supports network slicing and automation. Some well-known orchestration platforms are Tacker, COMEC, open network automation platform (ONAP), JOX, M-CORD, SONATA, OPNFV, 5G NORMA, Cloudify, OpenBaton, OpenStack HEAT, and Open-O [20]–[26]. The primary aim of these platforms is to enable programmability to automate the network resources deployment over the infrastructure. The network administrators define the policy configurations for the deployment of the resources. For example, the OpenStack platform is used to deploy the VNFs and SDN-based controllers are used for chaining the VNFs and handling transport network operations.

The 3GPP has also introduced an architecture for the management of the 5G networks that includes communication service management function (CSMF), network slice management function (NSMF), and network slice subnet management function (NSSMF) [18]. The CSMF acts as a central management entity for deploying and managing e2e slices. It creates the network slice requests and sends them to NSMF for further operations. The MNOs used CSMF functions to plan, design, and activate the network slices. Conversely, NSMF translates the slice requirements and generates domain-specific configurations. Further, those configurations are forwarded to NSSMF to deploy the network slice instances. Each domain has separate NSMF, e.g., RAN, core, edge. Also, NSSMF ultimately manages each slice instance. Moreover, the CSMF receives network slice requests with QoS requirements from the customers and forwards those requests to NSMF. NSMF converts slice QoS into policies and activates the resources with the cooperation of NSSMF. The slice instances are appropriately monitored, and CSMF performs autoscaling of the network resources whenever needed [27].

Meneses et al. [28] developed a slice automation system to instantiate e2e network slices. It can deploy multiple network functions physically and virtually. This plugin-based system, SliMANO, interacts among network orchestration entities, such as orchestrators and controllers, for an e2e slice activation. These entities can be NFVOs, SDN controllers, and RAN controllers. They have compared their system with OSM's recent network slicing feature (NetSlice) by creating and deleting e2e slices. It achieves less delay for slice creation and deletion. This SliMANO allows the interaction of multiple SDN and RAN controllers. Li et al. [29] discussed network slicing in transport, core, and RAN domains. The presented work used a separate slice controller or orchestrator for each domain. Experimental results were conducted to prove the benefits of network slicing in RAN by using a two-level radio resource allocation method. They achieved the performance of e2e network slicing with better granularity, slice creation, deletion, and scheme adjustment.

However, existing orchestration platforms require manual, domain-specific, and complex network configurations to activate resources. These platforms also require expertise and human involvement, which is highly error-prone. In addition, configuring and activating multidomain resources, including core VNFs, RAN, and transport network resources, are very complex because each domain requires configurations in a specific format such as JSON template, YAML template, etc. Besides, the standardized bodies are working towards zero-touch network orchestration and management, which automatically performs resource configuration and activation without human interactions. So, our IBN system with AI-driven network data analytics architecture is a step towards zero-touch automation, which automatically performs orchestration and management of core VNFs and RAN resources in a network slicing context.

B. Intent-Based Networking

Another standardization body IETF has introduced the specifications for automating the network through IBN [13]. IBN is an intelligent system that guarantees self-configuration, self-assurance, self-planning, and self-healing capabilities for the future network. Many industrial organizations such as Cisco [30], Huawei [31], APSTRA [32], etc., also adopted

IBN technology for the orchestration of 5G networks. IBN works in four phases. Firstly, users need to input business intents through the system's dashboard; Secondly, received service requirements are translated into policies through the translation engine. Thirdly, translated policies are deployed over the physical and virtual infrastructure. Finally, deployed services are appropriately monitored to ensure the services and updates in case of failure. According to the IETRF [13] IBN for closed-loop automation, a proper IBN system (IBNS) contains two fundamental qualities that make it more than just a fancy configuration management platform: intent fulfillment and intent assurance. IBNS can recognize and generate user intents, translate them into policies, refine from NOs to validate intent, configure the resources, monitor the deployed resources, analyze network status, validate the QoS, and report to the operator for service assurance. Another well-known industrial organization Apstra [32] have also developed a solution for automating the network. This solution can be achieved through the following three technologies: Firstly, they have developed an IBN platform for service designing, translation of policies, self-reporting, and validation. Secondly, Single Source of Truth (SSOT), closed-loop network automation, and data analytics mechanisms are achieved through a graph datastore. Thirdly, they provide complete openness and vendor independence. They have an IBN analytics mechanism that allows insight into the complete infra status and collects and stores more important data. Apstra has an Apstra operating system (AOS) that allows real-time designing, building, deploying, and validating networks.

C. AI and ML approaches for Network Resource Utilization Prediction

Many studies have used AI and ML approaches in multiple areas for proactive automation and management of cloud resources: AI-assisted network data analytics, network resource usage prediction, anomaly detection, mitigation, radio resource management, QoS assurance, Proactive management, and mobility prediction [15].

Sevgican et al. [33] have developed AI-based NWDAF with 3GPP standard specifications. A synthetic dataset based on cell-level information was generated for 5G, incorporating anomalies. Network load prediction was conducted using recursive neural network (RNN) based long short-term memory (LSTM) and linear regression (LR) methods. Additionally, extreme gradient boosting (XGBoost) and logistic regression models were employed for anomaly classification. The simulation results showed that neural network-based algorithms outperformed LR for resource load forecasting. However, the XGBoost algorithm outperformed logistic regression while detecting network anomalies. Therefore, these estimations can enhance the 5G network performance through NWDAF.

Ouhame et al. [34] developed a hybrid LSTM and convolution neural network (CNN) based method to forecast multivariate resource usage: memory, CPU, and throughput utilization. Initially, the vector auto-regression technique filtered linear interdependencies from the dataset. After the LSTM stage, the residual data is computed and inputted into a CNN, which captures intricate attributes of each virtual machine utilization component. The implemented model utilizes an activation function based on a scaled polynomial constant and is compared against alternative predictive models. It improved the accuracy performance up to 3.8% to 10.9% and reduced error to approximately 7% to 8.5% than existing methods.

Abdullah et al. [35] implemented the support vector regression (SVR) algorithm to predict multi-attribute resource utilization. Inside SVR, radial basis kernel function and sequential minimal optimization algorithm (SMOA) were hypertuned to improve the forecasting accuracy. They used real datasets from Google Cluster Workload Traces (GCWT), BitBrain (BB), and PlanetLab (PL) to validate their results. They improved the accuracy to 4%-16%, with the reduction in error percentage at around 8%-60%.

Iqbal et al. [36] implemented a novel method that identifies the most suitable method adaptively and automatically to predict future resource usage. Multiple ML models have been trained on a historical dataset. An ML classifier was used to select the best prediction method for a current time slot. Besides, the adaptive model was evaluated using real-time data and compared to multiple baseline methods. The results indicated that the adaptive method outperforms existing approaches by improving 6% to 27% prediction accuracy. Moreover, an EL-based hybrid model was developed for CPU usage prediction in this work [37]. The hybrid model is a combination of multiple lightGBM models. It shows a maximum $R^2 = 0.91$ accuracy while performing CPU prediction.

Nowadays, AI has been widely used in 5G networks, but there is still a lack of standard solutions to build an operational system. Bega et al. [15] proposed the NWDAF mechanism using AI methods for long-term and short-term future forecasting. Their proposed framework used various AI techniques to develop a fully operational system. The domain orchestrators use the results from implemented AI methods for controlling and managing the resources.

The constant changes in cloud resource usage affect the accuracy of forecasting algorithms, which is challenging. Therefore, RNN was applied for predicting CPU utilization in terms of single time-step and multiple time-steps by Duggan et al. [38]. It predicts more accurately than traditional approaches for time series problems. Islam et al. [39] developed ML and slide window-based forecasting models to scale cloud resources proactively. Historical and current resource utilization data were used to forecast future utilization. It shows 80% prediction accuracy. Kumar et al. [40] used a neural network and self-adaptive differential evolution to predict future resource usage. This proposed approach provided better accuracy than the standard back-propagation algorithm concerning RMSE. Gupta et al. [41] have implemented multivariate and bidirectional LSTM models for resource usage forecasting. The generated results of both models were compared with different fractional-based techniques and outperformed existing models based on Google cluster trace.

D. ML-based Approaches for Network Attack Detection

This section explains the ML-based methods for network attack detection. In this work [42], authors developed a hybrid EL-based model that includes the support vector machines (SVM), Kalman Filter, and fuzzy K-means for network anomaly detection. The proposed method was evaluated using DARPA 1998 and KDD 1999 datasets. This method shows more than 92% of F-score, precision, and accuracy. This work [43] used a Hidden Markov model for network attack detection. Through their findings, it was discovered that the Hidden Markov model exhibits superior performance compared to other anomaly detection techniques, as evidenced by lower false-positive rates and higher average anomaly detection rates.

