

Received 28 November 2025, accepted 16 December 2025, date of publication 24 December 2025,  
date of current version 7 January 2026.

Digital Object Identifier 10.1109/ACCESS.2025.3648065

## RESEARCH ARTICLE

# Cloud-Native Multivariate Probabilistic Forecasting for Next-Generation Resource Allocation in O-RAN

VAISHNAVI KASULURU<sup>1</sup>, LUIS BLANCO<sup>1</sup>, CRISTIAN J. VACA-RUBIO<sup>1</sup>, (Member, IEEE),  
ENGIN ZEYDAN<sup>1</sup>, (Senior Member, IEEE),  
ANGELOS ANTONOPOULOS<sup>2</sup>, (Senior Member, IEEE),  
AND ALBERT BEL<sup>1</sup>

<sup>1</sup>Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), 08860 Castelldefels, Spain

<sup>2</sup>Nearby Computing S.L., 08006 Barcelona, Spain

Corresponding authors: Vaishnavi Kasuluru (kasuluru.vaishnavi@upc.edu) and Luis Blanco (lblanco@cttc.es)

This work was supported in part by Spanish MINECO Program UNICO I+D under Grant TSI-063000-2021-54 and Grant TSI-063000-2021-55; in part by MINECO through the “NextGenerationEU” Program (FREE6G-RadEdge) under Grant TSI-063000-2021-121; in part by the UNITY-6G Project funded by European Union’s Horizon Europe Smart Networks and Services Joint Undertaking (SNS JU) Research and Innovation Program under Grant 101192650; in part by the Swiss State Secretariat for Education, Research and Innovation (SERI); and in part by European Union’s Horizon 2020 Research and Innovation Program through the Marie Skłodowska-Curie under Grant 861116.

**ABSTRACT** Effective resource allocation in telecommunication networks can benefit from accurate demand forecasting to optimize performance and prevent inefficiencies such as resource over-provisioning or shortages. Traditional forecasting approaches often fail to capture uncertainty and multivariate interdependencies. This frequently results in suboptimal decision-making, particularly in dynamic, multi-tenant environments such as Open Radio Access Networks (O-RAN). To mitigate this issue, multivariate probabilistic forecasting provides a more robust approach by capturing the interdependencies among multiple resource time series, including but not limited to PRB, and by providing confidence intervals for predictions, enabling more adaptive and reliable resource management in O-RAN. This paper proposes a cloud-native resource allocation framework for O-RAN, integrating advanced probabilistic forecasting models within RAN Intelligent Controllers (RICs). We implement and evaluate Gaussian Process Vector Autoregression (GPVAR) and Temporal Fusion Transformer (TFT), comparing their performances against multivariate Long Short-Term Memory (LSTM) networks. The proposed solutions are deployed as a containerized radio application (rApp) and integrated with Swagger’s REST API to facilitate network exposure & seamless deployment within the O-RAN framework. Through extensive experiments, we demonstrate that for longer time series, TFT provides accurate and reliable forecasts, particularly in dynamic multi-tenant scenarios. In contrast, GPVAR offers better performance and a strong balance between accuracy and computational efficiency for shorter time series. Additionally, we investigate low-rank approximations in GPVAR, showing that mid-range rank configurations optimize computational complexity while maintaining predictive performance. We also analyze the computational trade-offs, showing that TFT incurs higher training costs but offers faster inference and lower resource overhead compared to GPVAR and LSTM. By bridging theoretical research with practical O-RAN deployment, this work provides a robust foundation for AI-driven resource management in next-generation networks.

**INDEX TERMS** Open RAN, 6G, probabilistic multivariate forecasting, cloud native, swagger.

The associate editor coordinating the review of this manuscript and approving it for publication was Abbas Kiani<sup>1</sup>.

## I. INTRODUCTION

The rapid evolution of wireless communication technologies, mainly through Open Radio Access Network (O-RAN), has

redefined the development, deployment, and optimization of telecommunication networks [1]. Unlike traditional and proprietary Radio Access Network (RAN), depending solely on hardware, O-RANs open, interoperable, virtualized, and disaggregated framework stimulates flexibility, scalability, and vendor-neutral advancements [2]. Diverse applications are expected to demand dynamic and adaptive communication that is ultra-reliable, has low latency, and has high bandwidth [3]. Edge devices, if constrained by limited resources, struggle to meet these required performances [3]. In O-RAN environments, where dynamic, real-time radio and computation resource allocation are crucial, this problem becomes more definite [4]. Efficient resource and energy management is critical in dynamic networks [5], [6].

Radio resources such as Physical Resource Block (PRB)s are commonly considered important for facilitating effective radio spectrum allocation [4]. Resource allocation can be efficiently modeled as a time series forecasting problem, where resource usage patterns in the past are analyzed to predict future demands [7]. Various research and ongoing studies on resource allocation in evolved mobile networks use traditional forecasting approaches that are challenging when dealing with dynamic and distributed network requirements [8], [9]. Moreover, most papers only consider univariate traditional time series models to analyze a single variable over time and predict that single resource as a function of its past demand [7], [10]. In contrast, considering multivariate time series improves the understanding of network requirements and enables effective resource provisioning by capturing complex correlations and dependencies between multiple related variables [7]. Practical estimation and provisioning of resources require a comprehensive understanding of the trade-off between computational cost, energy efficiency, and performance [5], [6].

Traditional single-point estimators such as Long-Short Term Memory (LSTM) lead to suboptimal results as they fail to capture trends and interdependencies in network performance. LSTM is used as a strong baseline due to its widespread adoption and established reputation. In multivariate use cases, LSTM does not account for uncertainty and affects the confidence of service providers and researchers in their decisions due to single estimates of possible future demands [11]. Probability-based forecasting effectively addresses the above challenges by incorporating uncertainty into predictions, providing a wide range of possible results to make reliable and robust decisions regarding the required resources, and improving prediction performance [6], [12]. Additionally, the distribution of radio resources is adjusted based on predictions to adapt to changing resource needs, forecasts, and service requirements. In the context of O-RAN, accurate forecasting of resources like PRBs is crucial to ensure reliability in service demands, fulfill SLAs and support dynamic multi-tenant environments. Currently, O-RAN intelligence is centralized and it requires processing of large volumes of time-series data generated by distributed

RAN components. This centralized intelligence layer is responsible for handling high-dimensional inputs related to various metrics, including radio, compute, and traffic load indicators across multiple cells and slices. Traditional univariate or low-dimensional models fall short in capturing such complex interdependencies. To address this, this work explores advanced multivariate and high-dimensional probabilistic forecasting techniques that are capable of modeling uncertainty, handling the non-stationarity of the data and evolving patterns, and scaling with the demands of next-generation RANs [13]. Moreover, compelling predictions help achieve optimal resource provisioning and reduced resource waste, regardless of changing network requirements [14].

Scalability, optimizations, and quick provisioning gain significant advancements through the use of cloud-native solutions within the O-RAN architecture, facilitated by the dynamic management and orchestration of network functions [15]. Cloud-native strategies encompass containers, Continuous Integration/Continuous Deployment (CI/CD) pipelines, and other microservices, enabling faster service installation and seamless integration of third-party applications. CI/CD pipelines, in particular, help to automate the process of integrating code changes and delivering them to production efficiently and reliably. The transformed cloud-native strategies are a fundamental concept of O-RAN, which involves building an open, intelligent, and interoperable network [16]. Cloud-native O-RAN promotes adaptive, real-time resource management. However, with the increasing complexity of dynamic and multi-vendor infrastructures, real-time resource management, network exposure, and provisioning are becoming more challenging. The adoption of intelligent APIs, such as Swagger, is crucial for open and streamlined network function exposure, seamless interaction, and scalable service provisioning [17], [18], [19].

Resource allocation strategies gain a new perspective in the O-RAN framework [20], [21]. This paper presents a novel resource allocation radio App (rApp) that provides advanced resource management in O-RAN-compliant networks through multivariate forecasting estimators, allowing the predictions of hundreds of time series at the same time and containerization strategies. Although the experiments in this work focus on PRB forecasting, the proposed framework is not limited to PRB. It can be extended to other O-RAN resource types such as CPU usage, memory allocation, or network throughput. Prominent and robust multivariate probabilistic estimators, integrated into Non-Real Time RAN Intelligent Controller (RT RIC) as a cloud-native resource provisioning rApp, improve the management and prediction of radio resources and promote seamless interaction and analysis of estimator performance in real-time. The gap between theoretical research and real-time deployment in O-RAN is bridged by integrating modern cloud-native technologies with advanced estimators. This paper makes

a novel contribution to the field of resource management and orchestration by laying a solid foundation for future experiments in the field of O-RAN and building an intelligent and scalable telecommunication infrastructure. The primary contributions of this study are as follows:

- State-of-the-art multivariate probabilistic forecasting models (such as Gaussian Process Vector Autoregression (GPVAR) and Temporal Fusion Transformer (TFT)) are integrated into rApp to forecast resource requirements.
- The rApp is containerized and integrated with Swagger's Representational State Transfer (REST)ful APIs for effective network exposure and seamless interaction.
- The performance of probabilistic estimators is compared with multivariate LSTM predictors through quantitative traditional error metrics (Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE)) and computation complexities (training time, prediction time, CPU usage and memory usage).
- TFT multivariate probabilistic estimator is evaluated to illustrate its ability to model long-term dependencies and handle complex input time series data.
- High-dimensional GPVAR multivariate forecasting is analyzed more deeply through a low-rank Gaussian copula process to interpret and understand the scalability, robustness, and impact of different ranks on the performance of GPVAR. GPVARs performance is quantified through traditional error metrics, training time, and advanced performance metrics like train Negative Log Likelihood (NLL), test NLL, normalized deviation, and Continuous Ranked Probability Score (CRPS).

The remainder of the paper is organized as follows: Section II presents related work and motivates the need for probabilistic forecasting and cloud-native solutions. Section III describes the O-RAN framework and rApp deployment using Docker and Swagger. In Section IV, we describe the evaluated multivariate forecasting techniques, including their theoretical foundations and implementation details. Section V presents the simulation results, highlighting the performance of the forecasting techniques, followed by a detailed analysis of the low-rank Gaussian copula using the GPVAR estimator. Finally, Section VI concludes with key findings and future directions.

## II. RELATED WORK

The requirements for intelligent, vendor-independent, and scalable resource allocation in the O-RAN have increased in recent years, leading to significant advances in the forecasting and deployment of cloud-native solutions [4]. Traditional approaches are seen to rely more on univariate forecasting models to estimate a single type of resource based on its past requirements [7], [10]. Machine learning-based solutions are increasingly being used for dynamic adaptation

to real-time network conditions and user demands [7], [22]. Additionally, most previous work focuses on the use of advanced single-point forecasting models such as LSTM and Gated Recurrent Unit (GRU), which offer an improvement in accuracy, but they do not take into account the uncertainties during prediction [23], [24]. The studies [25] and [26] analyzed the performance of traditional estimators and showed that LSTM performed better among the single-point models in terms of execution time and network traffic prediction but faced challenges due to the stochastic nature of resources with complex inter-dependencies in dynamic multivariate environments. The rise of complex and dynamic environments in O-RAN systems is rendering deterministic single-point estimators, such as ARIMA and LSTM time series methods, ineffective due to their inability to capture multivariate dependencies and uncertainties effectively [22], [23].

Recent research has begun to focus on the use of probabilistic techniques such as Bayesian networks [27], Gaussian processes [28], and deep learning-based models [29] for predicting network requirements. ML models such as Random Forest and heuristic approaches are used for optimal PRB prediction in O-RAN [30]. Although they have succeeded in improving the resource utilization by 80% to 90%, they have had difficulties in understanding the dynamic network conditions [30], [31]. The authors in [32] used DeepAR's probabilistic forecasting model to facilitate admission control and improve resource provisioning by accurately predicting future slice demand. Despite their advantages, univariate models such as DeepAR process each time series independently. They do not capture the cross-correlations between multiple series, which are crucial for achieving more reliable predictions of resource requirements in a multivariate network environment such as O-RAN. The analysis and selection of appropriate estimators with advanced forecasting and detection techniques based on network use cases plays a crucial role in the optimal utilization of RAN resources, especially in multivariate environments [33]. It enables service providers to strategically plan resource allocation and optimize network performance in O-RAN without wasting or under-provisioning resources [33].

