Adaptive Retraining of AI/ML Model for Beyond 5G Networks: A Predictive Approach

Venkateswarlu Gudepu[$], Venkatarami Reddy Chintapalli[†], Piero Castoldi[•], Luca Valcarenghi[•], Bheemarjuna Reddy Tamma[#], Koteswararao Kondepu[$]
[$]Indian Institute of Technology Dharwad, India
[†]National Institute of Technology Calicut, Calicut, India
[•]Scuola Superiore Sant'Anna, Pisa, Italy
[#]Indian Institute of Technology Hyderabad, India
e-mail:212011003@iitdh.ac.in, venkataramireddyc@nitc.ac.in, k.kondepu@iitdh.ac.in

***Abstract*—Beyond fifth-generation (B5G) networks (namely 6G) aim to support high data rates, low-latency applications, and massive machine communications. Integrating Artificial Intelligence (AI) and Machine Learning (ML) models are essential for addressing the network's increasing complexity and dynamic nature. However, dynamic service demands of B5G cause the AI/ML models performance degradation, resulting in violations of Service Level Agreements (SLA), over- or under-provisioning of resources, etc. To address the performance degradation of the AI/ML models, retraining is essential. Existing threshold and periodic retraining approaches have potential disadvantages such as SLA violations and inefficient resource utilization for setting a threshold parameter in a dynamic environment. This paper presents a novel algorithm that predicts when to retrain AI/ML models using an unsupervised classifier. The proposed predictive approach is evaluated for a Quality of Service (QoS) prediction use case on the Open RAN Software Community (OSC) platform and compared to the threshold approach. The results show that the proposed predictive approach outperforms the threshold approach.**

***Index Terms*—Prediction, AI/ML, Retraining, 5G and Beyond.**

I. Introduction

The envisioned Beyond 5G (B5G) networks have brought a major transformation in the networking industry. With the advent of new use cases, such as ultra-reliable low latency communications (uRLLC), massive machine-type communications (mMTC), and enhanced mobile broadband (eMBB), network operators need to accommodate these demands while maintaining high quality of service (QoS) [1]. In view of the B5G network complexity, *intelligence* has become an essential component of it to enhance the network performance. The ability of Artificial Intelligence and Machine Learning (AI/ML) algorithms to handle complex network architectures makes them suitable and allows them to make more intelligent decisions (e.g., resource allocation based on predicted future traffic patterns).

Many consortia, such as European Telecommunications Standards Institute (ETSI), Open Radio Access Network (O-RAN) Alliance, Telecom Infra Project (TIP), 5G Infrastructure Association (5G-IA), and 6G Smart Networks and Services Industry Association (6G-IA) are working towards enabling *intelligence* in the B5G. The ETSI based zero-touch network service management (ZSM) framework aims to automate and orchestrate the network service management and eliminate manual intervention through the adaption of *intelligence* [2]. The ZSM framework can be realized through the O-RAN Alliance architecture [3], where *intelligence* is enabled through two logical RAN Intelligent Controllers (RICs): Non-real time RIC (Non-RT RIC) and Near-real time RIC (Near-RT RIC). The Non-RT RIC operates the use cases which have granularity $> 1\ sec$, whereas the Near-RT RIC manages the use cases that have a granularity between $10\ ms$ and $1\ sec$. The AI/ML models run at the Non-RT and Near-RT RIC are referred to as rApps and xApps, respectively.

The use of AI/ML models through RICs improves the performance of RAN, however, several challenges need to be addressed. The performance of AI/ML models greatly relies on the data used for training. The dynamic service demands from users can cause AI/ML model performance degradation, leading to resource allocation and utilization issues [4]. Here, over-provisioning of resources (e.g., bandwidth, storage, etc.) results in the waste of resources, network congestion, and an increase in operational expenditure (OPEX), while under-provisioning of resources results in reduced network performance, poor QoS, longer response time, and network outages.