Recently, various ML algorithms have been developed for detecting malicious traffic in SDN-enabled networks. This research paper focuses on detecting attack traffic in vehicular networks, leveraging the centralized control aspect of SDN [44]. The study proposes a combination of SVM with kernel principal component analysis (KPCA) for dimension reduction of feature vectors, complemented by a genetic algorithm (GA) to optimize various SVM parameters, ultimately enhancing detection accuracy. Additionally, an improved kernel function (N-RBF) is employed to mitigate the impact of feature differences-induced noise. The proposed model demonstrates superior classification accuracy and generalization compared to standalone SVM models. Furthermore, it can be integrated within the controller to establish security rules and preempt potential attacks by malicious entities.

This paper [45] presented a deep neural network-based DDoS attack detection method named Secure5G for the network slicing environment. This method detects the attacked UE connection and stops it from entering into 5G core VNFs. The developed mechanism achieved above 98% accuracy for attack detection. In this work [46], Authors have developed an LSTM-based method to detect network anomalies named as DeepSecure. It achieved a very good 99.97% accuracy for detecting the attack on the CICDDoS 2019 dataset. Danish Sattar and Ashraf Matrawy [47] have developed a mathematical slice isolation mechanism to mitigate DDoS attacks. This method enhances security and provides secure communication to 5G slicing users. This approach also increases the slice availability to the 5G users. So, various ML and DL-based solutions have been developed for attack detection from the networks, but most of the DL methods are computationally inefficient and take more training time than ML methods. However, we have developed an optimized and novel AutoML and EL-based mechanism for the detection of network anomalies for securing 5G slicing and future networks.

III. Proposed IBN and AI-driven Data Analytics System

The architecture of the proposed IBN and NWDAF mechanism for e2e slice automation and management is presented in Figure 1. It includes the IBN framework, NWDAF, OSM NFV orchestrator, FlexRAN controller, and monitoring method. The details of each component for automation and management of network slicing are explained below.

A. IBN Framework for Network Slice Automation and Management

The IBN system contains a web portal, e2e design and information repository, intent translation mechanism, network policy generation, and NWDAF. The IBN system provides closed-loop capabilities for managing and automating multi-domain resources. It automatically instantiates, activates, monitors, and deletes network slices. The higher-level service requirements are inserted from the web portal of IBN in the user intent form. The intent translation module converts inputted intentions into a specific slice resource requirements format. The e2e design and information repository is an IBN database that stores the information of underlying core and RAN resources, installed policies, APIs, and e2e network topology for slice connection.

The IBN translation module translates the user intents into resource requirements and prepares a VNF forwarding graph (VNFFG) using underlying resource database information. After that, the IBN manager forwards VNFFG to the network policy generation module to prepare the slice template for each domain. The underlying controller and network orchestrator accept the configurations in different formats, such as M-CORD in TOSCA format, OSM in JSON string, and FlexRAN controller in JSON template. Due to that, our policy generation module contains three policy generators for the core, edge, and RAN domains. Each domain policy generator translates the resource requirements into a domain-specific slice template and forwards them to the underlying orchestrator and RAN controller to deploy slice resources.

In our system, we have used the OSM orchestrator to deploy the slice VNFs according to the received slice template over the core domain. ETSI MANO-based OSM is a leading NFVO for VNF deployment, offering NFVO, VNFM, and VIM functionalities for network automation and management. OSM used integrated OpenStack as a virtual infrastructure manager (VIM) for deploying core VNFs. OSM accepts policies as JSON strings—Network Service Descriptor (NSD), Virtual Network Function Descriptor (VNFD), and Network Slice Template (NST)—via REST interfaces to facilitate resource deployment. The NSD outlines VNF connections for service provisioning, while VNFD contains networking, interfacing, and resource details [23]. The IBN mechanism utilizes REST-API to convey the prepared slice template in JSON strings to OSM, enabling the deployment of core network instances.

Conversely, to manage network slicing in the RAN domain, we employed FlexRAN, an SDN-based RAN controller known for its adaptability and versatility in accommodating various RAN operations [48]. Its support for dynamic RAN slicing makes it particularly adept. In our approach, IBN utilizes a REST interface to transmit the JSON-formatted slice template, crafted by the RAN slice template generator module, to FlexRAN. The RAN slice template includes the information of the dedicated core network VNFs for creating

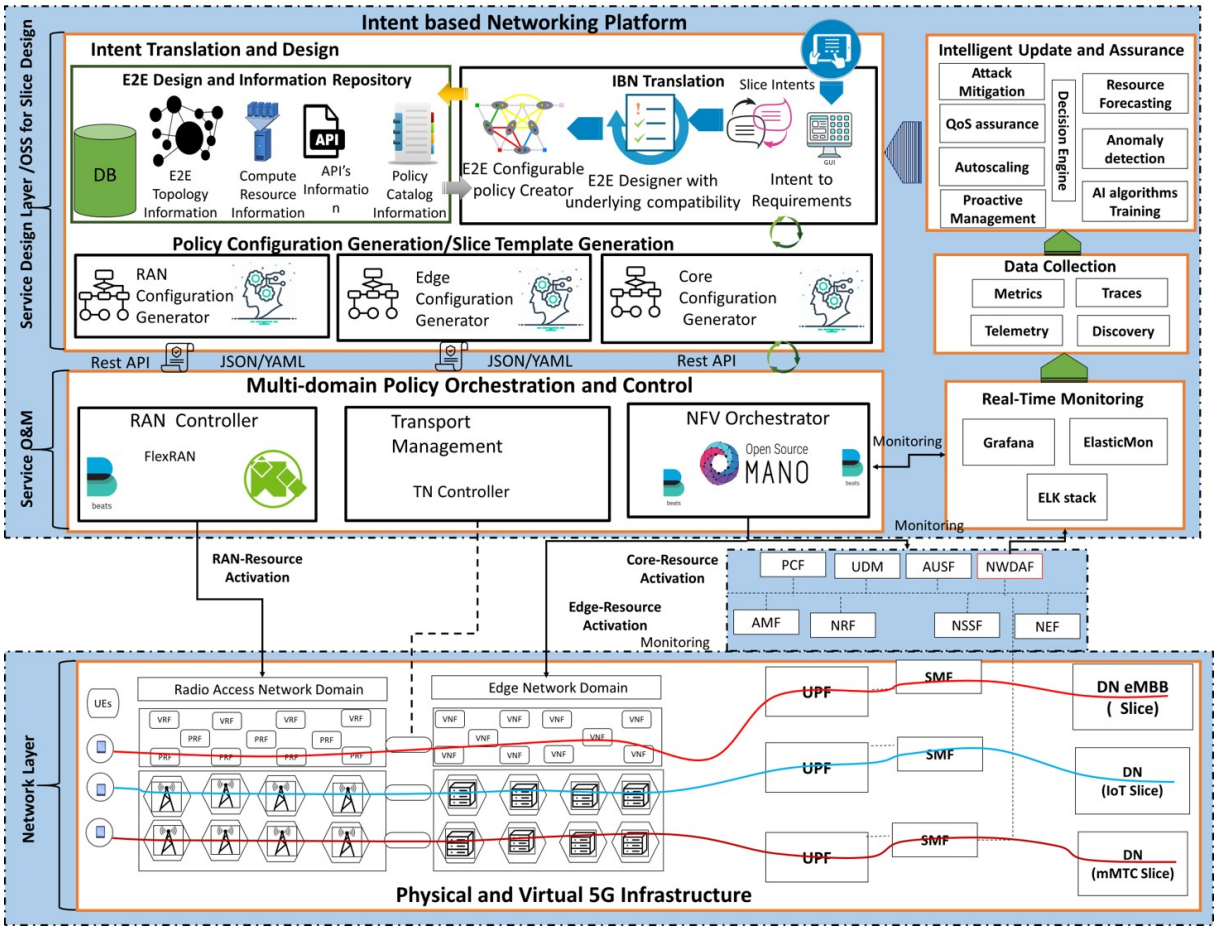

Fig. 1: Architecture of IBN and NWDAF for e2e network slice automation and management

an e2e connection. So FlexRAN facilitates the slicing of RAN resources according to Quality of Service (QoS) parameters. After activating the slice, the slice core and RAN resources are continuously monitored through Grafana, Prometheus, and ElasticMon-based monitoring mechanisms. It also collects and stores the data logs from core VNFs for performing future VNF load prediction.

B. IBN-based Network Data Analytics Mechanism for Proactive update and assurance

The ML-NWDAF is a novel feature of the IBN framework that provides intelligence for proactive updates and assurance of network resources. It assists the NOs in managing the resources and reduces QoS degradation by future prediction. NWDAF has pre-trained ML models on historical network data from the core network. We implemented two ML methods for resource (VNF) usage forecasting and network anomaly detection. NWDAF provides the outcomes to the IBN platform to achieve specific use cases such as resource utilization prediction, attack detection and prevention, mobility prediction, load balancing, etc.