Multivariate forecasting can significantly enhance resource provisioning by providing a deeper understanding of network demands and interdependencies among variables [7]. Effective network slicing and SLA management for maintaining service-level agreements are achieved through multivariate, multi-step forecasting estimates [22]. Predictive models integrated with spatial and temporal features demonstrate superior performance than temporal-only models in optimizing and estimating the cell- and slice-based network requirements through analyzing various RAN features [23]. The authors in [23] used multivariate, multi-step, spatio-temporal and SLA-driven models such as the "mvLSTM" approach to predicting the impact of single-cell, multicell, and multi-slice-based RAN features on downlink traffic volumes. Despite its strength in terms of low complexity,

the need for refined prediction approaches is emphasized to cope with the complex and dynamic traffic patterns in NextG networks. Similarly, a dynamic, multivariate, attention-based encoder-decoder model facilitated enhanced short- and long-term resource prediction for network slices [7]. The incorporation of fairness metrics through online learning techniques also helps enhance scalability and resource efficiency in complex network environments [34]. Furthermore, resource management strategies enabled by RAN Intelligent Controller (RIC) can facilitate non-RT RIC-based allocation and orchestration of multivariate resources, balancing energy efficiency and performance [5], [34]. Advanced probabilistic estimators, such as GPVAR and TFT, are crucial for efficiently processing high-dimensional data. They predict multivariate resource requirements by effectively capturing the relationships between features by integrating RNN with a low-rank covariance structure [13] and attention mechanisms with gating layers [35], respectively. Through these estimators, a precise and scalable forecasting of resource demands can be achieved, ensuring a sustainable O-RAN [6], [36].

In addition to their strengths, some studies have also identified certain limitations of probabilistic GPVAR and TFT estimators, motivating the evaluation in this work. To avoid computational overhead and address network scaling issues caused by changing dimensions and time-series length, GPVAR relies on low-rank and sparse approximations [13]. Furthermore, state-of-the-art accuracy can be achieved with the attention and gating mechanisms of TFT; however, on smaller datasets or highly correlated series, the TFT model may overfit and lead to unstable training [35]. Moreover, the performance advantage of TFT often comes at the cost of higher memory and computational requirements than the LSTM baseline model, due to multi-head attention layers, gating mechanisms, and deep sequence encoders [35]. These limitations make rigorous evaluation of estimators crucial under realistic multivariate O-RAN data constraints, as they directly influence the accuracy, robustness, and adaptability of results.

O-RAN is also positioned to promote sustainability in 6G by advancing research through concrete mechanisms and design trade-offs. Its openness, modularity, and AI integration capabilities are claimed to reduce energy requirements and e-waste without compromising performance [37]. Computation and power overhead can also be reduced at the system level through energy-aware xApp/rApp scaling and placement [38]. Moreover, safe evaluation of energy-saving policies and automation of closed-loop control can be achieved through digital-twin-assisted O-RAN [39]. At the control level, dynamic activation/sleep of RF chains via deep RL-based xApps enabled energy savings without degrading Quality of Service (QoS) [40]. Probabilistic and explainable AI/ML modelling has also gained attention in recent years for achieving more reliable energy optimisation solutions [6], [41]. This work complements these directions by providing

a baseline analysis of the use of cloud-native rApp for multivariate resource demand forecasting, which can be used by energy-saving policies and xApp scheduling algorithms to make sustainable and risk-aware decisions in O-RAN [6], [42].

The scalable and seamless deployment of these advanced forecasting models is the next crucial step towards intelligent and diverse resource management in the O-RAN [2]. Cloud-native solutions like microservices, Kubernetes, and containerized orchestration are revolutionizing resource management by promoting flexible, scalable, and automated network functions [18], [43]. Docker, being one of the leading containerization platforms, is the most widely used tool for such microservices-based deployments in O-RAN. They provide seamless lifecycle management, portability, and dynamic compatibility [44], [45]. Kubernetes enhances this process by enabling scalability and real-time orchestration of network components [46]. rApps and eXtended application (xApp)s in O-RAN are efficiently deployed through containerized solutions as they support simple lifecycle management and ensure application compatibility across different infrastructures and platforms [4]. When integrated with Swagger Application Programming Interface (API), they facilitate real-time updates and analysis of the resource allocation process by simplifying the interaction between users and applications [18], [47]. Through the REST APIs, Swagger exposes and interacts with network applications, promoting enhanced automation and transparent operations [17], [18]. Studies have also emphasized the importance of such tools for flexibility in managing different networks and effortless interaction between RIC controllers and other O-RAN elements [48]. Cloud-native solutions enable operators to develop modular and service-based architectures by fostering interoperability and simplifying deployment complexity [44], [49].

To summarize, existing research highlights the fact that effective resource management in O-RAN is important for proactive scaling, advanced resource sharing, and effortless placement of virtual network functions (VNFs), allowing O-RAN to dynamically adapt to service demands without wasting resources [50], [51]. Cloud-native solutions that meet the needs of the real world without much human intervention are expected to be developed for an open, intelligent, and interoperable O-RAN framework [52]. Our work leverages these insights to improve predictive analytics, accuracy, and proactive decision-making in O-RAN through multivariate probabilistic estimation and AutoML-based tuning of hyperparameters. Moreover, the containerization of rApp and integration of Swagger API aids in enhancing network function exposure through seamless interaction and scalable service provisioning. Therefore, this paper provides a solid foundation for analyzing the complexity and benefits of forecasting models that involve multiple tenants sharing resources, offering intelligent and scalable solutions for the evolution of advanced

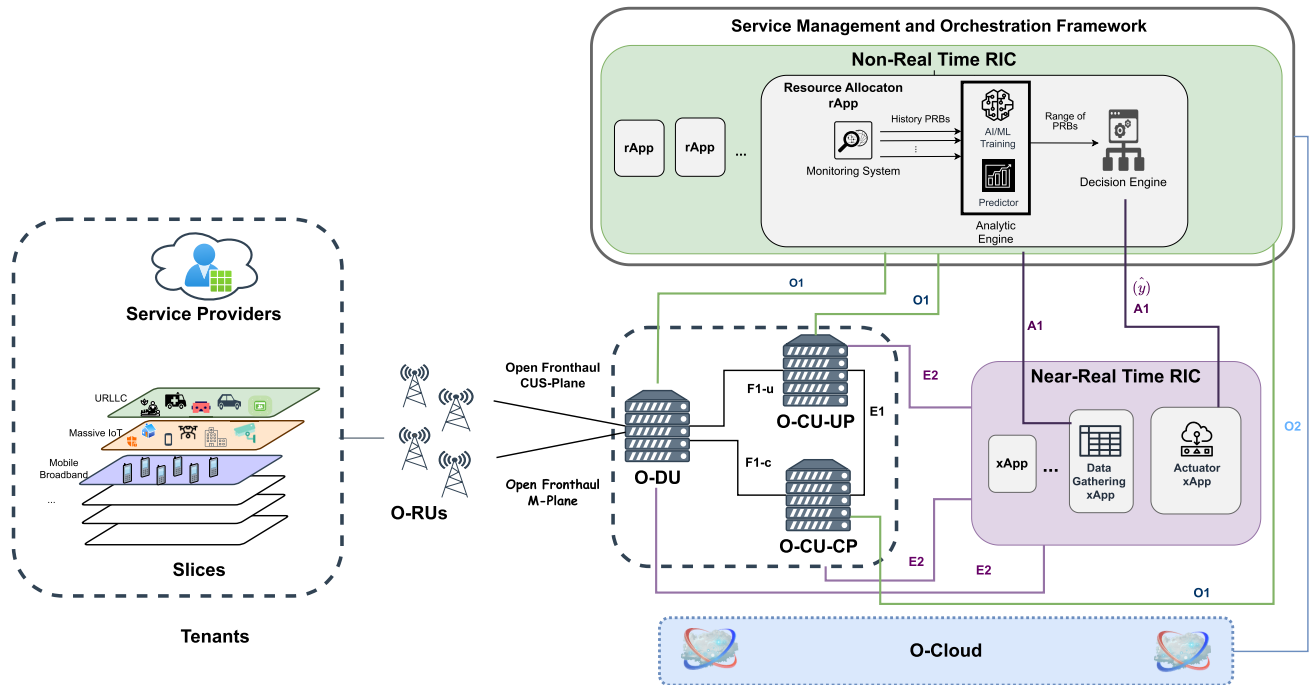


FIGURE 1. Cloud Native O-RAN architecture.

telecommunication networks, including the transition to 6G.

### III. CLOUD NATIVE O-RAN FRAMEWORK

Traditional RANs are highly hardware-dependent, leading to vendor lock-in issues and a massive increase in CAPEX and OPEX costs [4]. The network became challenging for integrating intelligence, building a reliable and cooperative telecommunication infrastructure, leading to establishment of an advanced software-defined RAN solution, O-RAN.

#### A. O-RAN SYSTEM MODEL

O-RAN aims to create an open, intelligent and interoperable RAN environment [16]. Figure 1 shows a cloud-native framework of O-RAN and illustrates the modernization and scalability of the architecture. The Service Management and Orchestration (SMO) layer is the core of the framework that interacts with Non-RT RIC and Near-RT RIC for effective resource orchestration, management and dynamic allocation of functions [2]. O-RAN supports multiple tenants and various services through these controllers as part of a coherent architecture. The functionalities of Open-Radio Unit (O-RU), Open-Distributed Unit (O-DU) and Open-Central Unit (O-CU) are similar to those of the disaggregated 5G RAN, but with additional support for O-RAN-based specifications and interfaces [2], [4]. Near-RT RIC is responsible for handling time-critical tasks such as data collection, load balancing, fault recovery, policy enforcement and control signaling with latency requirements between 10 ms and 1 second. They interact with O-DU and O-CU via the E2 interface

for real-time optimization and management. This work uses two xApps: Data gathering xApp and Actuator xApp. Data gathering xApp collects all historical PRB data from O-DU via the E2 interface, and the actuator xApp is responsible for executing the decisions received from the decision engine via the A1 interface.

Non-RT RIC in SMO is critical for managing long-term applications with latency requirements greater than 1 second, such as AI/ML training, policy management, and analytics in the form of rApps. We propose a containerized resource allocation rApp integrated with the swagger API<sup>1</sup> which consists of three main components:

- **Monitoring system:** The monitoring system receives the history PRB data from the data gathering xApp via the A1 interface and forwards it to the analytical engine.
- **Analytical engine:** The analytical engine initially processes the historical data and splits it into training, validation and test data sets. The Forecasting estimators are trained using the multivariate historical PRB data, and then all future PRB demands of the tenant are predicted.
- **Decision engine:** Finally, the decision engine obtains the range of estimated PRBs from the predictor in the analytical engine and provides computation and error metrics for detailed performance analysis of each estimator. The decision engine can utilize the predictions to perform probabilistically informed resource

<sup>1</sup>Interaction with containerized and swagger API integrated resource allocation rApp is briefly explained in the Annex Section.

provisioning, resulting in increased energy efficiency and effective resource management.

The separation between Near-RT RIC and Non-RT RIC highlights the hierarchical and interconnected nature of O-RAN, increasing flexibility and efficiency. The modular design of rApps facilitates independent containerization and deployment while promoting seamless management and updates. Moreover, the modular design of the rApp supports extension to additional resource types, making it suitable for broader O-RAN radio and computation resource management tasks.

The cloud stack element hosts the O-RAN components and provides a virtualized environment through the integration of cloud stack technologies [53]. It also manages containers and virtual machines of RICs to ensure application scalability, agility, and resource efficiency [53]. The incorporation of cloud-native principles enhances O-RAN's ability to adapt to dynamic network requirements. It supports O-RAN's vision to build an intelligent and interoperable radio network suitable for future 6G developments.

### B. SEAMLESS DEPLOYMENT WITH DOCKER

Containerization of resource allocation rApp modules is crucial for creating a scalable, effective, and platform-independent integration into the O-RAN infrastructure. Docker is a service platform that facilitates the management and creation of containers through virtualization at the operating system level. rApp modules and their dependencies are packaged together with the corresponding libraries, configurations and runtime requirements in Docker's lightweight and robust environment to ensure seamless operation on different platforms, regardless of the underlying hardware and software [52]. First, a Docker image is created for rApp modules and their dependencies. This image is then stored in the container registry and can be retrieved and deployed as required [54]. Behavioral consistency during development, testing and real-time deployment is ensured by initializing rApp as a container. Standardized and interoperable containerized rApp can be efficiently integrated with RIC [4]. In the context of the architecture in this paper, the containerized rApp includes a monitoring system for data collection, an analytical engine for training and predicting resource requirements, and a decision engine that presents the performance of the estimator while being detached from other network functions and rApps. In addition, scaling and deployment become easier as containers are lightweight. Probabilistic algorithms encapsulated with cloud-native solutions enable effective handling of dynamic network conditions [20], [21]. Scaled network functions and rApps can be effectively managed through orchestration tools such as Kubernetes, which provide high availability and resilience [54].

Containerization is an initial transformative step towards deploying an rApp in O-RAN. The integration of CI/CD pipelines is an exciting path for future development [52]. The

building, testing, and deployment of rApp is automated by the pipeline to ensure a smooth and accurate workflow. rApp updates can be evaluated and deployed in less time, minimizing human effort and making them relevant for the use of O-RAN and suitable for the diverse requirements of modern telecommunication networks. Deploying multiple rApps in parallel for specific tasks such as resource allocation, traffic monitoring, fault detection, and much more is made easier by supporting scalability through orchestration and CI/CD, promoting a robust and adaptable O-RAN environment. The integration of the CI/CD pipeline into the containerized rApp would pave the way for building a fully automated and scalable deployment strategy in the future to improve the efficiency and responsiveness of O-RAN deployment [20], [21], [52].