The above-specified factors influence Service Level Agreements (SLAs) between service providers and users. For instance, in a smart ambulance use case exploiting B5G, the SLA defines a minimum and maximum throughput of $25\ Mbps$ and $400\ Mbps$, respectively, with a response time of less than $20\ ms$. However, model performance degradation can cause violation of the defined SLAs [5]. Therefore, it is crucial to address the model performance degradation in B5G.

Retraining the model with newly arrived user data can help to ensure the AI/ML model performance. In [6], the authors proposed a threshold approach to trigger the model retraining. This approach monitors the model performance metrics such as accuracy, precision, recall, or the F1 score and initiates retraining if the values fall below or exceed a predefined threshold. However, determining the appropriate threshold value poses significant challenges, as setting it too low may result in excessive computational costs from frequent retraining, while setting it too high may lead to a decline in model performance and

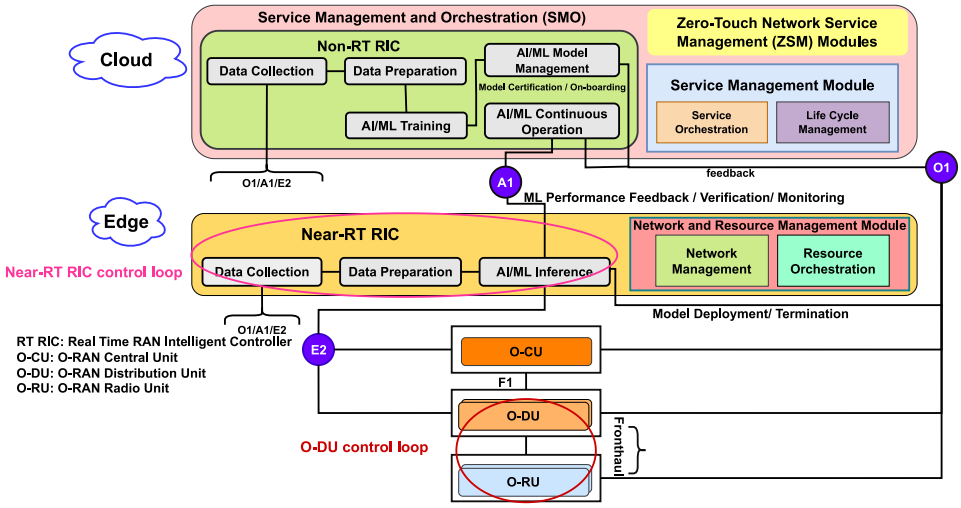

Figure 1. ZSM components realized through O-RAN architecture.

violate SLAs. These issues highlight the complexity of setting a threshold value in a highly dynamic environment. Another approach in [7] uses periodic retraining, where the models are updated at regular intervals regardless of fluctuations in user traffic or the current performance of the AI/ML model. The aforementioned threshold and periodic approaches for retraining an AI/ML model are susceptible to SLA violations during the retraining process, which is influenced by various factors such as the location of retraining (i.e., *cloud* or *edge*) and the time required for retraining the model.

The method of retraining an AI/ML model requires considering the location of the retraining either in the *cloud* or at the *edge*. The *cloud* provides advantages such as a larger pool of computational resources and the capability to process larger datasets. On the other end, retraining at the *edge* offers benefits such as reduced propagation latency by processing closer to the end-user. However, retraining at the *edge* may be constrained by the availability of computational resources, which indeed may increase the retraining time.

This paper presents a novel approach that predicts when to retrain by using an unsupervised classifier. This approach is designed to take into account the additional delays (location of retraining) involved in the retraining and replacement of the AI/ML model. By considering all relevant factors, the proposed approach reduces SLA violations and increases the efficiency of computational resources. The main contributions of this paper are summarized as follows: (i) mapping between the ZSM framework and the O-RAN architecture; (ii) integrating the proposed approach to predict when to retrain an AI/ML model; and (iii) evaluating the proposed approach by considering the Quality of Service (QoS) prediction use case over the O-RAN software community (OSC) platform [8].