Figure 2 illustrates the internal NWDAF ML pipeline mechanism with the IBN platform, where data is collected from the underlying data plane's different source (SRC) nodes. Using

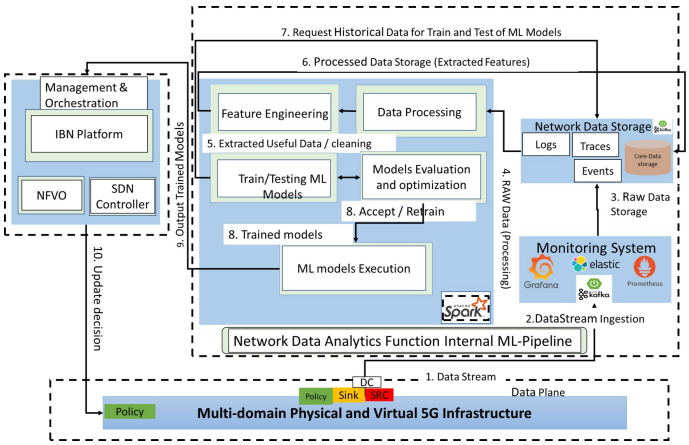

Fig. 2: NWDAF internal workflow and ML-pipeline for ML training and execution

the KAFKA framework, we deployed a data exporter node inside each domain for data collection (DC) purposes. The collected raw data has been stored separately in real-time NWDAF data storage. After that, raw data is preprocessed through cleaning, transformation, and feature extraction ML

operations. The extracted preprocessed data is stored in the data storage module. We have used the Spark big data analytics framework for data analytics and storage. The stored preprocessed data is further used for training ML models. Once the model has been trained, it is evaluated based on performance measures such as accuracy and mean square error (MSE). If the model provides satisfying accuracy, it will be deployed for execution. The results of various executed ML models have been forwarded to the IBN decision engine, which triggers update policies such as scale-up, scale-down, or DDoS attack detection and prevention policy. After that, the issued policy will be deployed over the sink nodes through domain controllers and NFVO. Our mechanism uses a stacking EL-based model inside the C-DAF module to predict core VNF resource utilization or resource forecasts and an AutoML and EL-based optimized model for anomaly detection. The other module decision engine of the IBN platform has a dynamic thresholding-based model that uses the NWDAF module's prediction results and performs the auto-scaling of the resources in case of host overloading and underloading. These VNF resource forecasting results will also be used for VNF consolidation and load balancing.

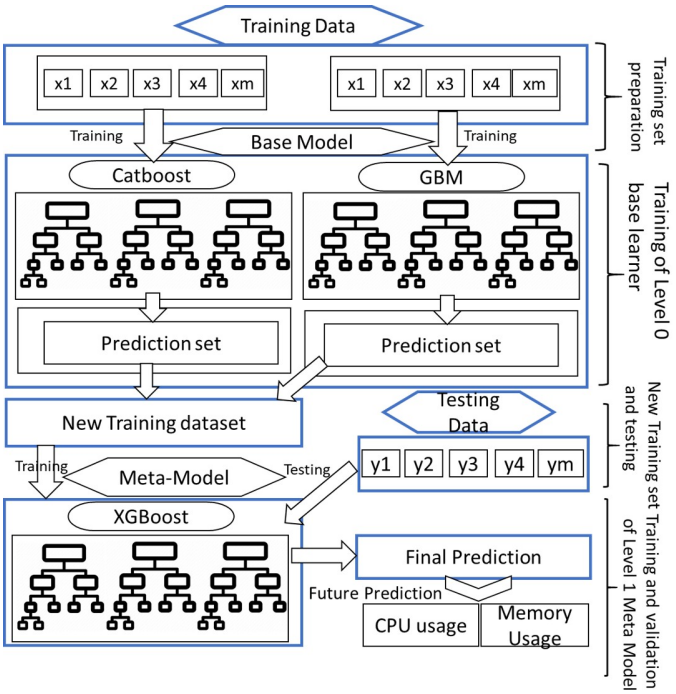

Fig. 3: Implemented mechanism of stacking ensemble-learning based method for multi-attribute resource workload prediction

1) Stacking Ensemble-Learning (STEL) model for VNF workload Prediction: Ensemble Learning (EL) is an ML technique that combines different models to produce optimized and improved accuracy. It is divided into three categories for implementation: bagging, boosting, and stacking. In this work, we have used stacking EL, also referred to as the stacked generalization technique, to improve and optimize network resource workload prediction. The stacking EL approach's prediction results are typically superior to that of individual models. In the stacking approach, several classifiers or regressor models are integrated using a meta-regressor or meta-classifier. The meta-regressor model is trained using the output from the base learner models (level-0). The meta-regressor model learns from the base learners' experience [49], [50] to reduce errors in the final prediction result.

Additionally, we used Extreme Gradient Boosting (XGBoost) as a meta-learner (Level-1) model and Gradient Boosting Machine (GBM) and Catboost as base-learner (Level-0) models. The workflow process of the STEL model is depicted in Figure 3. The implementation of considered ML models for network resource workload prediction are explained below:

a) Gradient Boosting Machine (GBM): Friedman invented GBM in 2002, a very effective ML method that precisely performs regression and classification tasks [51]. The primary idea behind GBM is to reduce the loss function by including an additional weak learner model to compensate for the weaknesses of the existing weak learner models. It minimizes bias and variance by adding base learner models and focusing on the incorrectly classified data. GBM uses several regression trees (base learner models) and combines their output to perform the final prediction. The boosting approximation function is explained in equation (1) where $h(t; b_k)$ is a function for base learner model, B_k is expansion coefficients, t represents explanatory variables, and b_k illustrates the model parameters.

$$F_(t_i) = \sum_{k=1}^{N} B_k h(t; b_k) \tag{1}$$

The following hyperparameters were used to implement the GBM model: 1000 trees, 0.1 learning rate, evaluation criteria is Friedman MSE, 1.0 subsample, and 0.1 validation fraction. So, various recently developed boosting algorithms such as Catboost, LightGBM, and XGBoost used GBM as a base algorithm to boost scalability.

b) XGBoost: Tianqi Chen and Carlos Guestrin developed the supervised learning technique XGBoost (eXtreme Gradient Boosting) in 2014 [52]. It operates on the boosting concept, which allows for creating an active learner from weak learners. It is a function approximation and regularization-based tree integration algorithm that effectively implements the GBM algorithm. It is a scalable boosting method and more than ten times faster than other boosting techniques. The most crucial factor in the XGBoost algorithm is parallel and distributed computation ability and innovative tree learning model for handling spare data. The sample forecast in the model used by XGBoost is the cumulative sum of the predicted values for a sample in each tree. Moreover, it reduced the overfitting problem due to the addition of regularization terms and loss functions optimization. In our scenario, dataset D is given with n features $D = [(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), \ldots, (X_n, Y_n)]$ to the model which has K number of trees for the target prediction $\hat{Y}_i$ as illustrated in equation (1). Also, $F_X = W_{Q(X)}, (W \in R^T)$ is Classification and regression Tree (CART) space, Q denotes

structure of each tree, T represents number of leaf nodes in the tree, W is leaf weights, and F_K presents K_{th} tree.

$$\hat{Y}_i = \sum_{k=1}^{K} F_k\left(X_i\right), F_k \in F \tag{2}$$

Equation (2) explains the objective function, which has two parts loss function regularization terms, where Y_i is true label values, $\hat{Y}_i$ is the predicted values,$\sum_{i}^{I} L(Y_i, \hat{Y}_i)$ is the loss per sample and loss function L can be customized in several ways. The main goal is to minimize the objective function. On the other side, $\Omega\left(F_k\right) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} W_j^2$ is regularization term where W is j_{th} tree leaf node weight, γ is regular leaf tree penalty term, and λ is regular leaf weight penalty term. These both penalty terms act as a smoothing factor and prevent the overfitting problem.

$$Obj\left(\theta\right) = \sum_{i}^{I} L(Y_i, \hat{Y}_i) + \sum_{k}^{K} \Omega\left(F_k\right) \tag{3}$$

$$Obj\left(t\right) = \sum_{i}^{n} L(Y_i, \hat{Y}_i^{(t-1)} + F_t(X_i)) + \Omega\left(F_t\right) \tag{4}$$

Equation (3) illustrates the objective function for t_{th} tree, whereas $t-1$ is the sum of the predicted values of the $t-1$ tree, $F_t(X_i)$ is the output of the t_{th} tree, and regularization values become the corresponding values of the latest tree. So, the final prediction is the sum of the output of numerous trees. The following setting parameters were used to tune the XGBoost Model: 5000 total number of trees, 50 early stopping rounds, maximum depth of the tree is 3, learning rate 0.1, linear regression objective function, and gbtree booster.

c) Catboost: Prokhorenkova et al. [53] introduced the symmetric decision tree-based Catboost model in 2018. It accurately processes categorical features with minimal information loss. First, it employs the GBM technique of ordered boosting to address target leaking difficulties. Second, it is the best algorithm for dealing with small datasets. Thirdly, Catboost computes the frequency of a certain feature category, applies statistics procedures, and then incorporates some hyper-parameters to produce numeral features. It converts the categorial features into numerical features by preprocessing. Moreover, it solves the issues of prediction shift, gradient bias, and overfitting and improves model accuracy. Equation (4) explains the transformation of categorical features into numerical, where C_c is the class counter, average target T_{avg}, P_r is the initial numerator value, and C_t is the total counter.

$$T_{avg} = \frac{C_c + P_r}{C_t + 1} \tag{5}$$

$$F_(t_i) = \sum_{k=1}^{N} c_k 1(t \in r_j) \tag{6}$$

In equation (5), $F_(t_i)$ is the tree function of t_i explanatory variable and r_j is the disjoint region in correspondence to the leaves of the decision tree. We have used the following hyperparameters to tune the Catboost model: 0.043 learning rate, the symmetric tree growing policy, plain boosting type, maximum leaves 64, 0.800 subsample, and 1000 iterations.

