IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 June 2022, accepted 3 July 2022, date of publication 12 July 2022, date of current version 18 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3190372

RESEARCH ARTICLE

5G Network Management System With Machine Learning Based Analytics

MADANAGOPAL RAMACHANDRAN[1], T. ARCHANA[1], V. DEEPIKA[1], A. ARJUN KUMAR[1], AND KRISHNA M. SIVALINGAM[2], (Fellow, IEEE)

[1]NMSWorks Software Pvt. Ltd., IIT Madras Research Park, Chennai 600113, India
[2]Department of CSE, IIT Madras, Chennai 600036, India

Corresponding author: Madanagopal Ramachandran (madan@nmsworks.co.in)

ABSTRACT Application of intelligent data analytics using machine learning in management of 5G networks can enable autonomous networking capabilities in 5G networks. This paper describes the design and implementation of CygNet MaSoN, a management system supporting advanced aggregation and analytics features combined with machine learning. The system supports detection of anomalous network behaviour, detection of degradation in network performance and service quality and also supports resource optimization. The main objective is to achieve self-organizing and closed loop automation functionalities expected as part of autonomous functioning of 5G networks. Details of the system architecture and components are presented. Three real-life use cases implemented on this system are then described. Machine learning models built and synthetic data generation methods adopted are presented with the features considered. The results obtained using the MaSoN system are also presented to demonstrate the effectiveness of the system in 5G network operations.

INDEX TERMS 5G network management, autonomous networking, closed loop automation, data analytics, machine learning.

I. INTRODUCTION

5G mobile networks have been designed to address the emerging networking needs such as enhanced broadband support, ultra-low latency communications, and massive-scale Internet of Things systems [1]. A 5G system is composed of radio access, transport, and core network components. 5G's Next-Generation Radio Access Network (NG-RAN) is realized using a set of Centralized Units (CU) and Distributed Units (DU) in addition to the radio interfaces. The 5G Core is built using the service-based architecture (SBA), where multiple instances for all core network functions can be deployed. 5G NG-RAN and Core networks are based on principles of control-data plane separation and network function virtualization, and can be realized using data centers and cloud computing [2].

Considering the commonly accepted operational expectation that 5G networks management need to process massive volume of data in real-time, autonomous networking capabilities have to supported as part of its management. This involves continuous system monitoring combined with automated changes in network configuration to handle system issues. Advanced data analytics and machine learning techniques can be used to reduce human intervention and decision making [3], [4].

The 3rd Generation Partnership Project (3GPP) Network Data Analytics Function (NWDAF) has been introduced to enable data analytics related functionality in 5G Core. This will provide inputs to other Network Functions (NFs) and Analytics Functions (AFs) for operational purposes. Management and Orchestration systems are critical components to achieve self-organizing and closed loop automation, and to support analytics functionality to realize autonomous network operations.

This paper describes the design and implementation of a system supporting 5G Core and RAN NF specific element management and virtualized network management functionality along with aggregation and correlation capabilities.

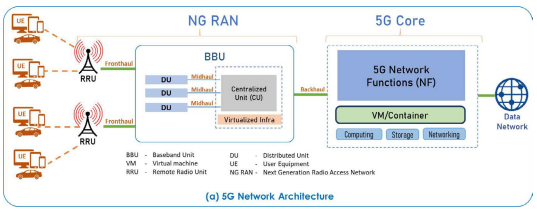
The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen.

This integrated 5G network management system is named **CygNet MaSoN (Manager for Softwarized Networks)**. The architectural and functional aspects of CygNet MaSoN system that support machine learning and analytics in 5G networks are described.

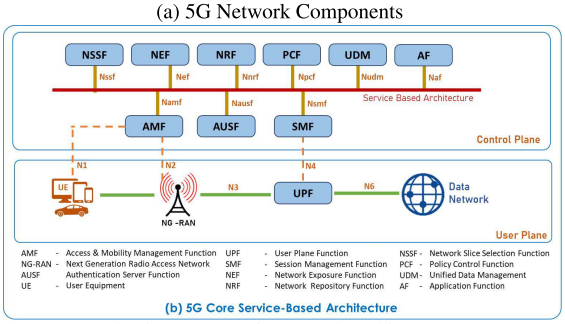
A preliminary version of the MaSoN architecture and two basic use-case studies were presented in a technical **demo** at COMSNETS 2021 conference [5]. In this current paper, significant additional details related to Machine Learning (ML) model features and more detailed performance results are presented. In addition, it presents the design and implementation of an additional use case on Session Management Function (SMF) resource usage prediction.

The major contributions of this paper are:

1) Three different 5G network analytics use cases which showcase the realization of autonomous networking have been implemented. The ML models implemented as part of the three use cases support configurable parameters which can be adjusted when the system is deployed in a real network.
2) The implemented 5G network management system (which has been named MaSoN) for analytics based machine learning is field deployable.
3) The implemented system supports practical analytics use cases in 5G network management and its architecture which is a unique aspect of this work. Further, the implementation framework is extensible for supporting several other practical 5G network management analytics use cases.



(a) 5G Network Components



(b) 5G Core Service-Based Architecture

FIGURE 1. 3GPP 5G architecture.

A demo system of the MaSoN application can be accessed as per the details shared in [6]. The demo system contains all the implemented features including the three implemented use cases described in this paper. Help page in the demo application can be referred to get exact description of the implemented features. Help page is available in the demo application at the top right side in the application client view indicated by the question mark icon.

The rest of the paper is organized as follows. Section II presents a brief background on 5G networks and machine learning for networking systems. Section III describes the details of the proposed CygNet MaSoN system architecture. Section IV presents the three different usecases that demonstrate sample applications of the MaSoN system. The performance results and relevant system operational details are presented in Section V. Section VI summarizes and concludes the paper.

II. BACKGROUND

This section presents the relevant background material and related work. The list of key abbreviations used in this paper along with their expansions is provided in Table 1.

A. 5G NETWORK ARCHITECTURE

Fig. 1(a) presents an overview of the 3GPP-standards based 5G network architecture [1], including the New Radio (NR), Next-Generation Radio Access Network (NG-RAN) and 5G Core components. The User Equipment (UE) nodes communicate using the NG-RAN network and the 5G Core network. As shown, both the NG-RAN and Core components can be realized using a virtualized computing framework, based on Software Defined Networking and Network Function Virtualization. Management and Orchestration of the Virtualized 5G network components can be achieved using the ETSI Network Functions Virtualisation Management and Orchestration framework (NFV MANO) specifications [2].

Fig. 1(b) presents the Service Based Architecture (SBA) of the 5G Core, which is realized using virtualized network functions (VNFs), with each NFV handling a part of core network functionality. The NFVs mostly deal with the control plane, except for the User Plane Function (UPF) that handles packet processing. One of the advantages of the NFV architecture is that the network functions (NF) can be deployed on virtual machines/containers executed on off-the-shelf computing servers, thereby eliminating the need for commodity, vendor-specific hardware. It also enables the system to be elastic in order to meet dynamically changing NF requirements based on the current system load.

B. MACHINE LEARNING FOR NETWORKS

Different standards bodies have proposed some of the high-level generic architectural frameworks related to using machine learning in networks, as described below.