### C. ENHANCED NETWORK SERVICE EXPOSURE WITH SWAGGER

Swagger API plays a vital role in the efficient management and operation of O-RAN rApps. It streamlines service exposure, control, and resource access effectively [18]. This open-source framework enables simplified design, documentation, flow control, and interaction with APIs [49], [55]. The API endpoints facilitate developers and researchers to interact, test, and visualize network services effortlessly in real-time through an interactive and user-friendly interface [47]. Through the interface, users can look for available operations and input requirements and obtain instant feedback, which significantly enhances the usability of applications in dynamic environments [20].

In O-RAN, containerized rApps integrated with Swagger API simplify interaction with the backend environment and enable effortless lifecycle management, including development, testing, deployment, and scaling [45], [49]. Swagger's browser-based interface facilitates simplified API management and interactions, eliminating the need for sophisticated tools [49]. The network exposure is extended here by gaining access to the key performance metrics and predictions, which are significant for effective network management in real-time. Moreover, O-RAN components can be adjusted to expose telemetry data, such as traffic load, latency, and throughput, through a Swagger API for advanced analytics and forecasting [17], [43]. This work utilizes multivariate PRB data, which is fed into the selected forecasting model, to estimate resource demand, as briefly explained in the Appendix Section. All these advantages of the containerized Swagger API make it a powerful tool for the efficient accessibility of rApp [4], [47], [48]. The Swagger UI interface and API endpoints used for training, prediction, and metric visualisation are also shown in the Appendix. It briefly illustrates how users can interact with the rApp through REST API calls in real time.<sup>2</sup>

<sup>2</sup>More detailed figures and explanations are provided in the Appendix Section.

#### IV. ADVANCED FORECASTING MODELS

##### A. PROBLEM STATEMENT

Effective and accurate prediction of resources is crucial for making efficient decisions in various environments. Radio resource demands, namely PRBs, are predicted using multivariate forecasting algorithms that combine several influential factors for effective resource provisioning. Univariate algorithms focus solely on past information of the single tenant target variable, but multivariate estimators consider various relevant characteristics, such as history PRB data, time, day of the week, peak hours, etc., of multiple tenants simultaneously to accurately estimate future resource requirements. This paper considers history time-series of  $K$  tenants PRB data as relevant characteristics. More accurate and robust predictions are achieved by gaining insight into the complete relationships between different PRB demands of various tenants.

This section begins with a detailed description of a traditional multivariate LSTM estimator [56] that provides single-point estimates of resource requirements. Traditional forecasting estimators provide single-point estimates without any information about the uncertainty of future demands. This limitation leads to overconfidence in their forecast, causing service providers to make suboptimal decisions. In contrast, probabilistic forecasting algorithms such as GPVAR [13] and TFT [35] are robust and powerful estimators that provide more accurate predictions in the form of probability distributions. The distribution helps to quantify the uncertainties in the forecast and to make more informed decisions in real-time.

Let  $\{y_t\}_{t=1}^T$  denote a univariate time series, where  $y_t \in \mathbb{R}$  is the observation at time  $t$ . In forecasting, a model uses a fixed context window of length  $L_c \in \mathbb{N}$ , called the context length, which includes the past  $L_c$  observations  $\{y_{t-L_c+1}, \dots, y_t\}$  as input.

The model then produces a sequence of future predictions  $\{\hat{y}_{t+1}, \hat{y}_{t+2}, \dots, \hat{y}_{t+L_h}\}$ , where  $L_h \in \mathbb{N}$  is the horizon length, indicating how many future steps are forecast.

Formally, the forecasting function can be written as:

$$f : \mathbb{R}^{L_c} \rightarrow \mathbb{R}^{L_h},$$

$$f(y_{t-L_c+1}, \dots, y_t) = (\hat{y}_{t+1}, \dots, \hat{y}_{t+L_h}) \quad (1)$$

This formulation generalizes to multivariate time series by replacing scalars  $y_t$  with vectors  $\mathbf{y}_t \in \mathbb{R}^d$ , such that:

$$f : \mathbb{R}^{L_c \times d} \rightarrow \mathbb{R}^{L_h \times d} \quad (2)$$

The multivariate PRB allocation forecasting problem is represented as a time series at time  $t$  by  $\mathbf{y}_t \in \mathbb{R}^K$  with  $K$  number of input time series aiming to model the joint conditional distribution as follows:

$$P(\mathbf{y}_{t_0:T} | \mathbf{y}_{1:t_0-1}). \quad (3)$$

where  $t = 1, 2, \dots, t_0 - 1$  represents the history data timestamps and  $t = t_0, \dots, T$  represents the prediction length.

##### B. MULTIVARIATE SINGLE-POINT FORECASTING

###### 1) LONG SHORT-TERM MEMORY (LSTM)

We consider multivariate time series  $\mathbf{Y} = \{\mathbf{y}_t\}_{t=1}^{t_0-1}$  with  $\mathbf{y}_t \in \mathbb{R}^K$  representing the observation across  $K$  time series at time  $t$  aiming to predict future PRB demands  $\mathbf{Y} = \{\mathbf{y}_t\}_{t=t_0}^T$  within time range  $t_0$  to  $T$ . The principal feature of LSTM at time  $t$  includes [56]:

- *Input gate*: Defines how much input information  $\mathbf{y}_t$  is used to update the cell state  $\mathbf{c}_t$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{y}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i). \quad (4)$$

where  $\sigma$  is the sigmoid function, with  $\mathbf{W}_i \in \mathbb{R}^{m \times n}$ ,  $\mathbf{U}_i \in \mathbb{R}^{m \times m}$  representing weight metrics and  $\mathbf{b}_i \in \mathbb{R}^m$  depicting bias vectors.

- *Hidden state*: Modulation of output gate  $\mathbf{o}_t$  and tanh activation of cell state

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \quad (5)$$

where  $\odot$  denotes element-wise multiplication.

- *Output gate*: Symbolizes contribution of updated cell state to the hidden state  $\mathbf{h}_t$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{y}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o). \quad (6)$$

where  $\mathbf{W}_o \in \mathbb{R}^{m \times n}$ ,  $\mathbf{U}_o \in \mathbb{R}^{m \times m}$  are weight matrices and  $\mathbf{b}_o \in \mathbb{R}^m$  is the bias vector for the output gate

- *Forget gate*: Determine the quantity of past cell state to be retained using the sigmoid activation function

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{y}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f). \quad (7)$$

where  $\mathbf{W}_f \in \mathbb{R}^{m \times n}$ ,  $\mathbf{U}_f \in \mathbb{R}^{m \times m}$  are weight matrices and  $\mathbf{b}_f \in \mathbb{R}^m$  is the bias vector for the forget gate.

- *Cell state update*: Input gate, forget gate, and candidate values  $\tilde{\mathbf{c}}_t$  are used to update the cell state

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{y}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c). \quad (8)$$

where  $\mathbf{W}_c \in \mathbb{R}^{m \times n}$ ,  $\mathbf{U}_c \in \mathbb{R}^{m \times m}$ , are weight matrices and  $\mathbf{b}_c \in \mathbb{R}^m$  is the bias vector for the cell state update. Then, the actual cell state is updated as follows:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t. \quad (9)$$

- *Backpropagation Through Time (BPTT)*: BPTT trains LSTM, where network parameters  $\theta$  are optimized by minimizing the loss function  $\mathcal{L}$

$$\mathcal{L} = \frac{1}{T - t_0} \sum_{t=t_0}^T \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|. \quad (10)$$

where  $\hat{\mathbf{y}}_t$  is the prediction of  $\mathbf{y}_t$ . The gradients of the loss function with respect to  $\theta$  are computed through chain rule and backpropagation through time.

**C. MULTIVARIATE PROBABILISTIC FORECASTING**

**1) GAUSSIAN PROCESS VECTOR AUTO REGRESSIVE (GPVAR)**

The main problem is adapted to the context of GPVAR here. The multivariate time series is represented as  $y_{k,t}$ , with  $k \in \{1, 2, \dots, K\}$  indexing each time series component at time  $t$ . The observation vector  $\mathbf{y}_t$  uses  $t_0 - 1$  history PRB data  $\mathbf{y}_1, \dots, \mathbf{y}_{t_0-1}$  and predicts future PRB demands from  $t_0$  up to  $T$  for estimating joint conditional distribution (3) [13]. GPVAR, in particular, for the case of multivariate time series, focuses on optimizing the multivariate normal distribution losses. A non-linear and deterministic state space model is obtained with a state  $\mathbf{h}_{k,t} \in \mathbb{R}^k$  evolving independently for each time series based on transition dynamics  $\varphi$ ,

$$\mathbf{h}_{i,t} = \varphi_{\theta_h}(\mathbf{h}_{k,t-1}, y_{k,t-1}). \quad k = 1, 2, \dots, K \quad (11)$$

with transition dynamics  $\varphi$  parametrized by LSTM [56]. Parameters  $\theta_h$  are shared for all the time series, but the LSTM is applied for each time series separately. With state values  $\mathbf{h}_{k,t}$  known for all time series  $k = 1, 2, \dots, K$  and representing it as  $\mathbf{h}_t$ , the state values gathered for all series at time  $t$ , the joint distribution is parameterized using a Gaussian copula [57],

$$p(\mathbf{y}_t | \mathbf{h}_t) = \mathcal{N}\left(\left[f_1(y_{1,t}), f_2(y_{2,t}), \dots, f_k(y_{K,t})\right]^T, \mu(\mathbf{h}_t), \Sigma(\mathbf{h}_t)\right). \quad (12)$$

The functions  $f_i$  are responsible for transforming the data for the  $k$ -th time series such that it follows a standard normal distribution. State  $\mathbf{h}_t$  is mapped to the mean and covariance of a Gaussian distribution over the transformed observations using  $\mu(\cdot)$  and  $\Sigma(\cdot)$  functions.<sup>3</sup> Moreover, Gaussian Couples allows the modeling of dependencies between multiple random variables even when the marginal distributions are non-Gaussian [58]. Monte Carlo samples of the factorized joint distribution are produced

$$\begin{aligned} p(\mathbf{y}_{t_0}, \dots, \mathbf{y}_T | \mathbf{y}_1, \dots, \mathbf{y}_{t_0-1}) &= p(\mathbf{y}_{t_0}, \dots, \mathbf{y}_T | \mathbf{h}_{t_0}) \\ &= \prod_{t=t_0}^T p(\mathbf{y}_t | \mathbf{h}_t). \end{aligned} \quad (13)$$

by sampling  $P(\mathbf{y}_t | \mathbf{h}_t)$  sequentially and updating  $\mathbf{h}_t$  using  $\varphi$ . Network parameters  $\theta$  are learned from observed data  $\mathbf{y}_1, \dots, \mathbf{y}_{t_0-1}$  through minimizing the loss function using gradient descent-based optimization.

$$-\log p(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{t_0-1}) = -\sum_{t=1}^{t_0-1} \log p(\mathbf{y}_t | \mathbf{h}_t). \quad (14)$$

Efficient multivariate modeling is enabled in GPVAR through a low-rank Gaussian approximation to handle scalability, computation complexity and parameterize the

covariance matrix [13]. One of the principal challenges arises from the dimensionality of the covariance matrix  $\Sigma$ , which increases quadratically with the number of time series  $K$  in the dataset. To mitigate this issue, the covariance matrix is approximated using a low-rank Gaussian model. Instead of computing the full  $K \times K$  covariance matrix, a lower-dimensional representation is estimated, comprising the sum of a diagonal matrix plus a low-rank component. Instead of estimating the full covariance matrix, GPVAR considers the following decomposition, which ensures positive definiteness of the covariance matrix by:

$$\Sigma(\mathbf{h}_t) = \mathbf{D} + \mathbf{L}\mathbf{L}^T, \quad (15)$$

Here,  $\mathbf{D} \in \mathbb{R}^{K \times K}$  is a diagonal matrix, and  $\mathbf{L} \in \mathbb{R}^{K \times r}$  is a learnable low-rank matrix that captures correlations across output dimensions, with  $r < K$ . This approach reduces the computational complexity of the Gaussian likelihood from  $O(N^3)$  to  $O(Nr^2 + r^3)$ . It also enables scalability by transforming each time series using Gaussian Copulas and effectively captures the evolving interdependencies among time series without significant performance degradation [13].