II. System Model and Proposed Approach

This section describes the system model and the proposed approach to predict when to retrain an AI/ML model. Figure 1 shows the O-RAN architecture which is realized based on the ZSM concepts and principles [9]. ZSM aims at orchestrating network service provisioning, management, and scaling by minimizing human intervention.

The ZSM-E2E service management domain is adopted as the Service Management and Orchestration (SMO) module in O-RAN, which is responsible for hosting all service modules, management of network configurations, resource inventory, and policy making. The functionalities of ZSM modules like *intelligence*, *analytics*, and *orchestration* are realized as AI/ML model management services using control loops, assessing and improving the deployed services, and network management services in O-RAN, respectively. These modules can be realized through the Non-RT RIC in O-RAN.

Another component of the ZSM is *domain controller*, which takes care of resource management and it can be realized through Near-RT RIC in O-RAN, which controls radio resources through E2 control messages. Some common modules, such as domain data collection and services, are adopted through a common database between Non- and Near-RT RIC. Non-RT RIC runs in the *cloud*, whereas Near-RT RIC runs at the *edge* in the considered scenario. The AI/ML model management module in the Non-RT RIC is primarily responsible for selecting when to retrain an AI/ML model using the proposed approach, which predicts the requirement for retraining in advance based on the newly arrived user traffic.

The proposed approach predicts when to retrain the AI/ML model by exploiting an unsupervised classifier on the arrival of user traffic. The classifier observes the arriving user traffic and determines whether the incoming traffic belongs to the observed or is different. If the new traffic arrives over a certain period of time, then the proposed approach triggers retraining. Algorithm 1 depicts the proposed approach and Table I reports definitions of the parameters/variables used in it. The Algorithm 1 takes T_{ds}, T_{e2e}, ds, and an unsupervised classifier (C) as input parameters and predicts whether an AI/ML model retraining is required or not as an output. Here, T_{e2e} can be obtained from the SLAs between operators and users defined for various use cases [5]. The algorithm starts by initializing a variable called N_{co}, which represents the number of consecutive data chunks (i.e., a small set of data from ds) [10] required to

check on the incoming data stream ds, that has new data (see line 3). Here, the number of consecutive data chunks to be observed (N_{co}) is defined as the ratio between T_{ds} and T_{e2e}. This definition is derived after observing the convergence of the model's accuracy for various datasets with different values of T_{ds}. Noticed that the considered approach is working well for the datasets under consideration [11], [12].

The ds splits into consecutive chunks of data with a length of N_{ds} and the classifier (C) performs classification over each data sample in the data chunk (see lines 6-7). The classifier assigns $+1$ if the data sample in the data chunk belongs to the observed; otherwise assigns -1 to indicate it as the newly arrived user data. The classification labels (i.e., $+1$ or -1) of each data sample in the data chunk are stored in $Y_{predict}$. Our approach counts the number of new data samples classified under -1 (i.e., N_{cd}) from the $Y_{predict}$ for each data chunk (see lines 9-13). If the value of N_{cd} is at least one, then the algorithm will check over the next data chunk. If the number of consecutive data chunks that have new data (i.e., N_{td}) is equal to the N_{co}, then the proposed approach triggers AI/ML model retraining (see lines 14-21). During the retraining, the previously trained AI/ML model weights are updated based on the newly available data and deployed as a xApp to predict further.