A generic architectural framework for machine learning in future networks including 5G networks (which come under International Mobile Telecommunications-2020 (IMT-2020)) is described in [7]. It lists multiple machine use cases relevant for 5G networks and describes a architectural framework considering the varied requirements across all use cases. The architecture includes ML pipeline, ML sandbox and few other components along with their reference

TABLE 1. List of abbreviations and their expansions.

Abbreviation	Expansion
3GPP	3rd Generation Partnership Project
AF	Analytics Function
API	Application Programming Interface
AR	Auto Regression
ARIMA	Auto Regressive Integrated Moving Average
CM	Configuration Management
CU	Centralized Unit
DU	Distributed Unit
EM	Element Management
ENI	Experiential Networked Intelligence
ETSI	European Telecommunications Standards Institute
FM	Fault Management
GBR	Guaranteed Bit Rate
GPRS	General Packet Radio Service
GTP	GPRS Tunnelling Protocol
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LR	Linear Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MANO	Management and Orchestration
MAPE	Mean Absolute Percentage Error
MaSoN	Manager for Softwarized Networks
ML	Machine Learning
MMPP	Markov-Modulated Poisson Process
MSE	Mean Squared Error
MVGD	Multi Variate Gaussian Distribution
NBI	North Bound Interface
NETCONF	Network Configuration Protocol
NF	Network Function
NFV	Network Functions Virtualisation
NG-RAN	Next-Generation Radio Access Network
NR	New Radio
NWDAF	Network Data Analytics Function
ONAP	Open Network Automation Platform
OSM	Open Source MANO
OSS	Operations Support System
PDU	Protocol Data Unit
PM	Performance Management
PRB	Physical Resource Block
QoS	Quality of Service
REST	Representational State Transfer
RM	Resource Management
RNN	Recurrent Neural Networks
SBA	Service Based Architecture
SBI	South Bound Interface
SDN	Software Defined Networking
SMF	Session Management Function
UE	User Equipment
UL	Uplink
UPF	User Plane Function
VAR	Vector Auto Regression
VNF	Virtualized Network Function
XGBoost	eXtreme Gradient Boosting
YAML	YAML Ain't Markup Language

interfacing details. Different network intelligence levels and dimensions for evaluating them have also been described.

Experiential Networked Intelligence (ENI) [8] is an industry specification group in European Telecommunications Standards Institute (ETSI) which considers use cases along with service, network, functional and non-functional requirements in defining a high level functional architecture for machine learning in future networks. It describes the critical components and external reference points as part of the architecture. This work is aimed at improving network operation experience using closed loop automation functionalities such that real-time changes and issues are handled effectively by minimizing manual intervention.

The above two frameworks are more generic at high level whereas the system described in this paper is actually implemented and can be deployed in actual networks. The entire system supports cloud native deployment since the container based deployment support is available. Since the machine learning models support configurable parameters to be changed to enable model tuning and more recent data is considered for training and prediction, the implemented ML models are adaptive to current trends in processed data.

A detailed survey on the application of machine learning techniques in networking which covers almost all networking areas and technologies are provided in [3]. A similar survey on machine learning application specific to Software Defined Networking (SDN) is provided in [9]. While these are more related to application of machine learning techniques for networking use cases where aspects related to feature selection, model parameters tuning and performance evaluation are covered, this paper in addition to that covers architectural and other aspects related to a implemented system.

III. CygNet MaSoN ARCHITECTURE

This section presents the CygNet MaSoN architecture and its machine learning components.

A. SYSTEM ARCHITECTURE AND COMPONENTS

CygNet MaSoN is a management system providing the Element Management (EM) functionality for the network functions and also part of the Network Management functionality for 5G networks, as initially presented in brief in [5]. The MaSoN architecture and components are shown in Fig. 2. MaSoN follows the ETSI NFV MANO architecture by implementing the EM component. It integrates 5G RAN and 5G-Core Virtual Network Functions and Physical Network Functions. It periodically collects performance, fault and resource management information. MaSoN also supports triggering configuration management requests to the various network functions for effecting configuration changes in the system. Resource Management (RM), Fault Management (FM), Performance Management (PM) and Configuration Management (CM), along with all their important sub-functions constitute the important components of the MaSoN architecture. MaSoN supports 3GPP and ETSI NFV MANO standard interfaces and Application Programming Interface (API) for integrating with 5G NFs for extraction and configuration of management information.

MaSoN integrates with all the NFs over a Representational state transfer (RESTful) API, defined in 3GPP specifications, to collect all the management data across different management functions. The RESTful API is the default and commonly used South Bound Interface (SBI) supported by MaSoN. The data model adopted in MaSoN is based on

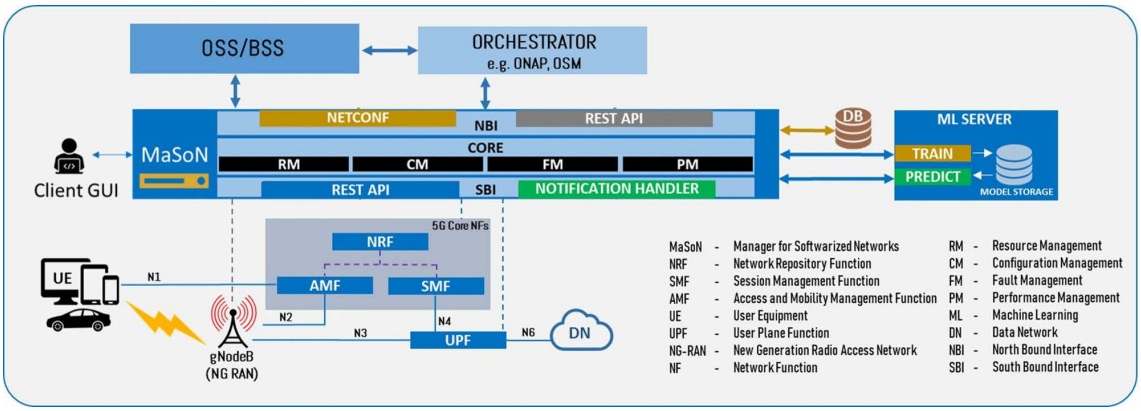


FIGURE 2. CygNet MaSoN architecture and its important components.

JavaScript Object Notation (JSON) and YAML Ain't Markup Language (YAML).

A North Bound Interface (NBI) is provided for integration with Orchestration systems, for handling VNF lifecycle management. Such systems include Open Source MANO (OSM) [10] and Open Network Automation Platform (ONAP) [11]. It also provides an NBI to Operations Support System (OSS) for providing aggregated fault and performance management information. It also handles publishing correlated faults/events and retrieval of aggregated performance data and resource information. A Network Configuration Protocol (NETCONF) interface for configuration management is provided.

MaSoN is built with Self-Organizing capabilities involving management functions so that closed loop automation use cases in 5G networks can be built. In addition, MaSoN exposes the processed analytics data to Orchestration systems and OSSs so that they can process and take high level coordinated actions. This allows MaSoN to be involved as part of a holistic closed loop automation use cases that covers all networking domains other than 5G domain which involves other components like Orchestration Systems, OSS also.

MaSoN collects, stores and presents all the management data from various NFs and other components in the 5G network. It also supports aggregation, related data processing and reporting to the network operators. In addition to this, MaSoN supports analytics and ML capabilities for enhanced network management. These capabilities include detection of network performance degradation and anomalous network behaviour, prediction of future service quality, and resource optimization. This enables the network to take corrective actions to handle the system issues reported. MaSoN has an integrated ML Server component which supports training multiple ML Models and prediction of network problems and anomalies using these models.