**2) TEMPORAL FUSION TRANSFORMERS (TFTs)**

TFTs can effectively capture complex temporal dependencies and quantify uncertainties through attention mechanisms and quantile loss, providing accurate predictions [35]. TFTs are also evaluated under high-dimensional multivariate settings. Like GPVAR, multivariate observation vector at time  $t$  is represented as  $\mathbf{y}_t$  and forecasts are depicted as  $\mathbf{y}_{t_0}, \dots, \mathbf{y}_T$  with the main aim of estimating joint conditional distribution (3). Let the prediction horizon time steps be  $\tau$ . TFTs use additional features like *unknown*  $z_{k,t}$ , *known*  $x_{k,t}$ , and static covariates  $s_k$ , providing measured entities contextual metadata independent of time to enhance the predictions. Prediction intervals are obtained in the form of quantiles at the output. The probabilistic TFT model uses a Quantile Regression likelihood loss function, and every quantile  $q$  forecast at time  $t$  of  $\tau$ -step-ahead is depicted as:

$$\hat{y}_k(q, t, \tau) = f_q(\tau, y_{k,1:t_0-1}, z_{k,1:t_0-1}, x_{k,1:T}, s_k). \quad (16)$$

with  $y_{k,1:t_0-1}$  showing values in the conditioning range,  $z_{k,1:t_0-1}$  being unknown,  $x_{k,1:T}$  known in the prediction range, and  $s_k$  static covariates independent of time. In our work, we use only known features such as the day of the week and hours to forecast. The key elements of TFTs are<sup>4</sup>:

- *Gating mechanisms*: The computational complexity is dynamically adjusted by gating mechanisms so that unused components can be skipped and the depth and network complexity can be optimized depending on the data set and prediction environment. Computing resources are effectively managed and performance is improved by balancing internal structures.
- *Variable selection networks*: With TFT, identifying and focusing on the required input features at each time step

<sup>3</sup>A detailed description of the  $\mu(\cdot)$ , and  $\Sigma(\cdot)$  can be obtained from [13].

<sup>4</sup>More detailed explanation can be obtained from [35].

is crucial. The most relevant features are dynamically selected for each prediction using this component, ensuring that the models focus on the most important features and ignore noisy data.

- *Static covariate encoders*: Contact vectors are generated to encode static features that are time-independent to condition the temporal dynamics of the model and provide stable contextual information for effective prediction.
- *Temporal processing*: Short-term temporal relations are processed locally with a sequence-to-sequence layer, while a multi-head attention block models long-term dependencies to ensure efficient learning and appropriate integration of temporal patterns derived from observed and known time-varying inputs.
- *Prediction intervals via quantile forecasts*: The range of possible targets is estimated for each prediction horizon by using this characteristic to measure forecast confidence and uncertainty.

The model is trained by optimizing the quantile loss function for quantile  $q$ , depicted as:

$$L_q(y_t, \hat{y}_t^q) = \begin{cases} q(y_t - \hat{y}_t^q) & \text{if } y_t > \hat{y}_t^q \\ (1 - q)(\hat{y}_t^q - y_t) & \text{otherwise} \end{cases} \quad (17)$$

with  $y_t$  and  $\hat{y}_t^q$  being true and predicted quantile value at time  $t$ . The total quantile loss across all quantiles and time steps is given as:

$$L = \frac{1}{|Q|} \sum_{q \in Q} \sum_{t=t_0}^T L_q(y_t, \hat{y}_t^q), \quad (18)$$

where  $|Q|$  represents the number of quantiles.

#### D. EVALUATION METRICS

The performance of forecasting models can be effectively quantified and analyzed using error metrics. The consistency of predictions is assessed through PRB history data collected from  $K$  tenants. A deeper understanding of the metrics is essential to select the best estimator for appropriate use cases [6]. The error metrics also provide information about the accuracy and reliability of the estimator, facilitating the comparison of models and enabling reliable decision-making. This paper primarily focuses on error metrics, including NLL, RMSE, MAE, MAPE, Normalized Deviation (ND), and CRPS, which are calculated between actual and predicted PRB estimates. To better understand metrics, let's consider the first series of test data, represented as  $y_{k,t}$ , where  $t = t_0, \dots, T$ .

RMSE, MAE, and MAPE are the most widely used metrics for measuring the performance of single-point estimators [59]. In case of probabilistic estimators, obtained prediction  $\hat{y}_{k,t}$  at a particular time  $t$  is a column vector which is represented as  $[\hat{y}_{k,t}^1, \hat{y}_{k,t}^2, \dots, \hat{y}_{k,t}^q]^T$ , where  $q$  represents the range of percentiles from 1-th to 99-th [12]. So, the median of forecasts at every time  $t$  is considered for computing

RMSE, MAE, and MAPE of probabilistic models. Single point estimators like LSTM are compared with probabilistic models in a fair way using only these three metrics [59], whose mathematical formulations are discussed below in detail.

- *Root Mean Square Error (RMSE)*: The Square root of the average of the squared difference between actual PRBs and their forecasts is measured to evaluate the model accuracy [59].

$$RMSE = \sqrt{\frac{1}{T} \sum_{t=t_0}^T |y_{k,t} - \hat{y}_{k,t}|^2} \quad (19)$$

- *Mean Absolute Error (MAE)*: Average magnitude of error in the set of forecasts is measured by equally treating the over- and under-estimations to make them more robust against outliers [59].

$$MAE = \frac{1}{T} \sum_{t=t_0}^T |y_{k,t} - \hat{y}_{k,t}| \quad (20)$$

- *Mean Absolute Percentage Error (MAPE)*: Difference between actual PRBs and their forecasted is computed in terms of percentage [59]. They are advantageous when datasets of varying scales or different contexts are used.

$$MAPE = \frac{1}{T} \sum_{t=t_0}^T \frac{|y_{k,t} - \hat{y}_{k,t}|}{|y_{k,t}|} \quad (21)$$

Metrics like ND, CRPS, and NLL quantify probabilistic forecasting models effectively, which are structured as follows:

- *Normalized Deviation (ND)*: Forecast accuracy is well assessed using this metric by normalizing the average deviation between actual and forecasted PRBs with a range of actual values [58]. Low ND signifies high prediction accuracy.

$$ND = \frac{1}{T} \sum_{t=t_0}^T \frac{|y_{k,t} - \hat{y}_{k,t}|}{\text{mean}(\mathbf{y}_k)} \quad (22)$$

- *Continuous Ranked Probability Score (CRPS)*: estimators' accuracy and robustness are more precisely assessed using CRPS. It calculated the difference between Cumulative Distribution Function (CDF) of predictions and empirical CDF of actual values to measure and quantify the deviation of predictions from true values [60].

$$CRPS(\mathcal{F}, y_{k,t}) = \int_{-\infty}^{\infty} (\mathcal{F}(\hat{y}_{k,t}) - 1(\hat{y}_{k,t} \geq y_{k,t}))^2 dy \quad (23)$$

where,  $\mathcal{F}$  represents forecast distribution.

- *Negative Log Likelihood (NLL)*: This most widely used metric efficiently assesses the performance of probabilistic estimators. It quantifies the likelihood of actual data under a forecasted probabilistic distribution [13].

Low NLL signifies high confidence and accuracy in prediction. The NLL is calculated for both the training and prediction datasets to gain a deeper insight into forecast model performance [13].

$$\text{Train NLL} = -\frac{1}{T} \sum_{t=1}^{t_0-1} \sum_{k=1}^K \log p(y_{k,t} | \hat{y}_{k,t}) \quad (24)$$

$$\text{Test NLL} = -\frac{1}{T} \sum_{t=t_0}^T \sum_{k=1}^K \log p(y_{k,t} | \hat{y}_{k,t}) \quad (25)$$

where,  $p(y_{k,t} | \hat{y}_{k,t})$  depicts the predicted probability of the actual PRB values  $y_{k,t}$  and  $t = 1, 2, \dots, t_0 - 1$  represents the time stamps corresponds to train data in Train NLL (1-month) and  $t = t_0, \dots, T$  represents total number of test time stamps in test data for the case of Test NLL.

## V. SIMULATION RESULTS

In this section, a detailed analysis of different multivariate forecasting models is proposed. The estimators are developed using libraries such as Pytorch forecasting and Keras, which are available in the Python programming language. They are further integrated into the O-RAN framework as an rApp for training and predicting downlink PRB allocation data. The models considered in this work for efficient PRB forecasting are LSTM, GPVAR and TFT.

The dataset used in this paper was generated by simulating traffic and mobility patterns for different numbers of end users in a realistic urban environment [61]. The simulation considered a city-wide RAN scenario, and user behavior was recorded across 50 base stations. Mobility patterns of various users were simulated, evenly distributed across enhanced mobile broadband (eMBB), ultra-reliable low latency communication (URLLC) and massive machine-type communication (mMTC) slices, reflecting the classic use cases of modern RAN deployment [61]. To ensure realistic traffic dynamics and mobility patterns, a highly populated geographical region of Milan, Italy, was considered [61]. The recorded dataset contains detailed information about PRB allocations, slice-specific traffic demand, SLA violations, and network conditions. The O-RAN mapping in [61] ensures the replication of real-world resource requirements and the relevance of the collected PRB provisioning data to this study.

In real-time, the historical PRB datasets can be derived from O-DU via the E2 interface. The data is split in a ratio of 80:20 for training & validation and testing. The 80% training and validation data includes the history of downlink PRBs allocated at different base stations for a period of 1 month, where the PRBs are recorded on an hourly basis. In addition, the next 48 hours or 2 days of PRB data are predicted during the forecast.

### A. MODEL PARAMETERS & SCENARIO CONFIGURATIONS

To evaluate the performance of the different estimators, the LSTM model, which can efficiently capture complex

temporal dependencies, is initially structured. The features of LSTM include 3 dense layers, 0.5 dropout rate, Adam optimizer, L2 regularization, and 150 maximum epochs as listed in Table 1. Setting the number of units to 10 and the batch size to 128, as well as carefully adjusting other parameters such as the learning rate and hidden layers using the AutoML framework Optuna [62], [63], helped to achieve an efficient trade-off between overfitting and model complexity. Rather than relying on early stopping or other tuning mechanisms, training configurations can be dynamically adjusted based on validation losses and optimization goals. They ensure the practical use of resources by reducing the need for human intervention and effort, and minimizing the problems of over- or under-utilization of resources through intelligent parameter tuning. This adaptive approach is crucial for positive progress in model performance.

Then, the probabilistic estimators GPVAR and TFT are developed with key parameters such as 128 batch size, maximum epochs of 150, and a gradient clip value of 0.1. They provide a forecast distribution by effectively understanding the data trends during training, unlike LSTM, which provides single-point estimates. In addition, GPVAR and TFT incorporate a mechanism to integrate and handle time-varying known real inputs directly into the prediction framework that may influence the forecasts. Sufficient historical data is provided in the form of a context length of 72, together with the “time varying known reals” incorporating parameters such as hours, weekdays, and weekends to ensure effective learning and accurate predictions. Specifically, with respect to the GPVAR configuration, 2 Recurrent Neural Network (RNN) layers are used along with multivariate normal distribution losses and Adam optimizer. In addition, in the case of TFT, a dropout of 0.1 is used together with the quantile loss and the Adam optimizer for structuring. For both the GPVAR and TFT estimators, Optuna is used to dynamically select the optimal learning rate and hidden size required for effective training and resource demand prediction. The integration of Optuna into advanced forecasting models enhances the optimization process of the model, eliminating the need for exhaustive manual tuning. Available resources and training time are efficiently utilized thanks to the adaptive nature of Optuna, allowing the model to converge faster and more effectively. This has improved the scalability and robustness of the estimator, simplifying the experimental process for model development. The detailed list of parameters is shown in Table 1.

All the following experiments are executed locally on a dedicated GPU-enabled computing environment for effective training, scalable experimentation and precise performance evaluation. For the purpose of the analysis, a scenario with network coverage by several base stations is considered. Based on the available dataset,  $K = 29$  variables representing the PRB demand of  $K$  tenants are provided as input for multivariate forecast estimators. The entirety of historical PRB demand serves as a crucial variable for understanding

TABLE 1. Model parameters and configurations.

Parameter	LSTM	GPVAR	TFT
Batch Size	128	128	128
Max Epochs	150	150	150
Layers	3 Dense Layers	2 RNN Layers	Attention Head, Hidden Continuous Size: 1,8
Optimizer	Adam	Adam	Adam
Loss Function	Mean Square Error (MSE)	Multivariate Normal Distribution Loss (Rank-28)	Quantile Loss
Dropout Rate	0.1	-	0.1
Gradient Clipping	-	0.1	0.1
Regularization	L2 Regularization	-	-
Context Length	72	72	72
Time Varying Known Reals	-	Hours, Weekdays, Weekends	Hours, Weekdays, Weekends
Hyperparameter Tuning	Optuna for selecting optimal learning rate and hidden size		

TABLE 2. Performance comparison of DeepAR and GPVAR on a 9 Tenant’s PRB demand.

Evaluation Metrics	DeepAR	GPVAR
RMSE	39.70	24.11
MAE	38.91	11.83
MAPE	15.23	8.06
Normalized Deviation	19.01	16.31
CRPS	0.505	0.402

comprehensive resource utilization, optimizing prediction, and managing resource allocation between different tenants. Together, they contribute to effective network performance through a stable quality of service provisioning. Understanding this helps to allocate the appropriate resources without waste or bottlenecks.