Algorithm 1 Predict when to retrain an AI/ML model

1: $Input: T_{ds}, T_{e2e}, C, ds$
2: $Output$: Retrain or not
3: $N_{co} \leftarrow \lceil \frac{T_{ds}}{T_{e2e}} \rceil$
4: $N_{td} \leftarrow 0$
5: **while** data_available **do**
6: **for** $n \leftarrow 0$ to $\lfloor \frac{length(ds)}{N_{ds}} \rfloor$ **do**
7: $Y_{predict} \leftarrow C\ (ds[(N_{ds} * n)\ to\ (N_{ds} * (n+1) - 1)])$
8: $N_{cd} \leftarrow 0$
9: **for** $i \leftarrow 0\ to\ length(Y_{predict})$ **do**
10: **if** $Y_{predict}[i]\ is\ -1$ **then**
11: $N_{cd}++$
12: **end if**
13: **end for**
14: **if** $N_{cd} \geq 1$ **then**
15: $N_{td}++$
16: **else**
17: $N_{td} \leftarrow 0$
18: **end if**
19: **if** $N_{td}\ is\ N_{co}$ **then**
20: *Retrain the AI/ML Model*
21: **end if**
22: **end for**
23: **end while**

Table I
DESCRIPTION OF VARIABLES USED IN ALGORITHM

Acronym	Referring to / Definition
ds	Incoming data stream
T_{ds}	Time taken to transmit the considered data samples from ds
T_{e2e}	End-to-end delay of an application under consideration [5]
N_{co}	The number of consecutive data chunks to observe for the new data
N_{td}	To keep track of the number of data chunks that have new data
N_{ds}	Number of data samples considered in T_{ds}
N_{cd}	To count the number of new data samples that arrived in each data chunk
$Y_{predict}$	Stores the classifier output in a list (i.e., Y)

III. EXPERIMENTAL SETUP AND RESULTS

This section presents the experimental setup considered to evaluate the QoS prediction use case over the OSC RIC platform, followed by a discussion of obtained results.

The proposed approach is evaluated for *QoS prediction* use case that predicts the service quality (e.g., throughput) provided by the network providers to their subscribers [13]. Figure 2 shows the experimental setup used to evaluate the considered QoS prediction use case, where the proposed approach is deployed at Non-RT RIC and integrated with the OSC RIC platform.

We deployed the OSC Near-RT RIC framework (F-release), an E2 simulator, and various open interfaces [8]. Specifically, the Near-RT RIC framework is deployed as a Kubernetes pod — it is defined as a collection of containers inside a node of a Kubernetes cluster. In addition, the OSC Near-RT RIC components include an E2 manager, a routing manager, a subscription manager, an app manager, and a shared database (i.e., InfluxDB), along with open interfaces like A1, E2, and O1 as shown in Figure 2. As stated, all these components are deployed as microservices in a Kubernetes cluster. Whenever a xApp is onboarded into the Near-RT RIC, it writes its statistics into the common database (i.e., InfluxDB).

A key performance metric monitoring (KPMMON) xApp in the Near-RT RIC writes the statistics of each layer of the RAN into the InfluxDB database. It periodically writes the statistics of the data units used for each layer and an API function inside the database calculates cell throughput upon the data is updated. The onboarded QoS prediction (QP) xApp also writes the predicted QoS values into the InfluxDB. The status of the QP xApp onboarded into the Near-RT RIC is shown in Figure 3. The QP xApp is built using a long short-term memory (LSTM) model with three layers, each with 100 hidden units followed by a rectified linear unit (ReLU) as an activation function to forecast the QoS.

An OSC E2 simulator is composed of a 5G core, central unit (O-CU), distributed unit (O-DU), and a radio unit (O-RU) which communicate with the Near-RT RIC using E2 messages such as: (1) E2 setup request; (2) E2 setup subscription, and (3) E2 setup indication as shown in Figure 2. Here, an E2 simulator is connected by multiple user equipments (UEs) with different data rates configured through a customized JavaScript Object Notation (*json*) file. These *json* files are used for launching multiple UEs to generate various data rates.