In this work, the prediction of network resource utilization is a regression problem, where a model f maps the input feature vector onto the label values (target features). The supervised learning model f is trained using the training dataset. The regression problem is represented as a minimization problem in equation (6). The first component of the objective function is an empirical risk, which is described by a loss function that determines the model's f quality. The regularization term is the second component that calculates the complexity of model f. $lambda$ denotes the regularization parameter.

$$min_(f_m) = \sum_{k=1}^{n} L(f_m((X_i), Y_i) + \lambda(f_m) \tag{7}$$

We have used network resource usage dataset d with n attributes $D = (x_1, y_1), (x_2, y_2), ...(x_n, y_n)$. We split dataset D into training and testing sets for training and validating the proposed STEL model. The hybrid STEL model's operational mechanism is described in Algorithm 1. Base learner models are trained with a training set, and the validation set is used to validate the final prediction outcomes. The base learners (Level-0) prediction results are inputted to train the meta-leaner XGBoost model. Consequently, the XGBoost was built using two base learners. The XGBoost model prediction results are validated using the test set. So, the final resource usage prediction results are the meta-regressor model results. The STEL algorithm's final resource usage prediction results are the linear combination of base learner models: Catboost and GBM, as formalized in equation (7). Where $P_(STEL)$ denotes the final prediction, P_{CB} and P_{GBM} represents output of Catboost and GBM method, α and β presents weighting coefficient obtained from XGBoost fitting, and γ denotes constant of correction.

$$P_(STEL) = (P_{GBM} * \alpha) + (P_{CB} * \beta) + \gamma \tag{8}$$

d) Cloud Datasets for Evaluation: To evaluate the performance of our STEL model, three real-time workload traces were used: Materna (Bitbrains) [54], Google Cluster Traces (GCT) [55], and VNFs traces collected from our testbed. The Materna traces were collected over a three-month period with 1750 VMs and included 12 performance metrics such as CPU usage, RAM, Disk usage, and throughput. The GCT workload dataset, which was collected by Google, includes CPU and memory utilization data of 12000 VMs every 5 minutes for 29 days. The third dataset, the VNF traces collected from our testbed, contains CPU consumption in percentage over a 5-minute interval. For a better understanding and analysis of dataset patterns, Heat map visualization of the Bitbrains dataset is illustrated in Figure 4, which depicts the correlation

Algorithm 1 Stacking ensemble learning model for network resource workload prediction.

$D \leftarrow \{(x_1, y_1), (x_2, y_2)...(x_n, y_n)\}$ ▷ network resource utilization dataset
x_n = Feature Vector, y_n = Final Prediction $=>$ n = Total Observations
$Base\_Learners(level-0) \leftarrow \{B_1, B_2...B_z\}$
$Meta\_Learner(level-1) \leftarrow M_l$
$Ensemble\_Total\_Size(level-1) \leftarrow N$
for n=1 to N **do**
    $B_z \leftarrow$ { creates $Base\_Learners$ from D }
end for
$\rightarrow$ creation of new dataset (D_{new}) for $Meta\_Learner$
$D_{new} \leftarrow 0$
for m=1 to M **do**
    for n=1 to N **do**
        $\rightarrow$ make prediction with $Meta\_Learner$
        $B_{mn} = B_z(x_m)$
    end for
    $D_{new} = D_{new} \cup \{((B_{m1}, B_{m2}...B_{mz}), y_m)\}$ ▷ combine different base regressor models
end for
$\rightarrow$ training of $Meta\_Learner$ with new dataset D_{new}
$M_l^t = M_l, (D_{new})$ ▷ training of level-1 model
Prediction results $\rightarrow$ final prediction of $Meta\_Learner$ (M_l^t)
Model validation $\rightarrow$ test performance based on $RMSE, MSE, MAE, MAPE, R^2$ error metrics.

between distinct features. This correlation heatmap effectively illustrates the mutual influence of various parameters on one another, notably highlighting a notable connection between CPU usage and memory usage. Figure 5 highlights the dataset's CPU and memory utilization patterns. It can be observed that the actual CPU and memory are within the range of 10 to 70% usage, but there are some spikes with 80% and 90% usage. So, the ML model learns from these patterns and predicts future CPU and memory usage, which will be helpful in detecting the overloading of resources causing performance degradation and SLA violation.

Furthermore, we preprocessed the dataset using various techniques, including outlier reduction, null value removal, and data cleaning operations. Additionally, we split the data with an 80:20 ratio for STEL model training and testing. The STEL model performs satisfactorily, with a maximum $R^2 = 0.93$ regression accuracy for CPU utilization and a maximum $R^2 = 0.94$ for memory consumption. It predicts future CPU usage and RAM usage of the core VNFs. We performed short-term and mid-term forecasts such as minutes and hours in C-DAF. Because of the recent studies, these two parameters are important and used in many statistical and heuristic-based algorithms to perform host mode detection, VNF autoscaling, and the consolidation of VNFs in the data center. Due to that, we have performed CPU and RAM

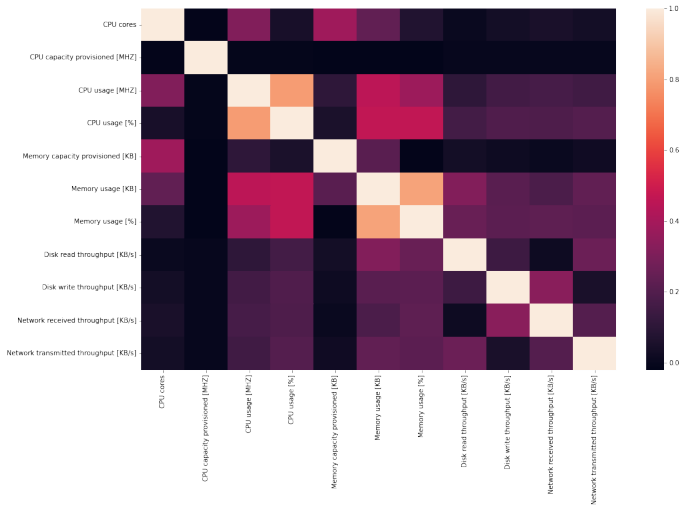

Fig. 4: Heatplot illustrates the correlation between features of Bitbrains dataset

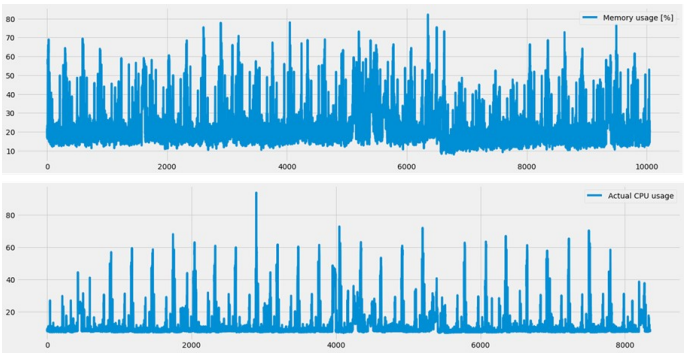

Fig. 5: Illustrates the CPU and memory usage patterns in percentage from Bitbrains dataset

usage prediction for autoscaling of VNFs, and overload and underload detection.

2) Network Anomaly Detection through AutoML and Voting Ensemble Learning: Similar to network resource prediction, anomaly detection and mitigation is also a crucial aspect of network automation. It is a critical use case of NWDAF defined by 3GPP. Network security has been a big challenge due to the recent increase in smart devices. Therefore, anomaly detection and mitigation have gained a lot of attention from researchers. So, an automated ML mechanism is required to detect the anomalies in the system and take proper action to mitigate them from the network. So, we have proposed an efficient H20 AutoML and voting ensemble learning-based mechanism to detect and mitigate the anomalies from the 5G network.