B. ANALYTICS AND MACHINE LEARNING IN MaSoN

Analytics of network performance and proactive prediction of future network problems is achieved in MaSoN using a dedicated ML server framework to support such capabilities. This framework supports multiple analytics related use cases to be implemented over a common platform. This enables integration using a unified interface with the management system components in MaSoN. It supports considering recent trends in different types of data monitored and collected so that prediction, anomaly detection and optimization to be more accurate and corresponding self-corrective actions to be very effective.

The ML server component supports multiple implementations for ML use cases related to 5G networks. The classic supervised and unsupervised techniques for handling classification, anomaly detection, clustering and prediction applications are included. The ML server is deployed by integrating libraries including Scikit-learn [12], TensorFlow [13] and PyTorch [14].

The ML server supports initial training with and without labelled data that can be used in supervised and unsupervised machine learning models respectively. Initial training will be typically done offline during deployment. All prediction requests are processed based on the model built specifically for the different use cases implemented. The key aspect of ML server is that it supports tuning the model which is based on retraining the ML models more periodically considering recent data monitored and collected. This helps in models adapting to recent network behaviour where the retraining is based on incremental data thereby avoiding complete ML model training. This feature reduces time taken for retraining which helps in invoking retraining more frequently for more accurate results. This retraining will be typically done online.

The retrained model would be persisted so that it can be loaded quickly to satisfy all future prediction requests. However, the retrained model would be updated only when automatic testing and validation results do not indicate drastic difference in quality of result to avoid any unexpected behaviour in deployed production environment. The ML server supports persisting all recent requests received and response sent for future analysis. It also supports detailed

logging of all steps in execution for debugging and understanding the working of the implemented use cases.

The ML server supports flexible deployment options, where it is enabled with both cloud based deployment and on-premises deployment capabilities. It can be deployed using *containers* (e.g. docker) in a distributed manner, in the cloud and in the service provider enterprise system. The distributed nature of deployments helps in high availability and fault tolerance and also to achieve load balancing in terms of handling prediction requests. The ML server can be configured to be built for standalone deployment and also containerized (e.g. docker based) deployment.

To achieve all the above capabilities in ML server, the following systems were explored.

1) TENSORFLOW SERVING [15]

This framework, maintained by Google, supports deploying machine learning in production. It supports model versioning and also better performance in large production environments. It works best with Tensorflow ML models and is not very capable of supporting other ML library based implementations.

2) ACUMOS AI PLATFORM [16]

This framework that supports onboarding multiple ML models to server and it offers REST API for all client requests including prediction and onboarding. It supports cloud based and container based deployment; defining workflows and model catalogues. However, it supports only few ML libraries and it involves deeper learning curve since it has a lot of dependencies.

3) FLASK [17] BASED ML SERVER

Flask is a micro web framework written in Python that supports creating multiple ML models and served as a Flask application, and with Docker support. It supports flexible choice in using different ML libraries for implementation since a single ML library or a combination of ML libraries for different use cases can be adopted. It supports deployment in docker and cloud where REST API can be used for all client requests such as training, prediction and retraining. It supports flexibility in configuration of deployment parameters and also it is good for small to medium scale production.

Based on these factors and our initial implementation-based studies, the Flask based ML server option was chosen. It allows a use case to be implemented in the most efficient manner since all ML libraries can be easily integrated into the ML server. It also offers flexibility in configuring the ML server for a particular deployment.

IV. 5G ANALYTICS/ML USE CASES

This section describes three different 5G network use cases studied using the MaSoN system. The first two use cases have been briefly described in our earlier work [5]; in this paper, we describe detailed results for these use cases and a new use case.

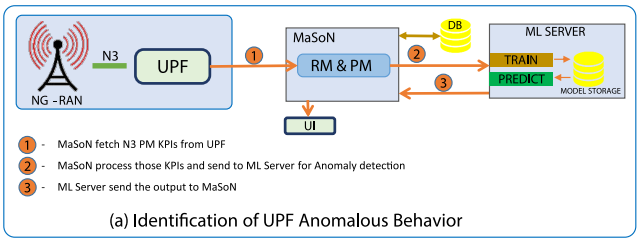


FIGURE 3. MaSoN components for UPF anomaly detection use case.

A. IDENTIFICATION OF UPF ANOMALOUS BEHAVIOUR

The first use case deals with the UPF that handles packet processing, including routing, forwarding, and per-flow Quality of service (QoS) handling. It serves as the interconnect between the NG-RAN and the external data network via the N3 interface. The UPF plays a major role in meeting the specified quality of service and quality of experience requirements of the users. Thus, traffic on the N3 interface must be monitored continuously to detect any anomalies in the UPF quickly.

In this use case, the PM Key Performance Indicators (KPIs) from N3 interface are monitored by MaSoN. The subset of components in the MaSoN architecture applicable for this use case is shown in Fig. 3. As shown, the RM and PM entities collect the measured data and store them in the database, that is subsequently used for training of ML algorithms.

Anomalous UPF behaviour is identified when a subset of the PM KPIs show abnormal change in the measured values. The list of UPF N3 interface KPIs (defined in 3GPP 28.552 [18]) selected for this use case, where packets correspond to the General Packet Radio Service (GPRS) Tunnelling Protocol (GTP), are: (i) Data volume of incoming GTP data packets per QoS level from RAN to UPF; (ii) Incoming GTP Data Packet Loss; (iii) Average Uplink (UL) GTP packets delay in UPF; and (iv) Link utilization.

The selection of the relevant KPIs is done such that their values have strong correlation and will also detect abnormal behaviour whenever there is sudden uncorrelated change. MaSoN performs continuous data collection for the above PM KPIs across all UPF instances and the latest set of collected data is sent to the ML server to identify whether the measured values represent any anomalous behaviour.

Anomaly detection techniques including eXtreme Gradient Boosting (XGBoost) [19] have been implemented in the ML server for the anomaly prediction. The XGBoost model implementation considered three QoS values, namely Guaranteed Bit Rate (GBR) typically used for conventional voice traffic, Non-GBR typically used for internet traffic/Over The Top (OTT) voice and video traffic and delay critical GBR type typically used for Automation related or latency sensitive traffic. The XGBoost model was created such that the QoS value along with the four UPF KPIs were considered as features of the model. Along with this, the UPF NF instance

identifier was also considered as a feature so that one model was able to cover multiple UPF instances and QoS values.

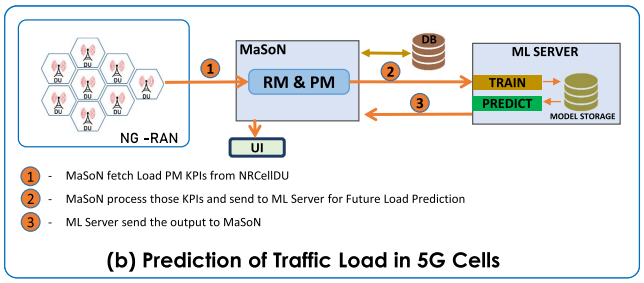


FIGURE 4. MaSoN components for cell-level load prediction.