**B. LIMITATIONS OF UNIVARIATE PROBABILISTIC ESTIMATORS AS BASELINES**

Initially, to depict the limitations of using univariate probabilistic estimators like DeepAR as a baseline, their performance is compared with that of the multivariate GPVAR forecasting model. DeepAR treats each tenant’s PRB time series independently and requires a separate model for each series for both training and inference. GPVAR, on the other hand, being a multivariate probabilistic model, processes all tenant’s information jointly and explicitly models their dependencies and relations, which is crucial for resource optimization in a multivariate O-RAN environment. As we consider  $K = 29$  tenants in this work, running 29 DeepAR models in parallel to process 29 tenant’s PRB data produces significant computational overhead and does not provide additional methodological benefits due to its inability to exploit cross-tenant correlation. Therefore, for a fair comparison between the univariate and multivariate probabilistic estimators under identical data conditions, we considered PRB demands of  $K = 9$  tenants.

Table 2 shows the performance comparison of DeepAR and GPVAR on 9-Tenant’s PRB demand with regards to multiple evaluation metrics. Compared with GPVAR, DeepAR shows lower prediction accuracy, with RMSE, MAE, and MAPE of 39.70, 38.91, and 15.23, respectively. Additionally, it indicated a weaker calibration and less reliable uncertainty

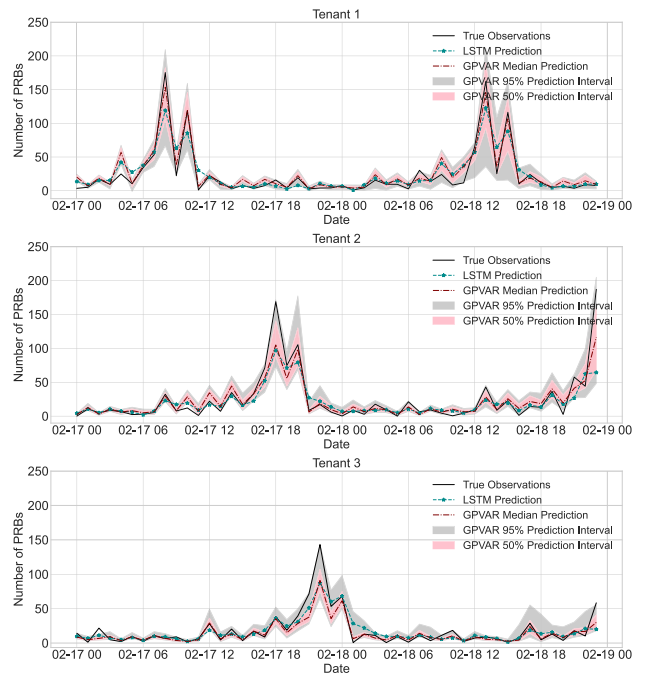


FIGURE 2. PRB data predictions using GPVAR estimator.

quantification with a high normalized deviation of 19.01 and a CRPS of 0.505, unlike GPVAR, which showed superior performance with normalized deviation and CRPS of 16.31 and 0.402, respectively. These results clearly show that multivariate frameworks such as GPVAR (and TFT in our broader evaluation) are more reliable and robust for predicting future resource demands while simultaneously capturing uncertainty and cross-dependencies with high accuracy.

**C. MULTIVARIATE PRB FORECASTING ASSESSMENT**

As a starting point for the performance evaluation, a detailed visualization of the predictions obtained using GPVAR and TFT provided in Figures 2 and 3 is examined. The figures consist of 3 subplots with PRB predictions of tenants 1, 2 and 3. Note that for reasons of space, PRB predictions for all 29 tenants are not shown here. The black line representing the

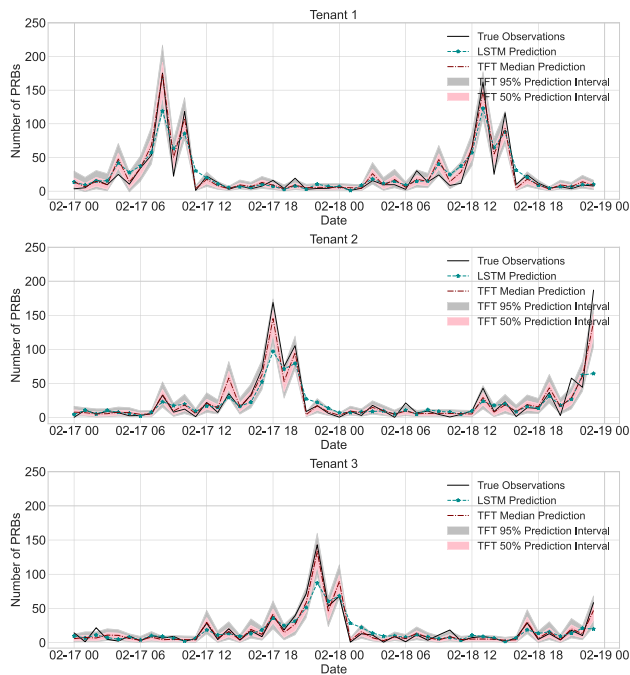


FIGURE 3. PRB data predictions using TFT estimator.

actual/real PRB values serves as a benchmark for evaluating the predictions. The LSTM estimates are shown in a cyan-blue star-marked line, and the probabilistic model estimates are shaded in maroon. The area with light maroon shading symbolizes the 95% prediction range with a wider range of potential PRBs, and the darker shading shows a 50% prediction range with the median shown in the maroon dot-dash line. The range helps service providers to assess the uncertainty of the model and the reliability of the prediction. A smaller range leads to more confident decision-making regarding resource requirements.

Figure 2 shows the predictions of PRB demand using the GPVAR estimator. The median follows the true PRB in the predictions of tenant 1 more closely. The 50% prediction interval seems to be a little biased, as it misses the actual requirements at some points, but the true values are mostly within the range of the 95% interval. Higher uncertainties can also be seen in all PRB demand forecasts for the date 02-18. For tenants 2 and 3, GPVAR performs comparatively well by effectively capturing the trends and fluctuations. LSTM shows the worst prediction, regardless of the tenants. Overall, the GPVAR model predicts the required resources efficiently, although there is some uncertainty due to the non-stationarity of the data, volatility, and temporal fluctuations.

Figure 3 shows PRB data predictions using TFT estimator. In this figure, TFTs show better data processing and a better understanding of complex patterns. The true observations are more accurately aligned with the median in all three subplots. The prediction intervals are narrower than the intervals of GPVAR in Figure 2, which underlines their great confidence in prediction with reduced uncertainty. The

peaks and troughs are captured more effectively thanks to robust modeling, temporal fusion techniques, a self-attention mechanism, and the precise performance of the estimator. It demonstrates competitive predictive ability compared to GPVAR, while LSTM shows less accuracy in estimating the resource requirements without quantifying the uncertainties in its predictions. The uncertainty quantification of GPVAR and TFT enables risk-aware resource provisioning with probabilistic guarantees, which are crucial for automated decision-making across the network.

#### D. PERFORMANCE AND COMPUTATION COMPLEXITY ANALYSIS

The CPU and memory usage recorded and analyzed in this section highlight the processing overhead during training and inference, independent of GPU acceleration. The estimator’s computational footprint on the system is emphasized here, along with the relative efficiency of all models under practical conditions. The consideration of CPU and memory utilization by estimators, together with their accurate prediction capability, is essential for real-world deployment. Figures 4 and 5 help us to better understand the complexity of the model. In all figures, the x-axis shows the time for training/prediction in seconds, while the y-axis shows the CPU utilization percentage. The memory usage percentage of all models during training/prediction is shown in circle color in the form of a heat map, and the CPU usage percentage is depicted by the varying size of the circle. Initially, Figure 4 explains the CPU and memory usage during the training phase of the model over time. The time during training includes the time required for preprocessing the data, splitting, training the model, and storage. CPU utilization fluctuates throughout the runtime due to the complex structures and resource requirements of all probabilistic estimators. The maximum CPU utilization for both GPVAR and TFT is between 46.8% and 48.9%, while LSTM requires very little Central Processing Unit (CPU) at around 10.5%. Despite the sequential processing of data, LSTM is also the fastest in terms of training time due to its simple architecture and fewer calculations.

A similar performance is evident in Figure 4 in relation to memory utilization. GPVAR requires the highest amount of memory around 97.5% during training due to the modeling of the multivariate distribution and temporal dependencies. TFT, on the other hand, exhibits stable behavior with moderate memory requirements but high training time due to the attention mechanism, variable selection network to select relevant inputs at each step, and temporal fusion. The LSTM memory requirement is the lowest, and the training time is very low due to its lightweight nature, smaller number of parameters, and simpler architecture, which illustrates the trade-off between computational complexity and network efficiency.

Figure 5 shows the CPU and memory utilization during PRB prediction. The prediction time is the sum of the loading

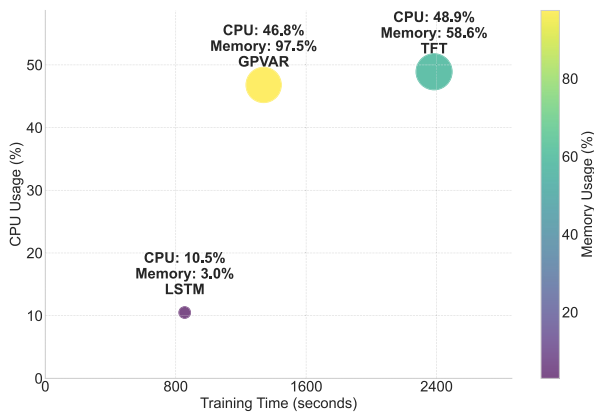


FIGURE 4. CPU and Memory usage of estimators during data training.

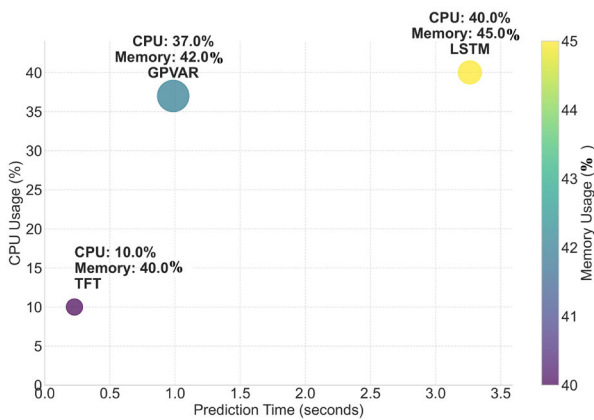


FIGURE 5. CPU and Memory usage of estimators during prediction.

time of the data and the trained model, the prediction time and the post-processing time. TFT requires the lowest CPU of 10% for the prediction. GPVAR and LSTM require almost the same amount of CPU, about 37% to 40%, due to the RNNs involved, but the time in which the CPU is occupied is comparatively high for LSTM due to the serialization of data processing. This metric demonstrates the model’s effectiveness in processing new data and making accurate predictions.

Figure 5 also shows the memory usage during the tests for all LSTM, GPVAR and TFT estimators. TFT and GPVAR require a moderate amount of memory, about 40% to 42%, for a lower duration than LSTM. The memory requirement depends on the complexity of the estimator’s architecture. LSTM requires a considerable amount of memory for sequential processing, additional memory for storage due to multivariate processing, large input embeddings and the calculation of internal gates. On the contrary, a sophisticated TFT framework requires the lowest memory and makes faster predictions due to the parallel processing of all time steps by the attention mechanism.

Table 3 quantifies the overall performance of the estimator using various error metrics, such as RMSE, MAE and

TABLE 3. Comparison of the performance of different forecasting models for varying data length.

Forecast Models	Training Dataset Length			
	1 Week	2 Weeks	3 Weeks	1 Month
RMSE				
LSTM	34.940	29.619	26.286	25.815
GPVAR	33.349	28.422	24.588	20.017
TFT	33.601	31.185	20.617	14.382
MAE				
LSTM	17.266	15.494	14.973	13.207
GPVAR	15.247	13.334	11.942	9.387
TFT	15.618	15.206	10.328	7.967
MAPE				
LSTM	22.786	17.956	14.152	8.818
GPVAR	6.321	5.407	4.924	4.062
TFT	6.388	6.087	4.071	3.293
Training time (Seconds)				
LSTM	189.044	326.337	559.475	855.055
GPVAR	581.76	876.11	1185.42	1339.37
TFT	476.36	1081.60	1575.75	2384.27
Prediction time (Seconds)				
LSTM	2.161	2.253	2.361	3.263
GPVAR	0.786	0.802	0.958	0.986
TFT	0.172	0.207	0.212	0.2284

MAPE, and computational complexity, such as training and prediction time. The history PRB data of different lengths (1 week, 2 weeks, 3 weeks and 1 month) during the training is taken into account for the calculation in the table. LSTM shows the highest values of RMSE over different data lengths due to the worst performance in predicting the PRB demand. TFT performs better across all data lengths as it has the advantage of capturing the data pattern with high precision. The performance of GPVAR is moderate and better than that of LSTM due to the covariates-based training. Moreover, the increase in data length gradually improves all error metrics due to improved learning, less underfitting, stability and robustness. MAE, which measures the error magnitude without considering its direction, shows similar performance to RMSE. TFT outperforms the other models due to its effective predictive ability. The percentage-based error called MAPE is considered the worst when the value is higher, like RMSE and MAE. Of all the error metrics, TFT performs the best, especially when 1 month of data is input during training, with RMSE values of 14.382, MAE values of 7.967 and MAPE of 3.293.