Figure 6 illustrates the proposed AutoML and voting ensemble learning method workflow for network anomaly detection. The functional aspects of the proposed method are explained in various modules. In module 1, raw historical data collected from the network is forwarded to the data preparation and preprocessing module. Module 2 contains

various data preprocessing operations such as data cleaning, data transformation, feature engineering, feature selection, and reduction. Data cleaning and transformation handles null values, outlier removal, and data normalization. Conversely, the role of feature engineering is to perform feature analysis to check the importance of all the features, remove irrelevant features, and select important features for training the ML models. After performing data preprocessing and splitting, the selected most important training features are forwarded to the AutoML module to check the output of various ML classifiers. The AutoML module ranked the ML classifiers based on the classification results. After that, in the next module, we selected the top three performers ML classifier (CL) from the AutoML module for performing the final prediction of network anomalies. These ML classifiers are Random Forest, Catboost, and XGboost. The ML classifiers' are trained on training data, and the results of these ML classifiers are combined through a voting ensemble learning approach for performing the final prediction. The voting ensemble learning method classifies the traffic as per most voters from the AutoML classifiers. So, testing data is used to evaluate the proposed AutoML and voting ensemble learning mechanism. Several error matrices, such as the receiver operating characteristic curve, F-score, and accuracy, have been used to validate the proposed work performance.

We have validated our mechanism on two datasets: a well-known CICDDoS 2019 dataset [56] and an attack dataset from a network slicing testbed. The first CICDDoS dataset was collected by the Canadian Institute for Cybersecurity, which contains 80 features and 12 different attack types. We have performed various preprocessing and feature engineering operations and selected the most important features, such as source port, destination port, flow duration, fwd packet length std, total fwd packet, protocol, ack flag count, MinSegSizeForward and destination port for training the proposed method. We have used the 80:20% data for training and testing the AutoML and voting ensemble learning-based mechanism. Conversely, the second dataset was collected from a 5G network slicing testbed containing 84 features, including slice information [57]. We have used the 12 most important features such as destination-IP, source port, flow duration, destination port, source-IP, fwd packet length std, ACK flag count, total fwd packet, protocol, slice-number, MinSegSizeForward for attack detection. The attack traffic was generated using the Hping3 tool, and normal traffic was recorded from UE to the data network(server). We have used 80% of the dataset for training and 20% for testing purposes. The AutoML-trained model, along with the voting ensemble-learning approach, efficiently identifies network anomalies and reports them to the IBN platform. This enables the execution of appropriate mitigation policies to halt the anomalous flow.

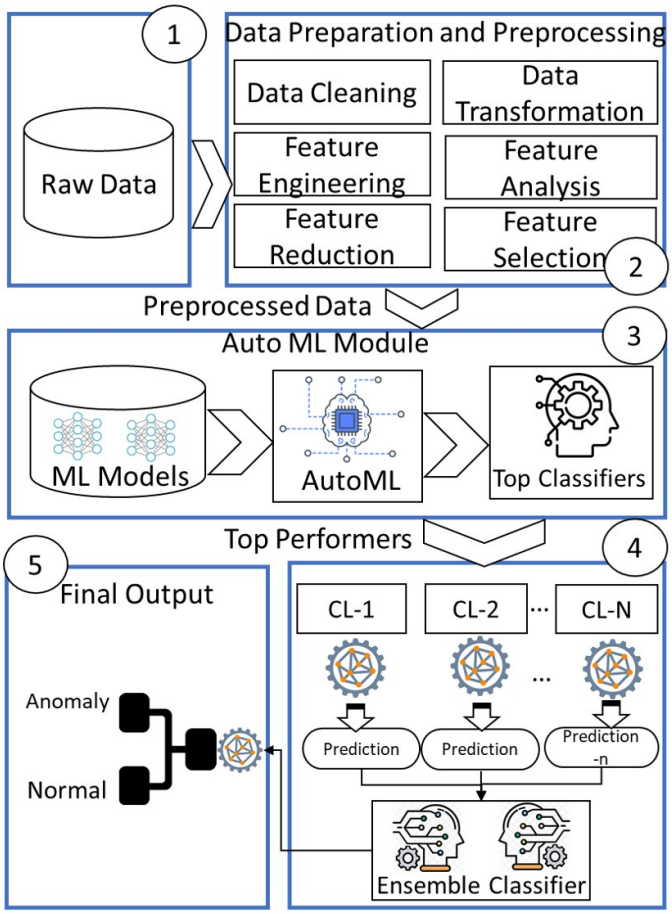

Fig. 6: Anomaly detection through AutoML and ensemble learning model

IV. Experimental Analysis and Results

A. Network Slicing Results

The testbed of our implemented system comprises the IBN framework, OSM orchestrator, FlexRAN, ML-NWDAF, and monitoring mechanism. We used 5G core components, eNodeB/gNodeB, and simulated UEs from Open Air Interface (OAI) for deploying 5G core and RAN setup [58]. 5G core contains core network VNFs such as AMF, UPF, and SMF. OSM with OpenStack operates within a server machine featuring 32 cores and 252GB of memory. The OAI eNB/gNB facilitates the RAN network capabilities, utilizing a software-defined radio (SDR) universal software radio peripheral (USRP) B210. Notably, the OAI gNB is implemented on a PC equipped with 16 GB RAM, Core i5 CPU, USB port 3.0, and Ethernet 1GBit/s. The FlexRAN controller, a key element, is hosted on a separate PC furnished with 16GB RAM, i5 7400 CPU, and Ubuntu 18.04. For performing e2e network slicing, OSM deploys core VNFs using VIM OpenStack, and FlexRAN has been used to perform RAN slicing. Moreover, dedicated core VNFs are assigned to each RAN slice. Connectivity between the deployed core and RAN networks is established following a 5G network topology.

Besides, IBN is a global orchestrator for automating the configuration generation process for OSM and FlexRAN. The IBN application was developed using Python, Java, and MYSQL languages and executed on a designated PC. IBN platform has a web portal where users can input slice require-

ments, including data rate (downlink and uplink throughput), latency, slice ID, and slice type, and in return, the system automatically translates the higher-level configurations into policies with the help of other modules. Afterward, the created policies are deployed using the FlexRAN controller and OSM orchestrator to activate the RAN and core slice. Moreover, network data analytics makes the IBN platform an autonomous system that can perform proactive autoscaling of core VNFs and detect overloaded and underloaded hosts.

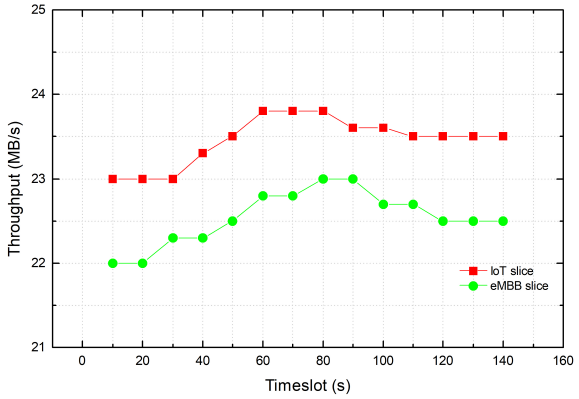

Fig. 7: Achieved downlink throughput results with two deployed slices through the IBN

The policy/slice template contains all the information related to the QoS requirements, such as bandwidth, memory, CPU, instance (VNF) image information, etc. For example, the RAN policy template generated by the RAN policy generator contains a public land mobile network (PLMN), uplink, downlink, slice-ID SNSSAI, and type of service. Further, these slice QoS requirements are converted into the required number of radio blocks through the RAN slicing application inside the FlexRAN controller. Besides, the FlexRAN controller deploys the resource through a static approach such as 50% 40% of RAN resources. So, in our mechanism, we have used both static and dynamic policies for slicing the RAN resources. Conversely, the dedicated EPC VNFs are activated similarly to the specified resources in the user contract and assigned to the RAN slice. The RAN slice template contains dedicated MME and other EPC VNF configurations for establishing an e2e connection. So, in this way, our IBN system instantiates and activates core and RAN network slices over the infrastructure. In addition, NWDAF ML models have been implemented using Python language, TensorFlow, Pandas, and Matplotlib. We have also used Anaconda Jupyter and Google Colab platforms for training and testing the ML models.