B. PREDICTION OF TRAFFIC LOAD IN 5G CELLS

Continuous variations in traffic patterns in different 5G network segments is common due to different practical factors and hence efficient utilization of available 5G network resources is very critical. The access network is divided into multiple cells (and cells into sectors), with each cell (sector) served by a base station. Since the traffic load can vary dynamically, it possible that some cells are overloaded while others are under-utilized.

The objective of this use case is to predict the traffic load in near term and long term to provide suggestions for load balancing among cells and suggestions for cell splitting and/or merging. This is achieved by continuously monitor the traffic load in all cells. The total Physical Resource Block (PRB) Usage for Uplink and Downlink is continuously measured for each 5G cell to denote its traffic load. Based on this, prediction of future traffic load is implemented using ML techniques which are used to identify overloaded cells. The subset of components in the MaSoN architecture applicable for this use case is shown in Fig. 4. As seen, the RM and PM entities of MaSoN collect data from the NG-RAN on a continuous basis and feed the data to the ML server.

Three different ML techniques have been considered, namely Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) and Linear Regression (LR). Let Δt denote one measurement duration (PM KPI granularity), taken to be 15 minutes in this study. The following features were used for training the ML models: (i) Average data rate of the previous two, four and eight Δt intervals; (ii) Percentage of the data rate difference between the last two Δt intervals; between the $(t - \Delta t)$ and $(t - 3\Delta t)$ intervals; and between the $(t - \Delta t)$ and $(t - 4\Delta t)$ intervals.

C. CLOSED LOOP AUTOMATION FOR SMF RESOURCE USAGE

The SMF in 5G core is responsible for interacting with the decoupled data plane, creating, updating and removing Protocol Data Unit (PDU) sessions and managing session context with the UPF. SMF would be continuously handling PDU sessions and QoS flows associated with them. This is carried out for each slice supported by the SMF instances. SMF is typically deployed as virtualized network function where the SMF handling capacity in terms of PDU sessions and QoS flows directly depends on virtualized Central Processing Unit (CPU) and memory resources available.

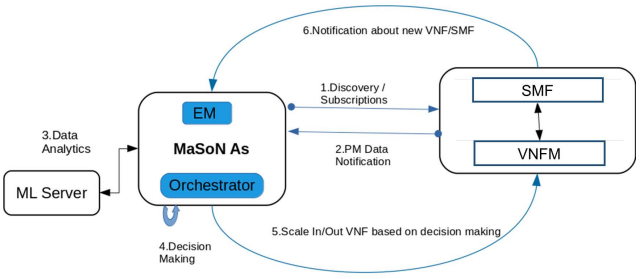


FIGURE 5. Closed loop automation for SMF resource usage.

The objective of this use case is to implement closed loop automation capability in the management of 5G networks. This is done by continuously monitoring the SMF resource utilization, predicting future resource utilization based on recent trend and triggering VNF instantiation of additional SMF instances to overcome high resource utilization. This is achieved by spawning new SMF instances which are triggered ahead of time to handle recent trend of high CPU or memory resource usage in currently available SMF instances resulting in seamless network behavior. This is implemented using machine learning techniques for resource usage prediction and intelligent decision making for instantiating new SMF instances to overcome recent high resource usage.

This is depicted in Fig. 5 where MaSoN system functions as both the EM layer system and the NFV Orchestrator as specified in the ETSI NFV-MANO architecture [2]. As an EM layer system, MaSoN subscribes to continuous collection and processing of SMF-wise slice level performance management parameters and KPIs along with the corresponding VNF level CPU and memory resource utilization measurements. The ML Server integrated with MaSoN system implements the machine learning algorithms for predicting future resource utilization by considering multiple relevant performance parameter values. The MaSoN system performs the decision making logic to determine whether new VNF instances have to be instantiated based on prediction output. When it decides to spawn new VNF instances, it functions as the NFV Orchestrator by instantiating new VNF instances using ETSI defined interfaces. The newly instantiated VNF instances are expected to reduce high CPU and memory utilization by means of load sharing thus achieving closed loop functionality. The newly instantiated VNF instances are considered for subsequent performance data collection and decision making and this process follows continuously.

The implementation considers recent patterns in the slice level number of PDU session creation requests received, percentage of such requests which were successfully processed and based on this predict the future CPU and memory

resource usage considering the recent CPU and memory usage at VNF level for SMF instances. In order to enhance the prediction output, factors such as the number of QoS flow creation requests received and the percentage of the successful QoS flow requests processed is also considered.

The following features are considered, where each data row value was generated by considering the data values in the previous time slots. The variables are: (i) the average total number and variance of PDU session creation requests over a specific duration (12, 24, and 48 hours); (ii) the average total number and variance of *successful* PDU session creation requests over a specific duration (12, 24, and 48 hours); (iii) the percentage of *successful* PDU session creation requests over a specific duration (12, 24, and 48 hours). Similar variables have been defined with respect to the number of QoS Flow Creation requests and the number of successful QoS Flow requests.

The prediction is considered as two sub-problems, namely regression problem and time-series problem. The regression problem aims to predict the CPU and memory utilization in the next time instants based on the recent SMF-wise and slice-wise performance measurements. The time-series problem aims to predict the CPU and memory utilization for multiple future time instances based on the CPU and memory utilization prediction output from the regression problem along with the recent CPU and memory utilization measurements. The future prediction of 10 CPU and memory utilization instances is performed. If a new SMF VNF instance needs to be instantiated due to high utilization, it could be done more proactively ahead of time to avoid over-utilization of CPU and memory and hence any possible traffic disruption.

In the regression problem, different models instances are considered, where model instances correspond to the slices configured in the network. This had been done to do slice-specific future resource usage prediction based on the slice's trend in measured resource utilization. The model training was done based on measured resource usage and the prediction output was validated with the test data.

In the time-series prediction, a single model instance for all SMFs supporting different slice instances was implemented. Here, the history of measured CPU and memory resource usage data was considered for the model training. As part of prediction, 100 recent CPU and memory resource utilization values are considered to predict the next 10 CPU and memory resource utilization values. Out of the 100 recent values, the most recent value is taken as the prediction output from the regression problem.

As part of the implementation for this use case, the random forest ML technique has been considered. The random forest implementation predicts the next 10 CPU and memory resource utilization values by classifying them into 10 classes which range from 0 to 10, 10 to 20 up to 90 to 100 since this resulted in better prediction output which took lesser time for execution. Since the regression problem has many features and related data, deep learning based techniques such as LSTM were considered to be more suitable.

For the time-series prediction problem, Auto Regression (AR) and Vector Auto Regression (VAR) were attempted but their output performance and the prediction output quality were found to be very moderate. After that, Auto Regressive Integrated Moving Average (ARIMA) was also attempted but its prediction output quality was also found to be very moderate. AR and ARIMA are uni-variate time-series prediction models and VAR is a multi-variate time-series prediction models. For all of these models, the next 10 prediction output values were more or less same with very less variations. LSTM for the time-series prediction problem was also attempted but it also did not give expected prediction output.

The decision making MaSoN logic considers the next 10 prediction output values to decided whether any new SMF VNF need to be instantiated. From the next 10 output values corresponding to next 10 time instants, count of time instants where either CPU or memory utilization values exceed a threshold is determined. As part of the implementation and evaluation, threshold of 80% is considered. If the count of time instants for which threshold gets exceeded is more than 5 out of the 10 time instants, then the MaSoN system decides to instantiate a new SMF VNF for balancing the load on existing available SMF instances proactively to avoid future traffic disruption. The threshold value and the count expected for instantiating new SMF VNF were chosen to avoid unnecessary instantiation of new VNF instances due to momentary fluctuations in CPU and memory utilization.