Note that the current F-release of OSC does not provide a complete implementation of SMO and Non-RT RIC frame-

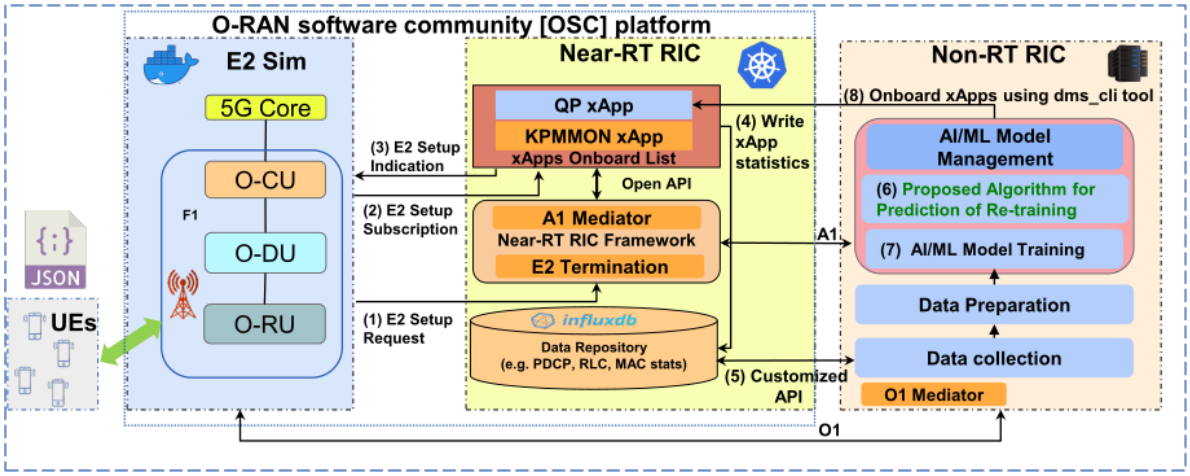

Figure 2. Experimental Setup

```
root@ric:/home/ric/appmgr/xapp_orchestrater/dev/xapp_onboarder# kubectl get pods -n ricxapp
NAME                                READY   STATUS    RESTARTS   AGE
ricxapp-hwxapp-684d8d675b-bsbgg     1/1     Running   0          36h
ricxapp-qp-5f6fc7b746-c4rbf         1/1     Running   0          22h
```

Figure 3. Status of the onboarded QP xApp in the Near-RT RIC.

works as of now. Thus, in order to evaluate the QoS prediction use case using the proposed approach, the missed functionalities in OSC platform are implemented and deployed in a bare metal server, for example, performing the off-line AI/ML models training and onboard them in the Near-RT RIC as a microservice [14] using the dms_cli tool. Moreover, the proposed approach is implemented as a REST API, and monitors changes in user traffic by accessing real-time data (i.e., throughput) from the InfluxDB database through a customized API.

The Near-RT RIC and E2 simulator are connected, and integrated with the proposed predictive approach — which used an unsupervised classifier to determine changes in user traffic and trigger retraining when necessary — deployed in Non-RT RIC to provide AI/ML model management functionalities such as data collection and preparation, AI/ML model training and its deployment.

In this paper, three different unsupervised classifiers are explored, such as the One-Class Support Vector Machine (OC-SVM), Isolation Forest (IF), and Local Outlier Factor (LOF) [15], to detect changes in user traffic. The performance of these classifiers is evaluated on the QoS dataset, which is used for building the QP xApp. The obtained results are presented in Table II. The LOF classifier outperformed the other two or performed equally in terms of accuracy, precision, recall, and F1-score. Hence, the LOF classifier is considered for detecting changes in user traffic. The LOF classifier calculates the local density deviation for each incoming user data sample and assigns "+1" or "-1" whether the data sample belongs to the trained data or not, respectively. The detailed description of metrics (e.g., accuracy, F1-score) is presented in [15] for interested readers.

The arrival of new user traffic can cause SLA violations and resource provisioning issues [16]. Thus, there is a strong need to detect changes in the incoming user data. The proposed predictive approach exploits unsupervised classifier (i.e, "+1", "-1") to detect the changes effectively in the newly arrival data as detailed in Algorithm 1. In addition, the threshold approach [6] is also used to detect the changes in newly arrived data. However, it has to admit the violations till the predefined threshold in order to trigger the retraining. The proposed predictive approach is compared with the threshold approach to detect the changes in the newly arrived data quickly and adapt changes through retraining.