We conducted iPerf tests to evaluate the stability of our IBN system by deploying three different types of slices IoT, eMBB, and URLLC slice. In the first case, we have deployed two slices, eMBB, and IoT, with 50% of RAN resources, by inserting QoS requirements through the portal, and the

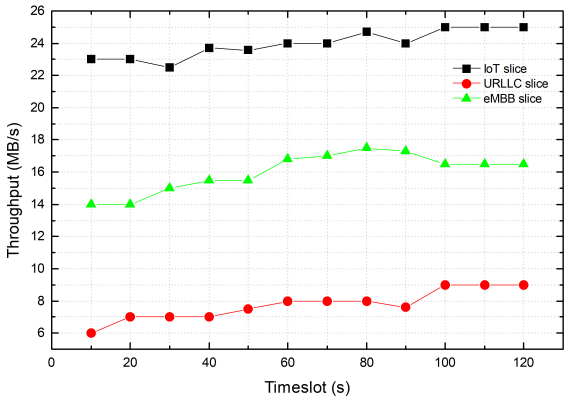

Fig. 8: Achieved downlink throughput results with three deployed slices through the IBN

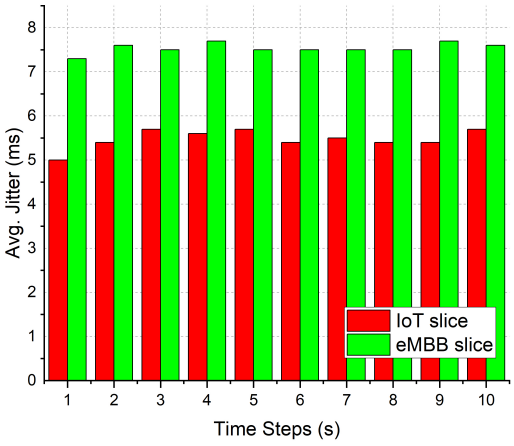

Fig. 9: Average jitter delay for two activated slices

system automatically activates the e2e network slice with the cooperation of other components. Figure 7 shows the downlink speed recorded through the deployed e2e eMBB and IoT slice. The eMBB and IoT slices show almost similar throughput maximum of 23MB/s because of the equal resources requested through the SLA. Conversely, Figure 8 presents recorded downlink throughput results of three slices with different requirements. We have deployed eMBB, URLLC, and IoT network slices with 30%, 20%, and 50% of total RAN resources, respectively. The IoT slice achieved a downlink throughput of 25 MB/s, and the eMBB slice 17 MB/s. Conversely, the URLLC slice achieved a maximum of 9 MB/s downlink throughput. We have calculated the average jitter delay on two deployed slices for more evaluation of our IBN-based network slicing testbed. Figure 9 depicts the average jitter by performing multiple tests on IoT and eMBB slices, where the IoT slice has achieved a lower jitter of around 5ms, and

eMBB achieved 7.1ms, showing satisfactory performance. The primary purpose of testing the throughput is not to compare the performance with industrial standards but to show the stability of the IBN framework for creating e2e slices.

B. ML-STEL method performance Analysis and Evaluation

This section provides the STEL method's performance analysis with the help of benchmark methods and error measures. It performs a short-term future prediction based on 5m. We used five error metrics to validate STEL method prediction performance: mean square error (MSE), mean absolute percentage error (MAPE), root mean square error (RMSE), and regression-score accuracy R^2. According to the literature, a high regression score R^2 and low MSE, RMSE, MAE, and MAPE are used to validate the regression model's results. The mathematical equations for each of the performance metrics are listed below. Where A_i and P_i are the actual and predicted value at the time i, and k is the prediction window size.

$$MSE = 1/k \sum_{i}^{k} (A_i - P_i)^2 \tag{8}$$

$$RMSE = \sqrt{1/k \sum_{i}^{k} (A_i - P_i)^2} \tag{9}$$

$$MAE = 1/k \sum_{i}^{k} |A_i - P_i| \tag{10}$$

$$MAPE = 1/k \sum_{i}^{k} |\frac{A_i - P_i}{P_i}| \tag{11}$$

$$R^2 = 1 - \frac{\sum_i^k (A_i - P_i)^2}{\sum_i^k (A_i - P^{-}{}^{i})^2} \tag{12}$$

The prediction results show that the proposed STEL method performs better than others. The STEL method achieved lower RMSE, MSE, MAE, and MAPE and high regression scores for memory and CPU utilization prediction. Conversely, it achieved the maximum of $R^2 = 0.94$ and $R^2 = 0.93$ accuracy for memory and CPU usage prediction. Table 1 highlights the achieved error metrics by the STEL model while performing memory and CPU utilization prediction on two datasets.

Figure 10 illustrates the CPU and memory utilization forecasting results of the implemented STEL model on the Bitbrains dataset. The CPU workload prediction results are close to the actual utilization pattern, showing the STEL model's satisfactory performance in accuracy. The predicted results are dependent on the dynamics of the resource usage dataset. Figure 10 (a) presents the CPU workload prediction results on the Bitbraisn dataset where the STEL method achieved lower error rates $RMSE = 4.51$, $MSE = 9.02$, $MAE = 3.93$, and $MAPE = 7.22$. Conversely, it achieved a $R^2 = 0.90$ regression accuracy for CPU prediction. However, it can be visible from the high CPU usage spikes in the diagram that there are some prediction errors. These errors are the result of a sudden increase in CPU usage. So, the model could have predicted these points better because these values are outliers and do not follow usage patterns in the dataset. Figure 10 (b) depicts the memory utilization prediction results on the Bitbrains dataset where the implemented STEL model achieved lower $RMSE = 2.01$, $MSE = 4.02$, $MAE = 1.44$, $MAPE = 4.22$ and a high $R^2 = 0.94$. These error measures are minimal as compared to other models.

TABLE I: Illustrates the achieved error metrics by proposed STEL method while performing network workload prediction

	GCT Dataset		Bitbrain Dataset	
Evaluation Metrics	Memory	CPU	Memory	CPU
MSE	8.23	4.50	4.02	9.02
RMSE	4.11	2.25	2.01	4.51
MAE	3.72	1.11	1.44	3.93
MAPE	6.82	4.55	4.22	7.22
R^2	0.91	0.93	0.94	0.90

Figure 11 illustrates the prediction results of the implemented STEL model on the GCT dataset. Figure 11 (a) presents the CPU utilization prediction outcomes where our STEL method obtained $RMSE = 2.25$, $MSE = 4.50$, $MAE = 1.11$, $MAPE = 4.55$ and $R^2 = 0.93$. Conversely, Figure 11 (b) depicts the memory utilization prediction results where our STEL method achieved $RMSE = 4.11$, $MSE = 8.23$, $MAE = 3.72$, $MAPE = 6.82$ and $R^2 = 0.91$. Overall, the STEL model performed well on the GCT dataset, with low error rates and a high regression score of $R^2 > 0.90$. Additionally, the STEL model demonstrated comparable performance on the VNFs dataset, with a high regression accuracy of $R^2 = 0.89$ when forecasting future CPU usage.

Table 2 compares the STEL method with state-of-the-art LR, LSTM, SVR, ANN, hybrid CNN and LSTM, and deep belief network (DBN) models. To perform a thorough comparison, these models were applied to all datasets. On the GCT dataset, the STEL method outperformed other models, achieving an RMSE of 2.25, while LR, LSTM, SVR, ANN, CNN and LSTM, and DBN obtained RMSE values of 7.79, 6.20,5.73, 12.77, 8.32 and 11.27, respectively. Similarly, on the Bitbrains dataset, the STEL method achieved an RMSE of 4.52, while LR, LSTM, SVR, and ANN obtained RMSE values of 14.23, 7.22, 12.48, 42.34, 10.13, and 9.51, respectively. The results show that the implemented STEL method outperforms other models in resource workload prediction, with lower error rates on both benchmark datasets.

The results indicate that the hybrid STEL model demonstrates adequate performance with higher accuracy and lower error rates when predicting multi-attribute network resource utilization. Moreover, the IBN decision engine is a decisive part that uses the STEL model's short-term prediction outputs for autoscaling resources, host overload, and underload detection. It contains dynamic thresholding such as Inter Quartile Range (IQR) and Local Regression (LR) based methodology that determines the VNF scaling decision, for example, overloading and underloading. IBN triggers a policy to deploy extra VNFs by a sudden increase in network traffic to prevent

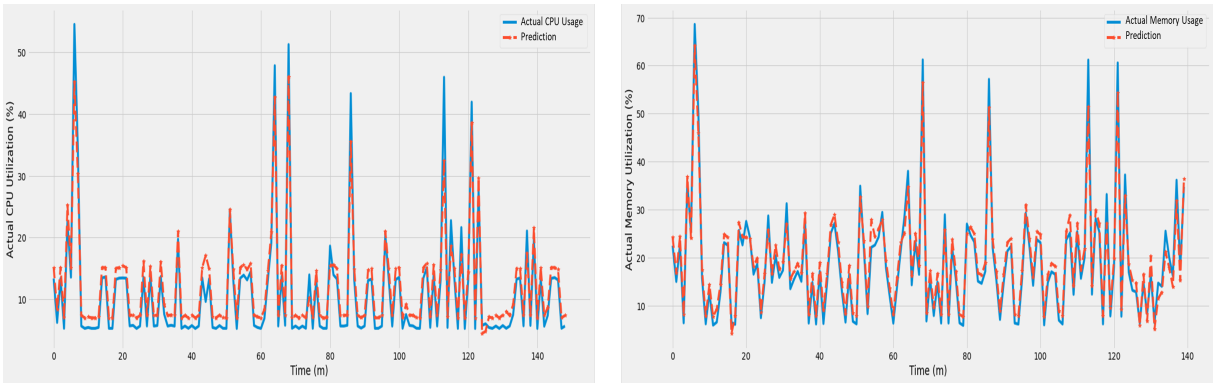

Fig. 10: Presents STEL model prediction results on Bitbrains dataset a) illustrates the actual and predicted CPU utilization b) Illustrates actual and predicted memory utilization