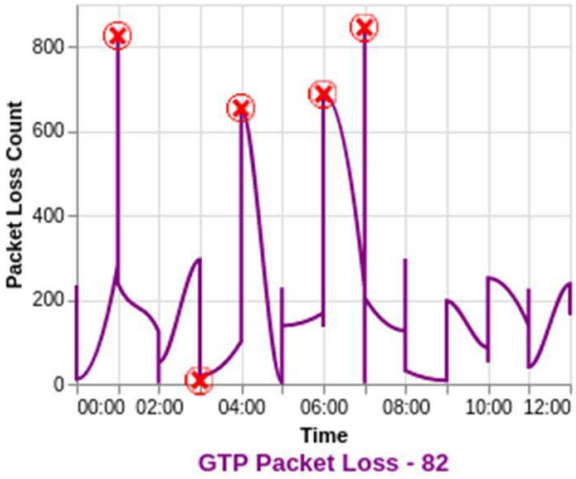
This implementation is aimed at proactively predicting possible high resource utilization in the future when there is recent continuous high CPU and memory usage in the available SMF instances. This will enable new SMF instance instantiation ahead of the time automatically to avoid probable resource exhaustion or resource utilization reaching critical levels which affects SMF functioning. This would help in avoiding service traffic disruption or degradation thereby improving network operations. This implementation is an important aspect in building autonomous network management and operations in 5G networks.

V. PERFORMANCE RESULTS

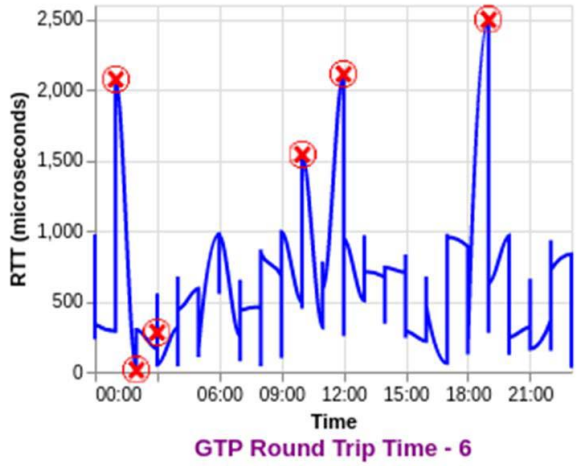
The performance of the described model implementations for the three use cases are evaluated and the results obtained are provided in this section. The metrics used in the performance evaluation are: F1 score for the first use case and Mean absolute error (MAE), Mean absolute percentage error (MAPE) and Mean squared error (MSE) for the second use case and MAE for the third use case, as defined in [20]. The proposed system is available for a trial demonstrator at [6].

A. IDENTIFICATION OF UPF ANOMALOUS BEHAVIOUR

For testing the implemented XGBoost model for identification of UPF anomalous behaviour, four different synthetic data sets were generated. Each set contained UPF PM KPI data entries with a granularity of 15 minutes for a total of 90 days, out of which some percentage of entries correspond to anomalous data. This resulted in a total of approximately



(a) Packet Loss Anomaly Count



(b) Delay or RTT Anomaly Count

FIGURE 6. Sample UPF anomaly detection output (screenshots taken from MaSoN UI).

25,000 data entries for each UPF instance considered during synthetic data generation. The 4 sets of synthetic data were generated such that every data set contained 8 different UPF instances across 2 networks where Network-1 contained 5 UPF instances and Network-2 contained 3 UPF instances. The 4 data sets were generated such that they contained 10%, 15%, 20% and 30% anomalous data entries for each UPF instance considered during synthetic data generation.

The total set of data entries was split into training data that contained 80% entries (includes anomalies in the different proportion mentioned above) and test data that contained 20% entries (includes anomalies in the different proportion mentioned above). Considering an N3 interface speed of 100 Gbps having 1% packet loss and 50 ms round trip time, each entry contains data for the 4 PM KPIs. The range of values used for data generation for typical values were 0 to 80 Gbps for data volume, 0 to 10,000 packets for packet loss, 5 to 50 milliseconds for round trip time and 0 to 80 for link utilization. Any value that exceeds the maximum value in the above ranges was considered as abnormal.

The synthetic data was generated for GBR, Non-GBR and and delay critical GBR type, as explained earlier. For evaluating the XGBoost based anomaly detection approach implemented, the F1 score (ranges from 0 to 1) was measured. From the results obtained, it was observed that the XGBoost approach resulted in the maximum F1 score, which indicates best performance. This indicates that all the anomalies identified were True Positives (TP). For the test data evaluated, XGBoost approach was able to identify all the abnormal entries accurately. For comparison purposes, we implemented the Multi Variate Gaussian Distribution (MVGD); this resulted in a very low F1 score (less than 0.2) compared to XGBoost, demonstrating the enhanced effectiveness of XGBoost. Sample output graphs depicting the anomaly counts for a particular UPF instance and QoS value for one day are shown in Fig. 6. Here, the spikes having abnormally high values indicate the anomalies detected by the implemented system. Spikes indicated as anomalies are displayed in all the four graphs for PM KPIs to convey how one particular PM KPI shows abnormally high values even though other 3 PM KPIs do not have such abnormality. The low values indicated as abnormal values for KPIs such as Packet Loss or Round Trip Time are actually due to abnormally high values in some other related KPIs from the 4 KPIs considered which are Data Volume, Packet Loss, Round Trip Time and Link Utilization.

B. PREDICTION OF TRAFFIC LOAD IN 5G CELLS

As mentioned earlier, three ML models were considered, namely, RNN, LSTM and LR. For both RNN and LSTM model training, the number of epochs used is 10. Even when the number of epochs is increased to 50, 100 and 500, very significantly less difference in loss rate was observed as part of performance evaluation, as shown later. The number of hidden layers was varied as 20 and 50; the activation functions studied were *relu* and *sigmoid*; the optimizers used were *adam* and *adamax*.

For testing the implemented approaches, data was generated using random number generators following different distributions to simulate different load variations. Synthetic data for Downlink and Uplink load (as percentage utilization value that indicates PRB usage for each cell in a 5G network, was generated. The initial load is generated as a random value between 0 and 100. The load value was incremented or decremented with a small change (Δ) from the previous load value. To ensure randomness in Δ, a weighted delta value was generated so that normal, sudden high and low load changes are accommodated. Four different weights (0.6, 0.25, 0.1 and 0.05) were considered. For each weight, different minimum and maximum delta load values were considered as part of generating random delta load value within the considered range. The range for delta values corresponding to the four weights are 0 to 5, 0 to 10, 0 to 20 and 0 to 25 respectively. Delta or change value at each time step is either increment or decrement chosen randomly from the previous value. The synthetic data was generated such that

it contained 43 different NRCellDU instances across two networks. Network-1 contained 27 NRCellDU instances and Network-2 contained 16 NRCellDU instances. The synthetic data generated contained NRCellDU PM KPI data entries with a granularity of 15 minutes for a total of 90 days. This resulted in a total of approximately 17,000 data entries for each NRCellDU instance considered during synthetic data generation. The total data set was split into training data (80% entries) and test data (20%).