Table II
COMPARISON OF UNSUPERVISED CLASSIFIERS

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
One-class SVM [15]	0.72	1.0	0.72	0.84
Isolation Forest [15]	0.91	1.0	0.91	0.95
Local Outlier Factor [15]	0.99	1.0	0.99	0.99

To evaluate the efficacy of the proposed predictive approach — which predicts when to retrain and update the model based on changes in user traffic — compared with the other two approaches. The trained LSTM Model is used for prediction without changing the model during the experimentation duration. The Retrained-LSTM (threshold approach), where the model is retrained periodically whenever the predefined threshold exceeds. The proposed predictive approach, named Retrained-LSTM (predictive approach), predicts when to retrain the model immediately after observing changes in user traffic by utilizing an unsupervised classifier over incoming user traffic.

The performance of the considered approaches is evaluated by predicting throughput over a period of time with varying user traffic rates. For the evaluation purpose, the UEs are configured with different data rates than the trained data rates. The values of T_{ds} and T_{e2e} are set to 15 ms and 10 ms in the proposed predictive approach after considering propagation, transmission, processing delays, and other factors [5]. The root mean square error (RMSE) over each data chunk served as the performance metric for the threshold approach, and it is set to 15. Note that the values used for T_{ds} and T_{e2e} are specific to the considered use case in this paper and may vary depending on the operator's requirements and the specific application.

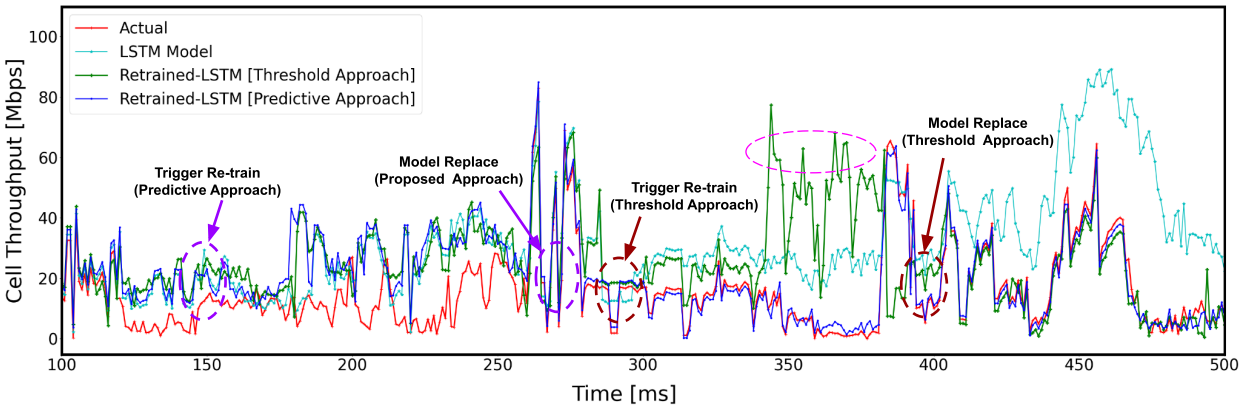

Figure 4. Evaluation of the proposed predictive approach using QP xApp

Figure 4 depicts the performance of the considered approaches in predicting throughput, and multiple UEs with different data rates are configured to create changes in user traffic, which can be observed from 120 ms onwards. The proposed predictive approach is triggered for retraining at 150 ms after detecting changes in user traffic over two consecutive data chunks ($[120-150]$ ms) and is replaced with retrained LSTM model at 260 ms. For the same user traffic, the threshold approach is able to detect changes in user traffic over the interval $[270-285]$ ms and trigger retraining at 285 ms when its RMSE exceeded the defined threshold. The retrained model of the threshold approach is replaced at 405 ms. A performance comparison reveals that the changes in user traffic are effectively adapted by the proposed predictive approach and reduce the violations compared to the threshold approach. Note that preparation of an updated model before the arrival of new data may cause minimal SLA violations and resource provisioning issues while avoiding unnecessary retraining to save significant resources.