For multivariate forecasting models, the training time does not depend on the tenants or base station types, and the PRB requests are forwarded to the estimators as a  $k$ -dimensional array. GPVAR requires the longest training time when the data length is low due to matrix operations, Gaussian processing and RNN complexity. The increase in data length affects the training time due to the higher computational complexity, making TFT the most time-consuming process at 2384.27 seconds. LSTM requires less time for training than GPVAR and TFT due to its simpler framework. As the data length gradually increases, the training time increases, in contrast to the improvement we saw in the error metrics. This shows the significant trade-off between complexity and

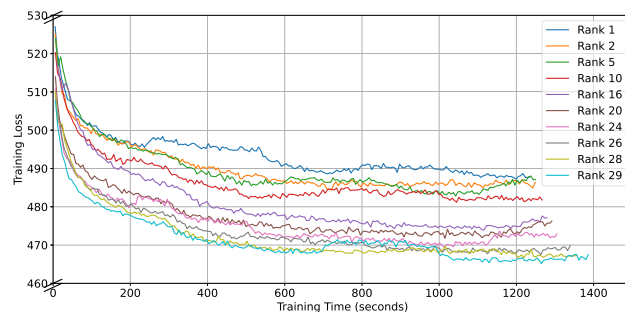
**TABLE 4. Comparison of the performance of different forecasting models for increasing tenants.**

Forecast Models	Number of Tenants				
	2	9	15	22	29
	<b>Training Time (Seconds)</b>				
LSTM	271.22	446.08	635.80	822.43	855.05
GPVAR	979.81	1039.65	1128.54	1334.39	1339.37
TFT	924.90	1268.24	1913.81	2004.67	2384.27
	<b>Prediction Time (Seconds)</b>				
LSTM	2.188	2.561	2.620	2.671	3.263
GPVAR	0.302	0.410	0.665	0.757	0.986
TFT	0.129	0.145	0.173	0.184	0.2284

reliability. Finally, TFT shows a faster prediction within 0.172 to 0.22 seconds for all data lengths, which is due to a lower complexity during inference. In TFT, complex operations such as inversion of covariance matrices are not required during prediction, unlike in GPVAR. The sequential processing of LSTM, along with the maintenance and updating of hidden states, slightly affects the prediction time.

Table 4 shows the effect of increasing complexity with a growing number of tenants (i.e., number of input time series in the multivariate technique) on training and prediction time. LSTM requires a shorter training time, regardless of the increasing number of tenants for all three forecast models, ranging from 271.2 seconds with two tenants to 855.05 seconds with 29 tenants. However, the performance of LSTM is not efficient due to its limited ability to process complex data. Among the probabilistic models, GPVAR takes longer to train than TFT, approximately 979.81 seconds, for a small number of tenants. However, as the number of tenants increases, TFT appears to take longer to train the model, as the computational complexity increases and the effort required to analyze the temporal relationships increases. Although the TFTs are computationally intensive, they slightly outperform the GPVAR in terms of performance, as can be seen in Table 4, due to effective attention mechanisms and dynamic feature selection. TFT exhibits superior performance compared to other models, with low prediction time, regardless of the increasing number of tenants. It takes about 0.1 to 0.2 seconds to predict future PRB demands as it processes the inputs in parallel through trained attention mechanisms. GPVAR requires a moderate amount of time, approximately 0.3 to 0.9 seconds, with a balanced complexity and performance. In contrast, LSTM requires more time, about 2.1 to 3.2 seconds, for prediction due to the sequential processing of the inputs.

Multivariate probabilistic models such as GPVAR and TFT are extremely effective in capturing hidden relationships to enable reliable predictions. TFT has demonstrated better performance in predicting PRB requirements and handling uncertainty, albeit with increased complexity at higher tenant counts. GPVAR, on the other hand, shows competitive performance at moderate complexity and allows for further analysis in terms of rank-specific behaviors, as shown in Figure 6 and Table 4.



**FIGURE 6. Training loss for GPVAR.**

**E. HIGH-DIMENSIONAL MULTIVARIATE FORECASTING ANALYSIS WITH LOW-RANK GAUSSIAN COPULA PROCESS**

The ranks determine the dimensions of the latent space, which in turn approximates the covariance matrix of the multivariate Gaussian process in GPVAR estimators. A low rank reduces the complexity of the model, but higher ranks are capable of capturing the temporal relationships in the data more efficiently. Analyzing and evaluating the trade-off between the model’s complexity and performance scalability through different ranks enables a better understanding of how to optimize its use in real-world scenarios.

Figure 6 shows the effect of GPVAR ranks on training time and training loss. Training loss decreases and converges with increasing training time, irrespective of changing ranks. Rank 28, which is nearer to the number of series, i.e.,  $K = 29$  tenants used in this work, shows better performance and effective model learning, unlike Ranks 1 and 8, which show the highest training loss. As the rank reaches 29, the performance becomes inconsistent, and the complexity increases due to high training time. Improvement in training loss with rank comes at the cost of increased training time due to the increased computational effort required. Lower ranks stabilize faster but at higher loss values. This can be interpreted better from Table 5.

Table 5 shows a quantitative analysis of the effect of ranks on the performance of GPVAR using various metrics, such as NLL, RMSE, MAE, MAPE, ND and CRPS. It can be seen that the performance is better at rank 28 with the lowest test NLL of 231.34, RMSE of 20.017, MAPE of 4.06, MAE of 9.387, ND of 15.31 and a competitive Train NLL loss and CRPS of 466.81 and 0.37, respectively. Lower ranks, such as Rank 1, show higher Test NLL (298.62), RMSE (24.69), and MAE (11.69), indicating weaker performance in generalizing to test data. With increasing rank, both the prediction accuracy and the probabilistic score improve noticeably, which is reflected in the reduced values for Test NLL, RMSE and CRPS. However, while Rank 28 stands out for its overall performance, it also achieves the lowest Train NLL (466.81), suggesting that it may better capture certain aspects of the probabilistic structure of the data. Overall, increasing the rank allows GPVAR to better model complex temporal relationships, leading to improved prediction accuracy and

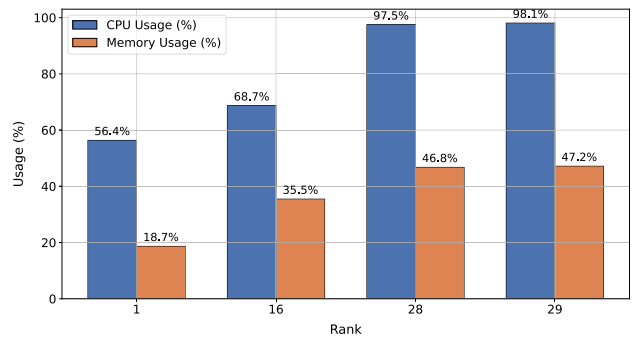
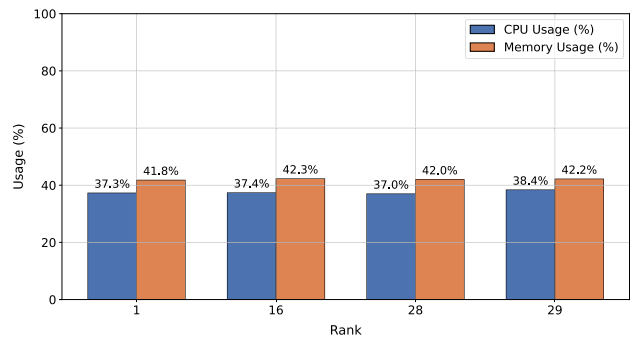
**TABLE 5.** Comparison of the performance of GPVAR for different ranks.

Rank	Train NLL	Test NLL	RMSE	MAE	MAPE	Normalized Deviation	CRPS
1	488.32	298.62	24.69	11.69	4.88	18.78	0.39
2	470.21	272.102	23.5	10.42	4.67	17.76	0.38
5	469.88	275.60	24.36	11.55	4.82	18.05	0.39
10	477.47	285.97	23.53	10.87	4.59	17.44	0.38
16	472.00	252.06	21.07	10.17	4.21	16.32	0.38
20	470.47	237.66	21.36	10.02	4.11	16.07	0.38
24	473.52	235.34	21.14	9.95	4.09	15.97	0.38
26	469.30	239.73	21.93	9.89	4.07	15.88	<b>0.37</b>
28	<b>466.81</b>	<b>231.34</b>	<b>20.017</b>	<b>9.387</b>	<b>4.06</b>	<b>15.31</b>	<b>0.37</b>
29	467.27	240.61	21.55	10.26	4.28	16.47	<b>0.37</b>

robustness, albeit with diminishing returns as the rank approaches its maximum (Rank 29). Ranks 1 and 10 show the worst performance across all metrics, indicating weaker performance and PRB demand prediction. Increasing the rank clearly improves accuracy and other metrics. However, when the rank reaches the maximum or is very close to the number of input data series, the performance starts to decrease due to the high complexity in capturing more intense patterns. These variations, as shown in both Figure 6 and Table 5, clearly highlight the effect of changing rank on model accuracy and computational complexity. The process of appropriate rank selection can also be automated by enabling early stopping, which progressively increases the rank and continuously monitors the estimator's performance. When model performance no longer improves, the training process is halted to prevent unnecessary computational overhead. This helps achieve an appropriate balance between computational cost and accuracy, making it suitable for real-time O-RAN deployment. Furthermore, the reliability of the selected rank regarding scalability, varying data distributions, and model generalisation can be further enhanced by evaluating and analysing the cross-validated NLL and statistical information criteria, such as the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC) [64].

Figures 7 and 8 show the CPU and Memory usage of GPVAR during training and prediction across various tensor ranks, respectively. It can be clearly seen in Figure 7 that increasing rank leads to higher CPU usage, rising from 56.4% at rank 1 to 97.5–98.1% at ranks 28 and 29. This trend reflects the need for greater computational overhead during higher-rank tensor operations in GPVAR training. Similarly, memory consumption increases from 18.7% at rank 1 to approximately 47% at rank 29, as larger low-rank covariance structures and parameters must be stored as the rank increases. This experiment clearly emphasizes the fact that changing rank directly affects training complexity. This makes it crucial to select an appropriate rank that provides a perfect balance between computational complexity and estimator performance.

On the contrary, predictions are less resource-intensive than training, irrespective of changing ranks, as noticed in Figure 8. CPU usage remains between 37% and 38%, while memory usage varies moderately around 42%. This behavior

**FIGURE 7.** CPU and Memory usage during training with GPVAR using different Ranks.**FIGURE 8.** CPU and Memory usage during prediction of PRBs with GPVAR using different Ranks.

is as expected during inference of future resource demands, as the prediction horizon is fixed to 48 hours and covariance parameters are precomputed during the training phase. Such consistency in computational needs during resource prediction using trained estimators makes it perfectly suitable to be used on edge or Near-RT RIC to make delay-sensitive decisions regarding O-RAN resource management.

#### F. LIMITATIONS AND DEPLOYMENT CONSIDERATIONS

From a model-level perspective, the use of GPVAR and TFT for resource provisioning in O-RAN clearly introduces a tradeoff between computational overhead, memory requirements, and latency. The input Data processing and probabilistic formulation in GPVAR depend on the evaluation

and low-rank approximation of the covariance matrix, which scale with increasing input dimensions. Model training remains sensitive to appropriate rank selection, even with low-rank copula approximations, as the number of series increases, both in memory usage and computational overhead [13]. Similarly, TFT achieves better accuracy when compared to LSTM baseline models due to gating and attention mechanisms, but at the cost of higher memory usage and parameter count. When deployed on an edge network with modest CPU capacity, this can lead to increased latency and may require more energy to process the input effectively. Moreover, with smaller datasets, regularization of TFT becomes crucial to avoid overfitting [43]. GPVAR is well-suited when calibrated uncertainty estimates are required, whereas TFT provides higher accuracy and faster inference on capable software. To summarize, selecting appropriate forecasting estimators based on hardware, latency requirements, use case, and available resources is crucial.