TABLE 2. Comparison of performance measures for load prediction. For the model parameters, (20, R, AM) denotes (20 layers, relu activation function, and adam optimizer); also, S and AX denote *sigmoid* and *adamax* respectively. For each ML model, the two smallest values for RNN and LSTM are shown in color, in each column.

Model	Parameters	MAE	MAPE	MSE
RNN	(20, R, AM)	17.7894	2.4440	828.1244
	(20, R, AX)	17.8897	2.4829	541.9406
	(20, S, AM)	25.7015	3.8997	885.0872
	(20, S, AX)	25.5824	4.0983	877.3069
	(50, R, AM)	18.5583	2.6435	709.2011
	(50, R, AX)	18.3474	2.4360	792.4941
	(50, S, AM)	25.7153	3.9035	886.8465
	(50, S, AX)	25.5813	4.0978	876.8651
LSTM	(20, R, AM)	17.7709	2.4544	480.6453
	(20, R, AX)	17.8967	2.4431	693.7534
	(20, S, AM)	25.7382	3.8624	888.1511
	(20, S, AX)	25.5816	4.0882	876.9000
	(50, R, AM)	18.9260	2.5900	537.8674
	(50, R, AX)	18.2297	2.5493	532.2069
	(50, S, AM)	25.7074	3.9050	885.7953
	(50, S, AX)	25.5840	4.0886	876.8872
LR	-	17.5036	2.4057	1122.7104

For evaluating the load prediction approach implemented, MAE, MAPE and MSE were measured for RNN, LSTM and LR. The values obtained for MAE, MAPE and MSE for each ML model and the applicable model parameters combinations are provided in Table 2. It can be observed that for RNN and LSTM, the *relu* activation function results in lower MAE, MAPE and MSE values compared to sigmoid. Apart from this, the results with 20 and 50 hidden layers and the results for optimizers *adam* and *adamax* do not show much difference in MAE, MAPE and MSE values. With respect to LR, the MAE and MAPE values almost match with RNN and LSTM values whereas MSE for LR is found to be higher compared to that of RNN and LSTM.

This can be further observed from the MAE comparison bar graph shown in Fig. 7 for RNN and LSTM model outputs, with the number of hidden layers as 20 and for different combinations of activation function and optimizer values. A sample screenshot from the MaSoN application client is shown in Fig. 8 which shows the dashboard view for the count of overloaded cells across two 5G networks for the last 30 days. This dashboard output is based on the implemented ML model for prediction of overloaded cells considering UL and DL traffic. The bottom right graph shows the actual and predicted traffic for one cell for one particular day from which it can be observed that the predicted load closely matches the

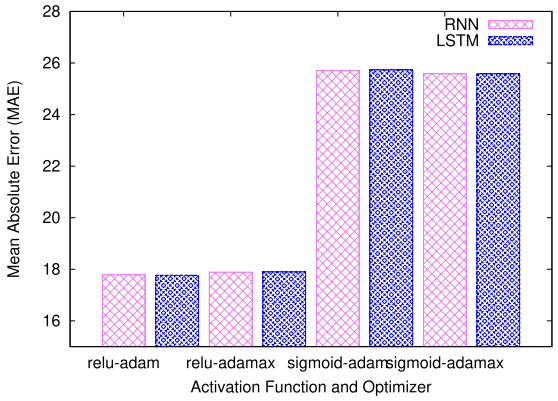


FIGURE 7. Results for load prediction use case.

actual load indicating the correct working of the implemented use case.

From the above analysis, it can be observed that the implemented approach was able to correctly identify all overloaded cells. This would help in proactively configuring the network for handling the future traffic load. The traffic load data would be continuously processed for providing the possible suggestions so that the network is effectively utilized with the available resources.

TABLE 3. SMF usage MMPP state-wise mean value for PDU session creation requests.

MMPP State	Mean Value
Normal	20% of Base Value
High	400% of Base Value
Low	–100% of Base Value

C. CLOSED LOOP AUTOMATION FOR SMF RESOURCE USAGE

For evaluating the performance of SMF resource usage, number of PDU session creation requests and number of QoS flows are considered for each SMF. For PDU session creation requests, mean total number of PDU session requests received and percentage of successfully processed PDU session requests have been considered all for a couple of previous intervals. Similarly, mean total number of QoS flow requests received and percentage of successfully QoS flow requests have been considered for a couple of previous intervals. Along with this, variance in the total number of PDU session requests and QoS flow requests received and percentage of successfully processed PDU session requests and QoS flow requests have also been considered individually for performance evaluation. For all the evaluations, the corresponding CPU and Memory usage are also considered as features. The duration considered in all cases are the last 12 hours, 24 hours and 48 hours.

Synthetic data for total number of PDU session creation requests for each SMF in a 5G network was generated. Here, the total number is incremented or decremented with a delta

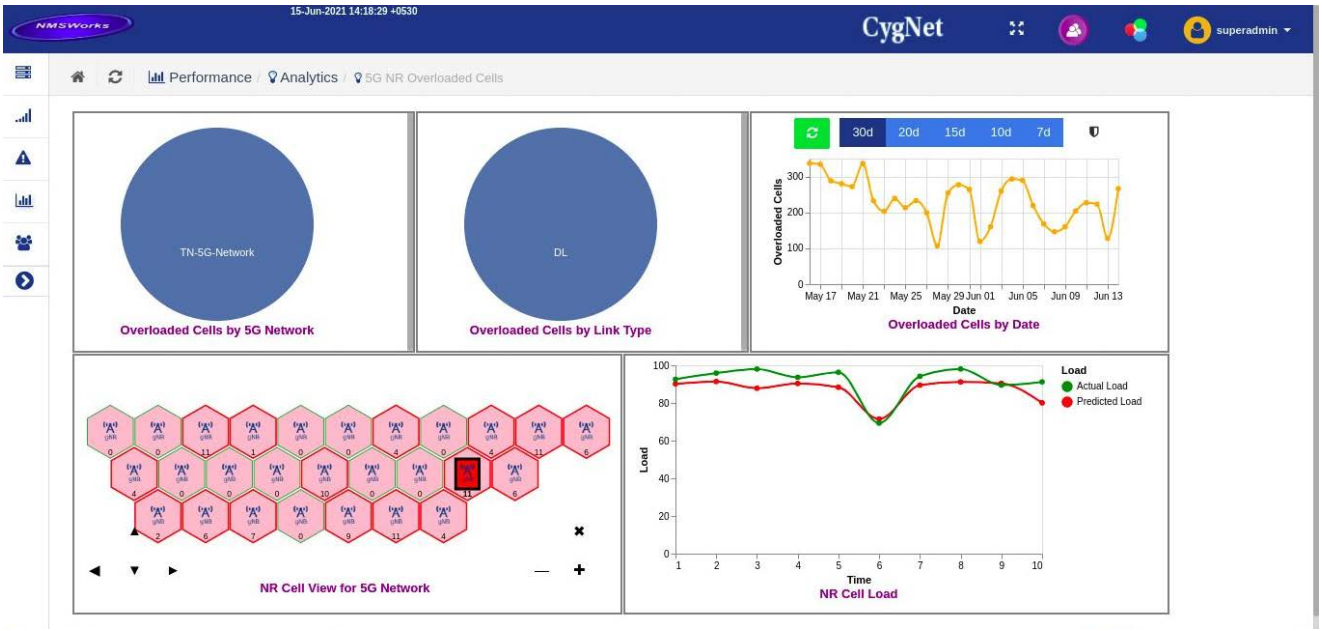


FIGURE 8. Load prediction sample screenshot.