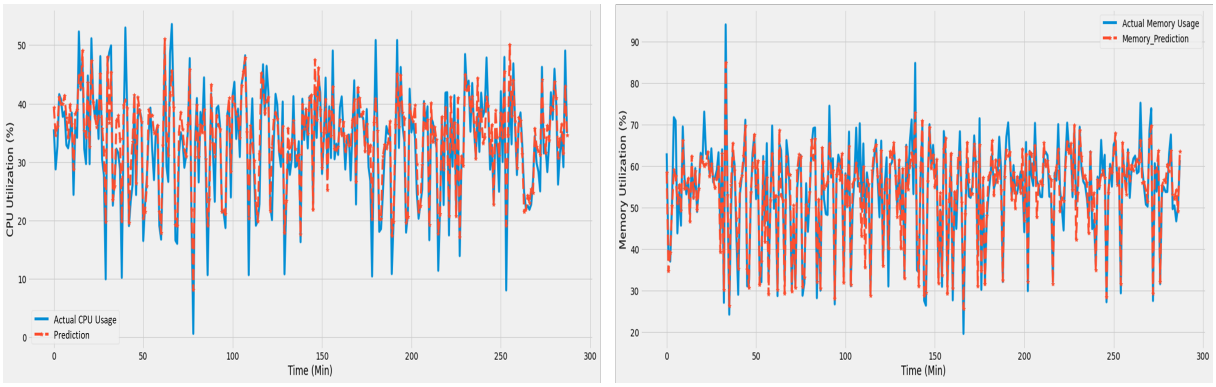

Fig. 11: Presents STEL model prediction results on GCT dataset a) illustrates the actual and predicted CPU utilization b) Illustrates actual and predicted memory utilization

TABLE II: Comparison of proposed STEL method with other state-of-the-art methods

Models	RMSE (GCT Dataset)	RMSE (Bitbrain Dataset)
LR	7.79	14.23
LSTM	6.203	7.22
SVR	5.73	12.48
ANN	12.77	42.34
CNN+LSTM	8.32	10.13
DBN	11.27	9.51
Proposed STEL	2.25	4.51

performance degradation. So, IBN ensures QoS assurance and SLA violation reduction due to directions provided by the decision engine based on ML-based STEL prediction results.

C. AutoML and Voting Ensemble Learning method performance Evaluation

This section explains the results achieved through the proposed AutoML and voting ensemble learning mechanism for attack detection and classification from the networks. We have used the 80:20 ratio of both datasets to train and test the model. Moreover, we have compared the proposed method with BLSTM, SVM, MLP, and deep neural network (DNN) based on well-known error measures such as accuracy, precision, recall, and F1 measure. These are the important metrics to validate the classification performance of ML models. Accuracy is used as an evaluation metric that shows the overall performance of the ML models. Equation 1 illustrates the mathematics behind the accuracy (A) metric, where T_p presents the number of correctly classified instances as N, F_p shows the number of cases wrongly classified as N, Tn highlights the number of correctly classified instances Not-N, and F_n denotes the number of wrongly classified instances Not-N.

$$A = \frac{T_p + T_n}{T_p + F_n + F_p + T_n} * 100\% \tag{13}$$

The results presented in Figure 12 illustrate the achieved error measures through the proposed AutoML and voting ensemble learning model and others BLSTM, SVM, MLP, and DNN on the CICDDoS dataset. It can be observed that our model achieved almost 99.22% accuracy, 99.20 F1-score, 99.31 recall, and 99.15 precision. Besides, the BLSTM, SVM, DNN, and ML achieved 97.53%, 96%, 90.22%, and 86.63% accuracy, respectively. Hence, our AutoML and voting ensemble learning model outperforms other models and performs optimally.

Figure 13 depicts the achieved error measures of the proposed method compared to other methods on the net-

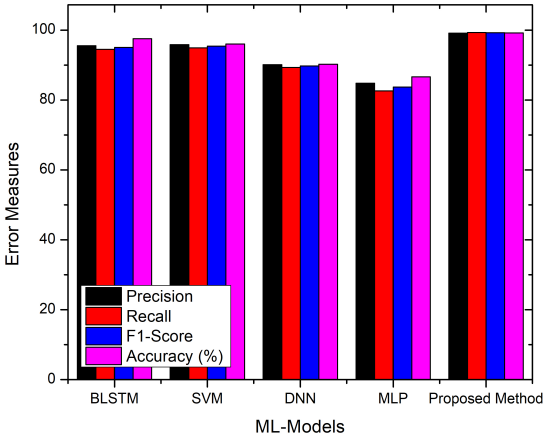

Fig. 12: Result of achieved error measures on CICDDoS attack dataset

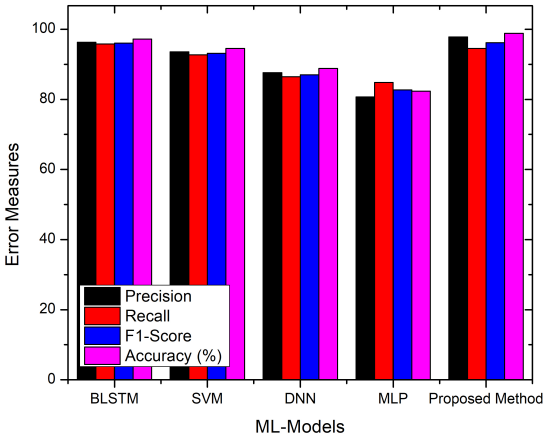

Fig. 13: Result of achieved error measures on network slicing attack dataset

work slicing dataset, where the proposed method achieved $accuracy = 98.84\%$, $F1 - sore = 96.146$, $recall = 94.54$, and $precision = 97.81$. The other methods achieved accuracy of $BLSTM = 97.21\%$, $SVM = 94.53\%$, $DNN = 88.82\%$, and $MLP = 82.84\%$. So, our method achieved 98.84% accuracy for network anomaly prediction and performed better than other methods. Further, MLP, DNN, and BLSTM models take more training time and consume more resources than our model. However, the proposed model's superiority indicates the model's effectiveness while performing attack classification on two different datasets. It is incredibly efficient in terms of speed and resource consumption. Besides, the IBN decision engine uses the attack detection and classification results generated by the hybrid model, which mitigates the abnormal flow from the system. Consequently, our NWDAF offers dual functionalities: detecting and preventing attacks within the network through optimized AutoML, as well as predicting future resource load using the STEL model. So, the IBN platform with NWDAF assures proactive and efficient management of the network resources.

D. Discussion

The presented results show that the IBN platform can automatically create, activate, and deactivate the network slices. The multiple VNFs are deployed through our system for creating core and RAN network slicing. Hence, our implemented system performs stable while creating multiple e2e network slicing. It is a closed-loop mechanism that can orchestrate and manage the complete lifecycle of e2e network slices. Moreover, the ML-based NWDAF module is an interesting feature of our system, which predicts future resource utilization through the STEL model and detects network anomalies. The IBN system uses these prediction results for efficient and proactive management of the core resources. So, our system can proactively prepare the resources whenever the traffic demands increase. It performs autoscaling by checking STEL prediction and efficiently placing the VNFs on the edge or core cloud location.

E. Limitations

Despite many advantages of the implemented mechanism, there are a few limitations. The dynamic allocation of RAN resources still needs to be improved. We plan to develop a reinforcement learning-based application to dynamically provision RAN resources for each slice type. Further, our system has yet to handle the mobility management of the users. We need to add more functionalities to NWDAF for the RAN part, such as RAN slice traffic prediction, dynamic provisioning of resources, and mobility prediction. Furthermore, our mechanism still needs to extend intent translation through natural language processing (NLP) to achieve zero-touch network automation. Intent translation through NLP needs a suitable vocabulary dataset related to network slicing. So, we will prepare a dataset related to network slicing for developing an intent translation mechanism through NLP.

V. Conclusion

This paper explained the IBN framework and novel network data analytics mechanisms for automating, managing, and proactively updating network slices for novel consumer electronics and B5G services. IBN follows a one-touch process where the user inputs abstract slice QoS requirements. In return, the system translates them into policies and deploys them over the core and RAN infrastructure using the various orchestrator and RAN controllers. It creates, activates, monitors, and deactivates the e2e slices in an automated fashion. It can also handle and manage the network slices over the multidomain infrastructure. Several tests have been conducted to evaluate IBN mechanism performance by deploying multiple slices over the virtual and physical infrastructure that show adequate

performance in resource stability, automated resource provisioning, customization, and resource assurance. Moreover, the STEL model and optimized autoML-based methods have been implemented inside NWDAF to predict network resource usage and detect network anomalies. So, AI-based network data analytics with IBN provides the functionalities of autoscaling core resources and detecting and preventing attacked traffic from the network. So, IBN uses these prediction results to guarantee QoS and improve network security. In the future, we will add more functionalities to the IBN system by using novel AI technologies for network automation, such as transformers for efficient resource forecasting and blockchain-based zero trust networks for future 6G networks.

REFERENCES

[1] S. J. Nawaz, S. K. Sharma, M. N. Patwary, and M. Asaduzzaman, "Next-generation consumer electronics for 6g wireless era," *IEEE Access*, vol. 9, pp. 143 198–143 211, 2021.