From a system-level perspective, the forecasting model's operations are constrained by the O-RAN stack. Non-RT RIC is suitable for training heavier inference workflows, long-term resource allocation predictions and dynamic policy generation, while, on the contrary, Near-RT RICs handle time-critical functions. Near-RT RICs on the edge would impose more CPU and memory restrictions on time-bound applications. Additionally, Containerization and orchestration of applications introduce scheduling overheads, which are crucial to consider during model computation [2], [43]. Prior analyses of cloud-native O-RAN deployments have shown that factors such as multi-tenancy, telemetry cost, and adaptability should be considered during the design of xApp/rApp applications. Some studies have also highlighted that failing to account for reproducibility and control-loop stability in xApp/rApp would result in complex operation of AI/ML in practice [45], [48]. Consequently, the appropriate choice of model configuration parameters (i.e., batch size, sequence length, etc.), depending on the available computational resources, target RIC, and monitoring limits, is also important for the smoother operation of xApp/rApp functions.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we introduced a containerized resource allocation rApp that leverages multivariate forecasting models such as GPVAR and TFT for effective resource provisioning and network analysis in O-RAN. Integrating these cloud-native solutions with Swagger API bridges the gap between theoretical insights and real-world applications by promoting simplified user interaction, seamless deployment, and service exposure. Furthermore, the performance estimates are optimized by selecting the optimal hyperparameters using AutoML techniques such as Optuna. The paper also provides brief comparative evaluations of traditional and probabilistic estimators in terms of error metrics such as MSE, MAE, and MAPE and their computational complexity through training

and prediction time. The accuracy, robustness, scalability, and flexibility of the forecasting technique are quantified using these metrics to efficiently fulfill the dynamic network requirements. In addition, GPVAR is explored in detail across different ranks to understand its impact on model complexity and optimization for adequate resource provisioning. The performance analysis showed that probabilistic estimators outperformed LSTM with respect to RMSE, MAE, and MAPE, achieving the lowest errors of 14.38, 7.967, and 3.293, respectively, using TFT. LSTMs show better performance in terms of computational requirements with a training time of 855.05 seconds, but require more time in prediction as the test cases need to be processed serially. GPVAR demonstrated moderate performance across all metrics, particularly with a higher number of tenants. A brief analysis of GPVAR with alternating ranks revealed that rank 28 yielded the best results, achieving a perfect balance of losses and errors.

The study highlights the trade-off between complexity and prediction accuracy. Future work would involve further developing the rApp to make informed decisions about resource allocation through predicting and analyzing other radio and computational resources like PRB, throughput and CPU usage data simultaneously. This includes supporting intelligent resource sharing among multiple tenants, refining the rApp implementation to accommodate necessary hardware interfaces and software dependencies for seamless integration within the O-RAN architecture, and evaluating its performance under realistic deployment scenarios. Furthermore, forecasting accuracy and reliability will be enhanced by incorporating additional graph-based features that effectively capture the spatial correlations among neighbouring cells [65].

## APPENDIX

### INTERACTION WITH CONTAINERIZED RESOURCE ALLOCATION rApp USING SWAGGER API

In this paper, the containerized resource allocation rApp with integrated Swagger API provides a clear and structured way of interaction through a single interface, as shown in Figures 7 to 11. The interaction steps include:

1. Navigate to "Training"
2. Collect train and validation data
3. Select forecasting estimators and Number of Tenants
4. Model training
5. Save trained model
6. Navigate to "Prediction"
7. Collect validation and test data
8. Select forecasting estimator and Number of Tenants
9. Load saved model
10. Resource demand forecasting
11. Visualization and analysis of metrics like Training time, prediction time, and forecasting errors

Figure 9 shows the initial interface of Swagger API for resource allocation rApp, which has two main functionalities:

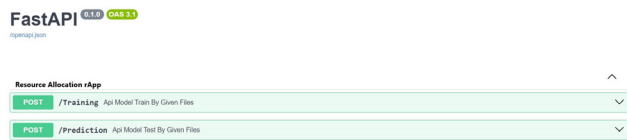


FIGURE 9. Swagger initial interface.

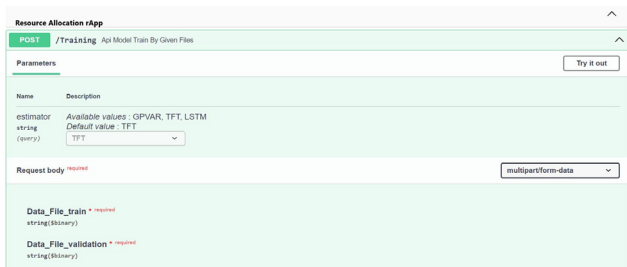


FIGURE 10. Swagger training functionality.

Training and Prediction. The functionalities and endpoints are clearly structured to enable seamless identification of available actions, understanding of requirements, and use of the API. Each function and its endpoint clearly describe the required input parameters and the expected output metrics. Figure 10 shows the initiation of the endpoint of the training function. It allows users to specify model-specific parameters such as the estimator type and the number of tenants, i.e. the number of input series, as well as pre-processed PRB history training and validation data. Then the training POST API is triggered to perform the model training with a selected estimator based on the uploaded history information. The training API ends its process after storing the trained prediction model.

Figure 11 shows the Prediction API, which is used for the evaluation of the trained estimators. Like the training API, the prediction function also expects input parameters such as the estimator type, the number of tenants and the data set for validation and testing. As soon as the prediction API is triggered, the container running in the backend automatically loads the required model from the corresponding location. This functionality is essential in O-RAN for analyzing feedback on model performance efficiency under dynamic and heterogeneous network conditions. Training and prediction APIs can also be performed in parallel if trained models are already available for the corresponding parameters. This can help users test multiple combinations of data sets and configurations without creating bottlenecks in fine-tuning the forecasting models and identifying areas for improvement.

Figures 12 and 13 show the results of the interaction with the training and the prediction API, where the estimator type is TFT and the number of tenants is set to 29. Figure 12 shows the time required for the TFT estimator to train the model with multivariate historical PRB time series data. In the future, further model-specific detailed parameters, such as hidden size, dropout rates, context length, etc., can

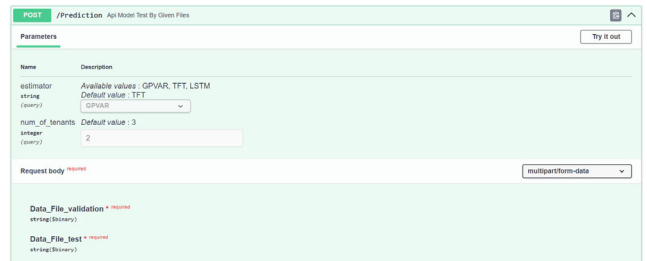


FIGURE 11. Swagger testing functionality.



FIGURE 12. Swagger training analysis of TFT.



FIGURE 13. Swagger prediction analysis of TFT.

also be included in the functionalities of Swagger API to ensure model optimization for specific network requirements and to obtain robust solutions under complex scenarios. Customization and organization of actions are easier to handle in this API. Finally, the analysis of the prediction can be seen in Figure 13, which shows the prediction time and error metrics such as MAE and RMSE for the TFT estimator. This helps users analyze model performance in terms of accuracy, error magnitude, and computational complexity.

To summarize, the management of API is simplified and the deployment of rApp is reliable and consistent thanks to the integration of Swagger with Docker. Such a centralized platform for interaction with rApp helps researchers to focus on improving core functions. The visualization of real-time results and immediate feedback on interactions ensures operational efficiency and rapid progress. In addition, they promote transparency so that users (such as researchers, developers or network operators) can efficiently evaluate and analyze the system. They also ensure agility, flexibility and scalability in the O-RAN ecosystem and easily overcome the difficulties arising from the evolution of telecommunication networks.

REFERENCES

- [1] B. N. Khalaf, W. H. Ali, R. S. Alhumaima, and H. A. J. Alshamary, "Delay aware resource allocation in ORAN through network optimization," *IET Wireless Sensor Syst.*, vol. 14, no. 6, pp. 528–538, Dec. 2024.
- [2] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," vol. 25, no. 2, pp. 1376–1411, 2nd Quart., 2023.

- [3] K. Qiao, H. Wang, W. Zhang, D. Yang, Y. Zhang, and N. Zhang, "Resource allocation for network slicing in open RAN: A hierarchical learning approach," *IEEE Trans. Cognit. Commun. Netw.*, vol. 11, no. 4, pp. 2584–2600, Aug. 2025.
- [4] V. Kasuluru, L. Blanco, E. Zeydan, A. Bel, and A. Antonopoulos, "Enhancing cloud-native resource allocation with probabilistic forecasting techniques in O-RAN," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit)*, Jun. 2024, pp. 1–6.
- [5] V. Kasuluru, L. Blanco, M. Á. Vázquez, C. J. Vaca-Rubio, and E. Zeydan, "Minimizing power consumption under SINR constraints for cell-free massive MIMO in O-RAN," in *Proc. 20th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2024, pp. 1–7.
- [6] V. Kasuluru, L. Blanco, C. J. Vaca-Rubio, and E. Zeydan, "On the impact of PRB load uncertainty forecasting for sustainable open RAN," in *Proc. IEEE 35th Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2024, pp. 1–7.
- [7] C.-N. Nhu and M. Park, "Dynamic network slice scaling assisted by attention-based prediction in 5G core network," *IEEE Access*, vol. 10, pp. 72955–72972, 2022.
- [8] Y. Xu, G. Gui, H. Gacanin, and F. Adachi, "A survey on resource allocation for 5G heterogeneous networks: Current research, future trends, and challenges," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 2, pp. 668–695, 2nd Quart., 2021.
- [9] N. Sharma and K. Kumar, "Resource allocation trends for ultra dense networks in 5G and beyond networks: A classification and comprehensive survey," *Phys. Commun.*, vol. 48, Oct. 2021, Art. no. 101415.
- [10] D. Dulas, J. Witulska, A. Wyłomańska, I. Jabłoński, and K. Walkowiak, "Data-driven model for sliced 5G network dimensioning and planning, featured with forecast and 'what-if' analysis," *IEEE Access*, vol. 12, pp. 50067–50082, 2024.
- [11] M. Q. Hamdan, H. Lee, D. Triantafyllopoulou, R. Borralho, A. Kose, E. Amiri, D. Mulvey, W. Yu, R. Zitouni, R. Pozza, B. Hunt, H. Bagheri, C. H. Foh, F. Heliot, G. Chen, P. Xiao, N. Wang, and R. Tafazolli, "Recent advances in machine learning for network automation in the O-RAN," *Sensors*, vol. 23, no. 21, p. 8792, Oct. 2023.
- [12] V. Kasuluru, L. Blanco, and E. Zeydan, "Enhancing open RAN operations: The role of probabilistic forecasting in network analysis," *IEEE Trans. Netw. Service Manage.*, vol. 22, no. 4, pp. 3676–3691, Aug. 2025.
- [13] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus, "High-dimensional multivariate forecasting with low-rank Gaussian copula processes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 6824–6834.
- [14] A. Perveen, R. Abozariba, M. Patwary, and A. Aneiba, "Dynamic traffic forecasting and fuzzy-based optimized admission control in federated 5G-open RAN networks," *Neural Comput. Appl.*, vol. 35, no. 33, pp. 23841–23859, Nov. 2023.
- [15] O-RAN Alliance, "O-ran: Towards an open and smart ran," Feb. 2020. Accessed: Dec. 5, 2023. [Online]. Available: <https://www.o-ran.org/resources>
- [16] (Feb. 2020). *O-RAN Use Cases and Deployment Scenarios Whitepaper*. Accessed: Dec. 5, 2023. [Online]. Available: <https://www.o-ran.org/resources>
- [17] A. Lacava, L. Bonati, N. Mohamadi, R. Gangula, F. Kaltenberger, P. Johari, S. D'Oro, F. Cuomo, M. Polese, and T. Melodia, "DApps: Enabling real-time AI-based open RAN control," 2025, *arXiv:2501.16502*.
- [18] F. Khodadadi, A. V. Dastjerdi, and R. Buyya, "Simurgh: A framework for effective discovery, programming, and integration of services exposed in IoT," in *Proc. Int. Conf. Recent Adv. Internet Things (RIoT)*, Apr. 2015, pp. 1–6.
- [19] *Swagger API Documentation*, Swagger, 2020. [Online]. Available: <https://swagger.io>
- [20] H. Liu, J. Zong, Q. Wang, Y. Liu, and F. Yang, "Cloud native based intelligent RAN architecture towards 6G programmable networking," in *Proc. 7th Int. Conf. Comput. Commun. Syst. (ICCCS)*, Apr. 2022, pp. 623–627.
- [21] A. Mohammadi and N. Nikaein, "Athena: An intelligent Multi-x cloud native network operator," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 2, pp. 460–472, Feb. 2024.
- [22] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 4, pp. 2595–2621, 4th Quart., 2018.
- [23] E. Tuna and A. Soysal, "Multivariate, multi-step, and spatiotemporal traffic prediction for NextG network slicing under SLA constraints," 2023, *arXiv:2309.03898*.
- [24] W. Wang, C. Zhou, H. He, W. Wu, W. Zhuang, and X. Shen, "Cellular traffic load prediction with LSTM and Gaussian process regression," in *Proc. ICC - IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.
- [25] C. Sun, U. Pawar, M. Khoja, X. Foukas, M. K. Marina, and B. Radunovic, "SpotLight: Accurate, explainable and efficient anomaly detection for open RAN," in *Proc. 30th Annu. Int. Conf. Mobile Comput. Netw.*, Dec. 2024, pp. 923–937.
- [26] K. Ali and M. Jammal, "ML-based dynamic scaling and traffic forecasting for 5G O-RAN," in *Proc. IEEE Int. Conf. Dependable, Autonomic Secure Comput., Int. Conf. Pervasive Intell. Comput., Int. Conf. Cloud Big Data Comput., Int. Conf. Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Nov. 2023, pp. 0444–0451.
- [27] W. Benrhaim and A. S. Hafid, "Bayesian networks based reliable broadcast in vehicular networks," *Veh. Commun.*, vol. 21, Jan. 2020, Art. no. 100181.
- [28] K. Chen, Q. Kong, Y. Dai, Y. Xu, F. Yin, L. Xu, and S. Cui, "Recent advances in data-driven wireless communication using Gaussian processes: A comprehensive survey," *China Commun.*, vol. 19, no. 1, pp. 218–237, Jan. 2022.
- [29] M. Li, Y. Wang, Z. Wang, and H. Zheng, "A deep learning method based on an attention mechanism for wireless network traffic prediction," *Ad Hoc Netw.*, vol. 107, Oct. 2020, Art. no. 102258.
- [30] H. Mahmoud, M. N. I. Farooqui, D. Mi, L. Guo, C. Lu, Y. Gan, Z. Gao, Z. Wang, and Y. Zhang, "Data-driven approach for optimising resource allocation of O-RAN networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2024, pp. 1–6.
- [31] M. M. H. Qazzaz, Ł. Kulacz, A. Kliks, S. A. Zaidi, M. Dryjanski, and D. McLernon, "Machine learning-based xApp for dynamic resource allocation in O-RAN networks," 2024, *arXiv:2401.07643*.
- [32] W. Jiang, Y. Zhan, G. Zeng, and J. Lu, "Probabilistic-forecasting-based admission control for network slicing in software-defined networks," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 14030–14047, Aug. 2022.
- [33] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, and H. Zhang, "Intelligent 5G: When cellular networks meet artificial intelligence," *IEEE Wireless Commun.*, vol. 24, no. 5, pp. 175–183, Oct. 2017.
- [34] F. Aslan, G. Iosifidis, J. A. Ayala-Romero, A. Garcia-Saavedra, and X. Costa-Perez, "Fair resource allocation in virtualized O-RAN platforms," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 8, no. 1, pp. 1–34, 2024.
- [35] B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *Int. J. Forecasting*, vol. 37, no. 4, pp. 1748–1764, Oct. 2021.
- [36] N. Kazemifard and V. Shah-Mansouri, "Minimum delay function placement and resource allocation for open RAN (O-RAN) 5G networks," *Comput. Netw.*, vol. 188, Apr. 2021, Art. no. 107809.
- [37] H. Ahmadi, M. Rahmani, S. B. Chetty, E. E. Tsiropoulou, H. Arslan, M. Debbah, and T. Q. S. Quek, "Toward sustainability in 6G and beyond: Challenges and opportunities of open RAN," *IEEE Commun. Standards Mag.*, vol. 9, no. 3, pp. 126–135, Sep. 2025.
- [38] S. Maxenti, S. D'Oro, L. Bonati, M. Polese, A. Capone, and T. Melodia, "ScalO-RAN: Energy-aware network intelligence scaling in open RAN," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2024, pp. 891–900.
- [39] H. X. Nguyen, K. Sun, D. To, Q.-T. Vien, and T. A. Le, "Digital twin for O-RAN toward 6G," *IEEE Commun. Mag.*, vol. 63, no. 3, pp. 174–181, Mar. 2025.
- [40] M. Bordin, A. Lacava, M. Polese, S. Satish, M. A. Nittoor, R. Sivaraj, F. Cuomo, and T. Melodia, "Design and evaluation of deep reinforcement learning for energy saving in open RAN," in *Proc. IEEE 22nd Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2025, pp. 1–6.
- [41] B. Brik, H. Chergui, L. Zanfi, F. Devoti, A. Ksentini, M. S. Siddiqui, X. Costa-Pérez, and C. Verikoukis, "Explainable AI in 6G O-RAN: A tutorial and survey on architecture, use cases, challenges, and future research," *IEEE Commun. Mag.*, vol. 27, no. 5, pp. 2826–2859, Oct. 2025.
- [42] N. Piovesan, D. López-Pérez, A. De Domenico, X. Geng, and H. Bao, "Power consumption modeling of 5G multi-carrier base stations: A machine learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2023, pp. 3633–3638.