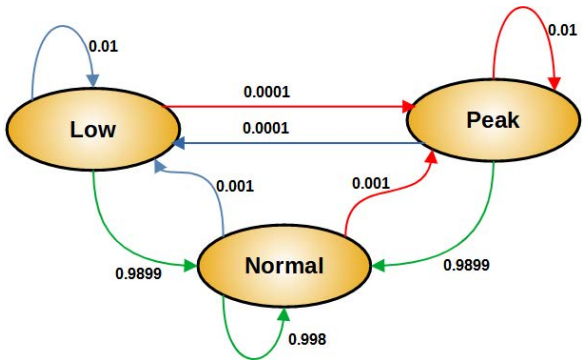


FIGURE 9. Markov-modulated poisson process (MMPP) state transitions.

value from the previous number. To generate synthetic data such that randomness in delta value is present, a delta value is generated so that normal, sudden high resource usage and low resource usage are accommodated. For successfully processed PDU session creation requests, a random percentage of requests have been considered as failed, where a maximum of 10% of the total number of requests may be failed. A similar approach was followed for QoS flow requests, where the total number of QoS flow requests for each PDU session is a random value within a range having defined maximum and minimum values. For CPU and memory usage values also, random values following different ranges for normal, sudden high resource usage and low resource usage have been considered.

Synthetic data generation that represents a real-life scenario has to consider gradual changes in the total number of PDU session creation requests and also sudden high and low

TABLE 4. SMF usage prediction regression problem LSTM parameters.

Parameter	Value
No. of Hidden Layers	20
Activation Function	leakyrelu
Loss Parameter	MAE
Optimizer	Adam
No. of Epochs	250 and 500

TABLE 5. SMF usage time-series prediction problem random forest parameters.

Parameter	Value
No. of Estimators	50 and 100
Min. Samples Split	2, 5 and 10
Max. Features	auto, *sqrt* and *log2*

TABLE 6. Performance measures for SMF usage LSTM regression problem.

No. of Epochs	MAE Value
250	2.9453
500	2.5445

changes in them to reflect unexpected deviations in traffic requests which are typically less probable. This is implemented as a sequence of steps that are carried out continuously at the end of each measurement period:

1) Generate the slice-wise count of PDU session requests as a MMPP considering all slices across all available SMF instances
2) Introduce change in the count based on increment or decrement of a delta change value from a Uniform Random Distribution

TABLE 7. Performance measures for SMF usage random forest time-series problem.

No. of Estimators	Min. Samples Split	Max. Features	CPU Accuracy	CPU F1	Memory Accuracy	Memory F1
50	2	auto	0.9520	0.9649	0.9484	0.9598
50	2	sqrt	0.9536	0.9656	0.9477	0.9597
50	2	log2	0.9518	0.9647	0.9462	0.9583
50	5	auto	0.9521	0.9643	0.9460	0.9581
50	5	sqrt	0.9514	0.9636	0.9471	0.9591
50	5	log2	0.9519	0.9646	0.9482	0.9591
50	10	auto	0.9464	0.9594	0.9396	0.9514
50	10	sqrt	0.9471	0.9600	0.9406	0.9527
50	10	log2	0.9464	0.9591	0.9387	0.9514
100	2	auto	0.9523	0.9645	0.9491	0.9603
100	2	sqrt	0.9527	0.9655	0.9480	0.9596
100	2	log2	0.9537	0.9653	0.9491	0.9601
100	5	auto	0.9521	0.9649	0.9476	0.9595
100	5	sqrt	0.9528	0.9644	0.9480	0.9594
100	5	log2	0.9533	0.9654	0.9483	0.9600
100	10	auto	0.9489	0.9616	0.9416	0.9529
100	10	sqrt	0.9483	0.9610	0.9400	0.9522
100	10	log2	0.9486	0.9613	0.9399	0.9521

3) Consolidate the counts slice-wise and split the resulting total value across different slices in all available SMF instances where the split is done with small variations around average value to avoid even split across all slices
4) Calculate other KPIs such as count of successful PDU session requests, total and successful QoS flow requests based on the value obtained after split

The MMPP is implemented as a system with three states representing normal, high and low creation of traffic requests. For each of the three states, different values for mean distribution of PDU session creation requests have been defined. Also, state transition probabilities for all combinations of transitions from one state to another state have been defined to reflect real-life scenario of normal and sometimes unexpected changes in traffic requests. This is depicted in Fig. 9, where the three states along with their mean value for PDU session requests and state transition probabilities for all possible state transitions have been mentioned.

The MMPP mean value is used to get delta or difference in PDU session creation request value which is applied to the previous value to get the current PDU session creation requests value. This is the used as the basis for all further calculation in each time step as part of data generation. The different MMPP mean values considered for the three states namely normal, high and low are specified in Table 3. The Base Value considered for the entire data generation was configured as 200. To provide for random variations from the MMPP mean value in each value generation at every time instant, a random value based on uniform distribution in the range -12.5 to 37.5 which is the percentage of MMPP mean value obtained is applied.

To follow the trend of PDU session requests variations, CPU and memory usage variations have to consider the delta changes in PDU session requests. From the PDU session creation requests value, SMF-wise CPU and Memory utilization values are calculated based on the delta PDU session requests value obtained by means of scaling it with a variation factor of 10%. This results in avoiding similar CPU and memory utilization usage across different slices handled in the SMF instances.

For performance evaluation, the LSTM model was used for the regression problem and the Random Forest model was used for time-series prediction where both the model parameters considered along with their values used are listed in Table 4 and Table 5.

For this study, synthetic data was generated for 3 SMF instances and a total of 5 Slice Ids across the 3 SMF instances were considered. The generated synthetic data contained SMF PM KPI data entries with a granularity of 15 minutes for a total of 100 days. This data was processed such that one data entry is generated for each hour to facilitate considering last 12 hours, 24 hours and 48 hours for predicting future SMF resource usage based on that. This resulted in a total of approximately 2,500 data entries for each SMF instance considered during synthetic data generation. The total set of data entries were split into training data which contained 80% entries and test data which contained the remaining 20% entries used for evaluating the performance of load prediction. For the LSTM model training, the number of epochs used was 250 and 500. The MAE values obtained for the number of epochs 250 and 500 in LSTM model are presented in Table 6. The performance results obtained which are Accuracy and F1 score for CPU and Memort usage for the Random Forest model for the different combinations of model parameters are presented in Table 7.