IV. Conclusions and Future Directions

This paper presented an approach to predict when to retrain an AI/ML model which can prevent severe SLA violations and facilitate efficient resource provisioning. We realized the ZSM framework through O-RAN architecture components. We evaluated the proposed predictive approach with the QoS prediction use case by using the Open RAN Software Community (OSC) platform and compared it with the existing threshold approach. The future work will focus on exploring Reinforcement Learning (RL) approaches to predict when to retrain an AI/ML model.

Acknowledgment

This study is partly supported through a DST SERB Startup Research Grant (SRG-2021-001522). This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART") and by the KDT-JU project Collaborative edge-cLoud continuum and Embedded AI for a Visionary industry of thE futuRe (CLEVER) (grant agreement no. 101097560). KDT-JU receives funding from the Horizon Europe Research Framework and the National Authorities.

References

[1] M. Liyanage, Q.-V. Pham, K. Dev, S. Bhattacharya, P. K. R. Maddikunta, T. R. Gadekallu, and G. Yenduri, "A survey on Zero touch network and Service (ZSM) Management for 5G and beyond networks," *Journal of Network and Computer Applications*, p. 103362, 2022.

[2] V. R. Chintapalli, K. Kondepu, A. Sgambelluri *et al.*, "Orchestrating edge-and cloud-based predictive analytics services," in *2020 European Conference on Networks and Communications (EuCNC)*. IEEE, 2020, pp. 214–218.

[3] M. Polese, L. Bonati, S. D'Oro *et al.*, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, 2023.

[4] V. Gudepu, V. R. Chintapalli, L. Valcarenghi, and K. Kondepu, "Exploiting drift detection techniques for next generation radio access networks," in *2023 15th International Conference on COMmunication Systems NETworkS (COMSNETS)*, 2023, pp. 489–491.

[5] TM forum, "5G for Vertical Industries," Aug, 2020.

[6] A. H. A. Muktadir and V. P. Kafle, "Prediction and dynamic adjustment of resources for latency-sensitive virtual network functions," in *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2020, pp. 235–242.

[7] K. Samdanis, A. N. Abbou, J. Song, and T. Taleb, "AI/ML Service Enablers Model Maintenance for Beyond 5G Networks," *IEEE Network*, pp. 1–10, 2023.

[8] O-RAN software community (OSC). [Online]. Available: https://wiki.o-ran-sc.org/display/ORAN/O-RAN+Software+Community

[9] S.-C. Lin, C.-H. Lin, and W.-C. Chen, "Zero-Touch Network on Industrial IoT: An End-to-End Machine Learning Approach," *arXiv preprint arXiv:2204.12605*, 2022.

[10] T. chung Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.

[11] G. Barlacchi, M. De Nadai, R. Larcher *et al.*, "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Scientific data*, vol. 2, no. 1, pp. 1–15, 2015.

[12] D. Raca, D. Leahy, C. J. Sreenan *et al.*, "Beyond throughput, the next generation: a 5G dataset with channel and context metrics," in *Proceedings of the 11th ACM multimedia systems conference*, 2020, pp. 303–308.

[13] Qualcomm, "Setting off the 5G Advanced evolution," 2022.

[14] S. Zhao, M. Talasila, G. Jacobson *et al.*, "Packaging and sharing machine learning models via the acumos ai open platform," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 841–846.

[15] Scikit learn, "Comparing anomaly detection algorithms for outlier detection," 2023.

[16] J. Ahmed, A. Johnsson, R. Yanggratoke *et al.*, "Predicting SLA Violations in Real Time using Online Machine Learning," *arXiv preprint arXiv:1509.01386*, 2015.