- [43] N. Li, X. Xu, Q. Sun, J. Wu, Q. Zhang, G. Chi, and N. Sprecher, "Transforming the 5G RAN with innovation: The confluence of cloud native and intelligence," *IEEE Access*, vol. 11, pp. 4443–4454, 2023.
- [44] D. Kim, E. Lee, and S. Pack, "Design and implementation of cloud-native open RAN testbed," in *Proc. 15th Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2024, pp. 2167–2168.
- [45] J. Baranda, A. Bel, S. Barrachina-Muñoz, M. Payaró, and J. Mangues-Bafalluy, "Distributed sequential cloud-native deployment of an end-to-end 5G network with O-RAN functions," in *Proc. 15th Int. Conf. Netw. Future (NoF)*, Oct. 2024, pp. 142–144.
- [46] I. Vilá, O. Sallent, and J. Pérez-Romero, "On the implementation of a reinforcement learning-based capacity sharing algorithm in O-RAN," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2022, pp. 208–214.
- [47] B. De, *Api management: An Architect's Guide to Developing and Managing Apis for Your Organization*, 2017. [Online]. Available: <https://link.springer.com/book/10.1007/979-8-8688-0054-2>
- [48] M. Hoffmann, S. Janji, A. Samorzewski, Ł. Kułacz, C. Adamczyk, M. Dryjański, P. Kryszkiewicz, A. Kliks, and H. Bogucka, "Open RAN xApps design and evaluation: Lessons learnt and identified challenges," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 2, pp. 473–486, Feb. 2024.
- [49] T. Hagelberg, "Development of a serverless restful API," 2023. [Online]. Available: <https://www.theseus.fi/handle/10024/800866>
- [50] K. Ali and M. Jammal, "Proactive VNF scaling and placement in 5G O-RAN using ML," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 1, pp. 174–186, Feb. 2024.
- [51] Q. Wang, Y. Liu, Y. Wang, X. Xiong, J. Zong, J. Wang, and P. Chen, "Resource allocation based on radio intelligence controller for open RAN toward 6G," *IEEE Access*, vol. 11, pp. 97909–97919, 2023.
- [52] P. Song, H. Peng, and X. Zhang, "A micro-service approach to cloud native RAN for 5G and beyond," *IEEE Access*, vol. 11, pp. 130257–130271, 2023.
- [53] K. Alam, M. Asif Habibi, M. Tammen, D. Krummacker, W. Saad, M. Di Renzo, T. Melodia, X. Costa-Pérez, M. Debbah, A. Dutta, and H. D. Schotten, "A comprehensive tutorial and survey of O-RAN: Exploring slicing-aware architecture, deployment options, use cases, and challenges," 2024, *arXiv:2405.03555*.
- [54] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux j*, vol. 2014, no. 239, p. 2, 2014.
- [55] J. Ponelet and L. Rosenstock, *Designing APIs With Swagger and OpenAPI*. New York, NY, USA: Simon and Schuster, 2022.
- [56] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Phys. D, Nonlinear Phenomena*, vol. 404, Mar. 2020, Art. no. 132306.
- [57] G. Elidan, "Copulas in machine learning," in *Proc. Workshop Held Cracow Copulae Math. Quantum Finance*, Jul. 2013, pp. 39–60.
- [58] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *Int. J. Forecasting*, vol. 36, no. 3, pp. 1181–1191, Jul. 2020.
- [59] V. Kasuluru, L. Blanco, and E. Zeydan, "On the use of probabilistic forecasting for network analysis in open RAN," in *Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom)*, Sep. 2023, pp. 258–263.
- [60] J. Berrisch and F. Ziel, "Multivariate probabilistic CRPS learning with an application to day-ahead electricity prices," *Int. J. Forecasting*, vol. 40, no. 4, pp. 1568–1586, Oct. 2024.
- [61] F. Rezazadeh, L. Zanzi, F. Devoti, H. Chergui, X. Costa-Perez, and C. Verikoukis, "On the specialization of FDRL agents for scalable and distributed 6G RAN slicing orchestration," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3473–3487, Mar. 2023.
- [62] Z. Deng, D. Zhou, Z. Kang, and H. Dong, "Deep learning-based dynamic forecasting method and application for ultra-deep fractured reservoir production," *Frontiers Energy Res.*, vol. 12, Mar. 2024, Art. no. 1369606.
- [63] R. N. Muniz, W. G. Buratto, A. Nied, R. Cardoso, E. C. Finardi, and G. V. González, "Natural inflow forecasting in hydroelectric power plants using hypertexted TFT with Hodrick–Prescott filter," 2025. [Online]. Available: [https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=5159166](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5159166)
- [64] H. Hung, S.-Y. Huang, and C.-K. Ing, "A generalized information criterion for high-dimensional PCA rank selection," *Stat. Papers*, vol. 63, no. 4, pp. 1295–1321, Aug. 2022.
- [65] H. Chen, R. A. Rossi, K. Mahadik, S. Kim, and H. Eldardiry, "Graph deep factors for probabilistic time-series forecasting," *ACM Trans. Knowl. Discovery Data*, vol. 17, no. 2, pp. 1–30, Apr. 2023.



**VAISHNAVI KASULURU** received the B.Sc. degree in electronics and communication engineering from the SSN College of Engineering, affiliated with Anna University, Chennai, India, in 2016, and the M.Sc. degree in information and communication engineering from TU Darmstadt, Germany, in 2020. She is currently pursuing the Ph.D. degree with the SRCOM Research Unit, CTTC, Barcelona, Spain. She is a Researcher with the SRCOM Research Unit, CTTC, where she is involved in an EU-funded project called SEMANTIC. Her research interests include developing optimization and machine learning algorithms to solve challenging real-world telecommunication problems.



**LUIS BLANCO** received the M.Sc. degree in telecommunications engineering, the M.Sc. degree in research on information and communications technologies (MERIT), and the Ph.D. degree in telecommunications engineering from the Polytechnic University of Catalonia (UPC), Spain, in 2006, 2012, and 2017, respectively, and the M.B.A. degree from IDE-CESEM, Instituto de Directivos de Empresa (Madrid). In 2007, he joined the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), where he is currently a Researcher. He has actively participated in more than 20 research projects funded by public and private entities, including EC-funded projects, national projects, and industrial projects with top-tier international companies. His research interests include optimization and artificial intelligence (AI) for wireless communication systems in beyond 5G and 6G integrated networks.



**CRISTIAN J. VACA-RUBIO** (Member, IEEE) received the Telematics Engineering and master's degrees in telematics and telecommunication networks from the University of Malaga (UMA), Malaga, Spain, in 2018, and 2019, respectively, and the Ph.D. degree in wireless communications from the Connectivity Section, Electronic Systems Department, Aalborg University, Aalborg, Denmark, in 2023. He is currently with the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Barcelona, Spain. In parallel to his telematics and master's studies, he was a Research Assistant with the Communication Engineering Department, University of Malaga, from February 2018 until July 2019, in a joint project with DEKRA GmbH. For his Ph.D. studies, he received the Marie Curie Fellowship as an Early Stage Researcher (ESR) in the H2020 ITN Wind Mill Project, where he joined in September 2019. During his Ph.D. studies, he was a Visiting Researcher at the Mitsubishi Electric Research Laboratories (MERL), Cambridge, USA, for a period of nine months. He is the author of a submitted patent along with MERL co-authors. His research interests include wireless communications and sensing, machine learning, massive MIMO, mobile networks, telematics, and applied statistics in video streaming. He received the IEEE International Conference on Acoustics, Speech, and Signal Processing Top 3% Paper Award, in 2023.



MonB5G Project, from 2021 to 2023. His research interests include telecommunications, data engineering, and network security.

**ENGİN ZEYDAN** (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, USA, in February 2011. He is currently a Senior Researcher with the Services as Networks (SaS) Research Unit, CTTC, Barcelona, Spain. He is the Project Coordinator of the Horizon Europe UNITY-6G Project (2025–2027). Previously, he was the Project Coordinator of the H2020



coordinated two undergraduate subjects. In 2023, he joined as a Researcher at the Services as Networks (SaS) Research Unit, Centre Tecnològic de Telecomunicacions de Catalunya (CTTC). His research interests include open RAN, optimization of 5G and 6G networks, mostly focused on the testbed implementations.



**ANGELOS ANTONOPOULOS** (Senior Member, IEEE) received the Ph.D. degree from the Polytechnic University of Catalonia (UPC), in 2012. He is currently the R&I Director of Nearby Computing, leading the innovation efforts of the company in the edge/cloud orchestration domain. He has (co-)authored over 130 publications on various topics, including 5G/6G architectures, energy-efficient network planning, and zero-touch service management.

...