From the results obtained, it can be observed that for the LSTM regression, the number of epochs 500 resulted in lower MAE compared to the number of epochs 250. With respect to the time-series prediction, it can be observed that there is no significant difference in CPU and Memory accuracy and F1 score values across different combinations of number of

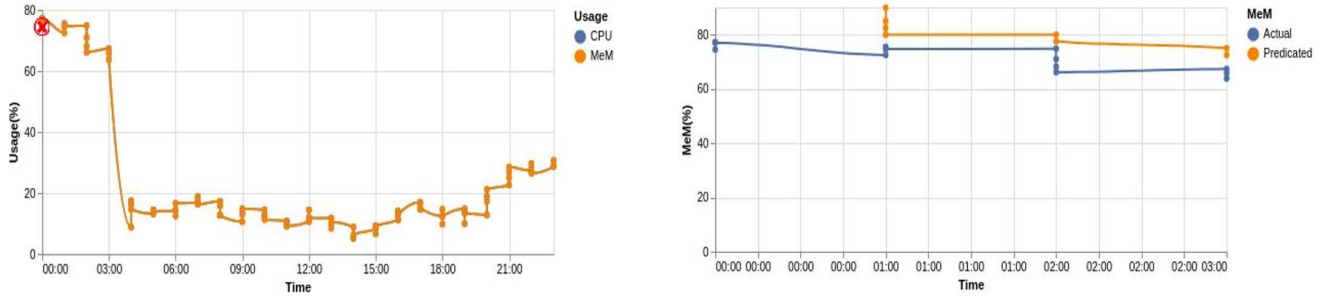


FIGURE 10. SMF resource usage prediction sample.

estimators, minimum samples split and maximum features values.

The described approach has been implemented in a test MaSoN system and a simulator was used to generate data as per the details specified. On observing the system behaviour for few weeks, it was noticeable that the MaSoN system instantiated new SMF instance(s) whenever high resource usage was observed based on recent trend in CPU and memory usage. Further, after every new SMF instantiation, the resource usage reduced significantly in the available SMF instances confirming the effectiveness of the implemented approach. This can be observed from the prediction output sample snapshot from the client of the implemented system presented in Fig. 10. The cross-mark symbol in the left side graph indicates the time instant in which new SMF instantiation was triggered. This snapshot clearly shows that the SMF resource utilization decreased significantly after new SMF instantiation and the predicted SMF resource usage closely matches the actual measured resource usage.

VI. CONCLUSION

This work describes the design aspects of the implemented management system enabled with analytics and machine learning capabilities to support autonomous networking capabilities in the management of 5G networks. The architecture of the system with its components and details of the three use cases implemented are also described. The use cases implemented are identification of UPF anomalous behaviour, prediction of traffic load in 5G cells and prediction of SMF resource usage. For the SMF resource usage use case, closed loop automation functionality has been implemented and it was observed to be effective in automatically instantiating SMF instances by prediction of high resource usage based on recent trend in resource usage. This helps to overcome or avoid potential future traffic disruptions proactively without any manual intervention. The system is designed to be flexible and can be extended for other use cases in a 5G network.

ACKNOWLEDGMENT

The authors thank Babu Narayanan and the 5G Testbed Team at CEWiT, Chennai, for their valuable inputs regarding analytics use cases using machine learning in 5G networks. CygNet is the brand of OSS/NMS products from NMSWorks Software.

REFERENCES

[1] *Specification Set: 5G*, document, 3GPP, 2019. [Online]. Available: https://www.3gpp.org/dynareport/SpecList.htm?release=Rel-15%26tech=4

[2] *Network Functions Virtualisation (NFV); Management and Orchestration*, document ETSI GS NFV-MAN 001, Version 1.1.1, 2016. [Online]. Available: https://www.etsi.org/technologies/nfv

[3] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 1, pp. 1–99, 2018.

[4] K. M. Sivalingam, "Applications of artificial intelligence, machine learning and related techniques for computer networking systems," 2021, *arXiv:2105.15103*.

[5] T. Archana, V. Deepika, A. A. Kumar, M. Ramachandran, K. M. Sivalingam, and K. J. B. Narayanan, "Demo—CygNet MaSoN: Analytics and machine learning enabled management system for 5G networks," in *Proc. 13th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2021, pp. 94–96.

[6] NMSWorks Software Private Limited. (Apr. 2022). *MaSoN Demo Application*. [Online]. Available: https://sec.nmsworks.co.in:1995/mason

[7] *Architectural Framework for Machine Learning in Future Networks Including IMT-2020*, document ITU-T Recommendation Y.3172, Jun. 2019.

[8] (2020). *ETSI Experiential Networked Intelligence*. [Online]. Available: https://www.etsi.org/technologies/experiential-networked-intelligence

[9] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, 1st Quart., 2018.

[10] (2021). *Open Source MANO (OSM)*. [Online]. Available: https://osm.etsi org/

[11] (2021). *Open Network Automation Platform (ONAP)*. [Online]. Available: https://www.onap.org/

[12] (2021). *Scikit-Learn: Machine Learning in Python*. [Online]. Available: https://scikit-learn.org/

[13] (2021). *TensorFlow: An End-to-End Open Source Machine Learning Platform*. [Online]. Available: https://www.tensorflow.org/

[14] (2021). *PyTorch: An Open Source Machine Learning Framework*. [Online]. Available: https://pytorch.org/

[15] (2021). *TensorFlow Serving*. [Online]. Available: https://www.tensorflow.org/tfx/guide/serving/

[16] (2021). *AcumosAI*. [Online]. Available: https://www.acumos.org/

[17] (2021). *Flask*. [Online]. Available: https://palletsprojects.com/p/flask/

[18] *Management and Orchestration; 5G Performance Measurements*, document TS 28.552, 3GPP, 2019. [Online]. Available: https: //portal.3gpp.org/desktopmodules/Specifications/SpecificationDetailsaspx?specificationId=3413

[19] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[20] A. Botchkarev, "Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology," 2018, *arXiv:1809.03006*.

MADANAGOPAL RAMACHANDRAN received the B.E. degree in computer science and engineering from Bharathiar University, Coimbatore, India, in 2002, and the M.S. and Ph.D. degrees in computer science and engineering from IIT Madras, Chennai, India, in 2008 and 2020, respectively.

He is currently a Senior Technical Architect with NMSWorks Software Pvt. Ltd., Chennai. He is also leading the design and development of a product for 5G network analytics, orchestration, and management. His research interests include service provisioning in transport networks, data analytics, autonomous networking, and orchestration for 5G networks, algorithms, and distributed systems.

T. ARCHANA received the B.E. degree in computer science and engineering from the Madras Institute of Technology, Chennai, India, in 2014. She is currently working as a Senior Software Engineer with NMSWorks Software Pvt. Ltd., Chennai, for the past eight years. She has spent the past two years working on 5G network management and analytics application development. Her research interests include machine learning, networking, and big data technologies like apache hadoop and apache spark.

V. DEEPIKA received the B.Tech. degree in information technology from the Madras Institute of Technology Campus, Anna University, Chennai, India, in 2019. She is currently working as a Software Engineer with NMSWorks Software Pvt. Ltd., Chennai, with two plus years of industry experience in back-end development in core java, networking, and machine learning. For the past two years, she is working in 5G network management and analytics application development.

A. ARJUN KUMAR received the B.Tech. degree in computer science and engineering from SASTRA University, Thanjavur, India, in 2019. He is currently working as a Software Engineer with NMSWorks Software Pvt. Ltd., Chennai, India, with two years of industry experience in data analytics, machine learning, and web development.

KRISHNA M. SIVALINGAM (Fellow, IEEE) received the B.E. degree from the College of Engineering, Anna University, Chennai, India, and the M.S. and Ph.D. degrees from SUNY Buffalo, USA. He had held faculty appointments at the University of Maryland, Washington State University, and the University of North Carolina Greensboro. He is an Institute Chair Professor with the Department of CSE, IIT Madras, Chennai. He is a fellow of INAE and an ACM Distinguished Scientist.