[2] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[3] K. Katsalis, N. Nikaein, E. Schiller, A. Ksentini, and T. Braun, "Network slices toward 5g communications: Slicing the lte network," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 146–154, 2017.

[4] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE communications magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[5] K. Abbas, M. Afaq, T. Ahmed Khan, A. Rafiq, J. Iqbal, I. Ul Islam, and W.-C. Song, "An efficient sdn-based lte-wifi spectrum aggregation system for heterogeneous 5g networks," *Transactions on Emerging Telecommunications Technologies*, p. e3943, 2020.

[6] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106984, 2020.

[7] S. Zhang, "An overview of network slicing for 5g," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111–117, 2019.

[8] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, "5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view," *IEEE Access*, vol. 6, pp. 55 765–55 779, 2018.

[9] M. Jemmali, M. Denden, W. Boulila, R. H. Jhaveri, G. Srivastava, and T. R. Gadekallu, "A novel model based on window-pass preferences for data-emergency-aware scheduling in computer networks," *IEEE Transactions on Industrial Informatics*, 2022.

[10] S. Bolettieri, D. T. Bui, and R. Bruno, "Towards end-to-end application slicing in multi-access edge computing systems: Architecture discussion and proof-of-concept," *Future Generation Computer Systems*, 2022.

[11] L. Xu, X. Zhou, X. Li, R. H. Jhaveri, T. R. Gadekallu, and Y. Ding, "Mobile collaborative secrecy performance prediction for artificial iot networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5403–5411, 2021.

[12] 3GPP, "Telecommunication management; study on management and orchestration of network slicing for next generation network," [Available Online], https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3091.

[13] IETF, "Intent-based networking," https://tools.ietf.org/html/draft-irtf-nmrg-ibn-concepts-definitions-01.

[14] T. S. 3GPP, "5g; management and orchestration; concepts, use cases and requirements," [Available Online], https://www.etsi.org/deliver/etsi_ts/128500_128599/128530/15.00.00_60/ts_128530v150000p.pdf.

[15] D. Bega, M. Gramaglia, R. Perez, M. Fiore, A. Banchs, and X. Costa-Perez, "Ai-based autonomous control, management, and orchestration in 5g: From standards to algorithms," *IEEE Network*, vol. 34, no. 6, pp. 14–20, 2020.

[16] K. Abbas, T. A. Khan, M. Afaq, and W.-C. Song, "Network slice lifecycle management for 5g mobile networks: An intent-based networking approach," *IEEE Access*, 2021.

[17] E. Pateromichelakis, F. Moggio, C. Mannweiler, P. Arnold, M. Shariat, M. Einhaus, Q. Wei, Ö. Bulakci, and A. De Domenico, "End-to-end data analytics framework for 5g architecture," *IEEE Access*, vol. 7, pp. 40 295–40 312, 2019.

[18] 5GPPP, "View on 5g architecture: white paper)," [Available Online], https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf.

[19] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa, and A. Ksentini, "Network slicing-based customization of 5g mobile services," *IEEE Network*, vol. 33, no. 5, pp. 134–141, 2019.

[20] N. F. S. De Sousa, D. A. L. Perez, R. V. Rosa, M. A. Santos, and C. E. Rothenberg, "Network service orchestration: A survey," *Computer Communications*, vol. 142, pp. 69–94, 2019.

[21] ONAP, "Onap: open networking automation platform," https://www.onap.org/;accessed:20.5.2020.

[22] OpenBaton, "Openbaton: Nfv mano-based framework," https://openbaton.github.io/;accessed:25.5.2020.

[23] OSM, "Open source mano," https://osm-download.etsi.org/ftp/Documentation/201902-osm-scope-white-paper/#!02-osm-scope-and-functionality.md;accessed:1.6.2020.

[24] Cloudify, "Cloudify: A open source network orchestrator," https://cloudify.co/;accessed:3.6.2020.

[25] K. Katsalis, N. Nikaein, and A. Huang, "Jox: An event-driven orchestrator for 5g network slicing," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[26] Mosaic5G, "Network sharing using flexran," https://gitlab.eurecom.fr/mosaic5g/mosaic5g/-/wikis/tutorials/slicing;accessed:5.5.2020.

[27] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5g evolution: A view on 5g cellular technology beyond 3gpp release 15," *IEEE Access*, vol. 7, pp. 127 639–127 651, 2019.

[28] F. Meneses, M. Fernandes, D. Corujo, and R. L. Aguiar, "Slimano: an expandable framework for the management and orchestration of end-to-end network slices," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*. IEEE, 2019, pp. 1–6.

[29] X. Li, R. Ni, J. Chen, Y. Lyu, Z. Rong, and R. Du, "End-to-end network slicing in radio access network, transport network and core network domains," *IEEE Access*, vol. 8, pp. 29 525–29 537, 2020.

[30] Cisco, "Intent-based networking," [Avaliable Online], https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/digital-network-architecture/nb-09-intent-networking-wp-cte-en.pdf.

[31] Huawei, "Intent-driven network," https://carrier.huawei.com/~/media/CNBG/Downloads/Spotlight/all-cloud-network-towards-5g/idn-en.pdf.

[32] Apstra, "Intent-based networking: A next-gen vision for the next-gen network," https://go.apstra.com/white-paper-apstra-intent-based-networking.

[33] S. Sevgican, M. Turan, K. Gökarslan, H. B. Yilmaz, and T. Tugcu, "Intelligent network data analytics function in 5g cellular networks using machine learning," *Journal of Communications and Networks*, vol. 22, no. 3, pp. 269–280, 2020.

[34] S. Ouhame, Y. Hadi, and A. Ullah, "An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model," *Neural Computing and Applications*, vol. 33, no. 16, pp. 10 043–10 055, 2021.

[35] L. Abdullah, H. Li, S. Al-Jamali, A. Al-Badwi, and C. Ruan, "Predicting multi-attribute host resource utilization using support vector regression technique," *IEEE Access*, vol. 8, pp. 66 048–66 067, 2020.

[36] W. Iqbal, J. L. Berral, A. Erradi, D. Carrera *et al.*, "Adaptive prediction models for data center resources utilization estimation," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1681–1693, 2019.

[37] H. Xia, X. Wei, Y. Gao, and H. Lv, "Traffic prediction based on ensemble machine learning strategies with bagging and lightgbm," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2019, pp. 1–6.

[38] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host cpu utilization in cloud computing using recurrent neural networks," in *2017 12th international conference for internet technology and secured transactions (ICITST)*. IEEE, 2017, pp. 67–72.

[39] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[40] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Generation Computer Systems*, vol. 81, pp. 41–52, 2018.

[41] S. Gupta and D. A. Dinesh, "Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks," in *2017 IEEE international conference on advanced networks and telecommunications systems (ANTS)*. IEEE, 2017, pp. 1–6.

[42] S. Garg and S. Batra, "A novel ensembled technique for anomaly detection," *International Journal of Communication Systems*, vol. 30, no. 11, p. e3248, 2017.

[43] J. Hu, X. Yu, D. Qiu, and H.-H. Chen, "A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection," *IEEE network*, vol. 23, no. 1, pp. 42–47, 2009.

[44] Y. Yu, L. Guo, Y. Liu, J. Zheng, and Y. Zong, "An efficient sdn-based ddos attack detection and rapid response platform in vehicular networks," *IEEE access*, vol. 6, pp. 44 570–44 579, 2018.

[45] A. Thantharate, R. Paropkari, V. Walunj, C. Beard, and P. Kankariya, "Secure5g: A deep learning framework towards a secure network slicing in 5g and beyond," in *2020 10th annual computing and communication workshop and conference (CCWC)*. IEEE, 2020, pp. 0852–0857.

[46] N. A. E. Kuadey, G. T. Maale, T. Kwantwi, G. Sun, and G. Liu, "Deepsecure: Detection of distributed denial of service attacks on 5g network slicing—deep learning approach," *IEEE Wireless Communications Letters*, vol. 11, no. 3, pp. 488–492, 2021.

[47] D. Sattar and A. Matrawy, "Towards secure slicing: Using slice isolation to mitigate ddos attacks on 5g core network slices," in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 82–90.

[48] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "Flexran: A flexible and programmable platform for software-defined radio access networks," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 427–441.

[49] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[50] F. Divina, A. Gilson, F. Goméz-Vela, M. García Torres, and J. F. Torres, "Stacking ensemble learning for short-term electricity consumption forecasting," *Energies*, vol. 11, no. 4, p. 949, 2018.

[51] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[52] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[53] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.

[54] Materna, "Dataset," http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna/ ;accessed:3.5.2020.

[55] Google, "Google cluster log," https://github.com/google/cluster-data; accessed:5.20.2022.

[56] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–8.

[57] M. S. Khan, B. Farzaneh, N. Shahriar, N. Saha, and R. Boutaba, "Slicesecure: Impact and detection of dos/ddos attacks on 5g network slices."

[58] OpenAirInterface, "Oai: An open-source community," https://www.openairinterface.org/;accessed:10.5.2020.