Generative Adversarial Networks-Based AI/ML Model Adaptive Retraining for Beyond 5G Networks

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

***Abstract*—Beyond fifth-generation (B5G) networks aim to support high data rates, low-latency applications, and massive machine communications. Artificial Intelligence (AI) and Machine Learning (ML) can help to improve B5G network performance and efficiency. However, dynamic service demands of B5G cause AI/ML performance degradation, resulting in violations of Service Level Agreements (SLA), over- or under-provisioning of resources, etc. Retraining is essential to address the performance degradation of the AI/ML models. 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 the generative adversarial networks (GANs) architecture. The proposed predictive approach is evaluated for a Quality of Service (QoS) prediction use case on O-RAN Software Community (OSC) platform and compared to the predictive approach based on the classifier and the threshold approach. The results show that the proposed predictive approach outperforms both the classifier-based predictive and threshold approaches.**

***Index Terms*—AI/ML Model, Beyond fifth-generation (B5G) Networks, Retraining, Generative Adversarial Networks (GANs).**

I. Introduction

Beyond fifth generation (B5G) networks are driving significant transformations in the networking industry by addressing diverse demands such as ultra-reliable low latency communications (uRLLC), massive machine-type communications (mMTC), and enhanced mobile broadband (eMBB). However, network operators are increasingly recognizing the need for a higher level of intelligence to effectively navigate the inherent complexity of B5G networks and meet the growing service demands. Artificial Intelligence/Machine Learning (AI/ML) models offer a promising solution as they can effectively handle complex network architectures and make intelligent decisions (i.e., resource allocation based on the predicted user traffic) [1], which makes them suitable for B5G networks.

Several ongoing projects, such as Operator Defined Open and Intelligent Radio Access Networks (O-RAN), Telecom Infra Project (TIP), and Open RAN Policy Coalition, are actively working towards enabling the *intelligence* in B5G networks. Among these, the O-RAN Alliance architecture enables *intelligence* through two logical RAN Intelligent Controllers (RICs): Non-real time RIC (Non-RT RIC) and Near-real time RIC (Near-RT RIC) [2]. The Non-RT RIC operates use cases with a granularity of at least $1\ s$, whereas the Near-RT RIC operates use cases on a timescale between $10\ ms$ and $1\ s$.

Utilizing the AI/ML models enhances the performance of B5G networks along with some challenges to deal with [3]. Traditionally, the performance of an AI/ML model depends on the data sets being used for training or observation. However, the highly dynamic service demands of users in B5G networks (i.e., changes in user traffic) can affect the AI/ML model performance degradation, resulting in resource allocation and utilization issues [4]. Over-provisioning of resources (e.g., bandwidth, computing) results in network congestion, and increased operational expenditure (OPEX). On the other hand, under-provisioning of resources results in reduced network performance, poor QoS, longer response time, and network outages.

The above factors profoundly influence the Service Level Agreements (SLAs) between service providers and users. For example, in an automated guided vehicle (AGV) use case exploiting B5G, the SLA defines a constant throughput of $80\ Mbps$, with a response time of less than $100\ ms$. However, when an AI/ML model performance degradation occurs, it can lead to violations of pre-defined SLAs [5]. Therefore, it is crucial to address the model performance degradation in B5G.

Retraining (or updating) the model with newly arrived user data is an effective strategy to ensure the AI/ML model performance. In [6], a threshold approach is proposed to trigger model retraining. This approach continuously monitors key performance metrics such as *accuracy*, *precision*, *recall*, or the *F1 score* and initiates retraining when these metrics 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 the model performance and violate SLAs. Another approach in [7] uses the periodic retraining, where the models are updated at regular intervals regardless of fluctuations in the user traffic or the performance of the deployed AI/ML model.

Our recent work in [8], presented a predictive approach, in which the AI/ML model retraining is predicted by determining the changes in user traffic through an unsupervised classifier (i.e., local outlier factor (LOF)) over the incoming user data.

This approach minimizes SLA violations and optimizes the resource utilization. However, it has certain limitations such as: (i) fails to determine changes in user traffic whenever the change is not continuous; (ii) may not be suitable for the incoming user traffic with high variance and can lead to frequent retraining; and (iii) the method used to determine the number of consecutive windows (or data chunks) to trigger model retraining might vary depending on the specific application being considered.

In order to overcome the limitations of statistical measures in the threshold approach [6] and unsupervised classifiers in the predictive approach [8] for retraining the AI/ML model, we propose a novel predictive approach that leverages an advanced AI/ML framework called Generative Adversarial Networks (GANs) due to their ability to capture the underlying distribution of the user traffic and adaptability to the dynamic environments [9]. The proposed approach reduces SLA violations and increases the efficiency of computational resources.

The main contributions of this paper are summarized as follows:

- A novel approach to predict when to retrain an AI/ML model using GANs.
- Evaluating the proposed approach by considering the Quality of Service (QoS) prediction use case over the O-RAN software community (OSC) platform [10].
- Performance of the proposed approach is compared with state-of-the-art approaches such as the predictive approach [LOF] and a threshold approach.

II. Background

Generative adversarial networks (GANs) [9] are advanced AI/ML framework designed to generate new user data that closely resembles the observed (or train) data. As shown in Figure 1, GANs employ two neural networks: the generator and the discriminator, where the generator network generates synthetic data, also known as "fake data or generated data," to mimic the distribution of the observed data. On the other hand, the discriminator network is responsible for distinguishing between the observed and the generated data.

During training, the generator initially produces fake data, and the discriminator quickly learns to differentiate between observed and generated data. However, as the training progresses, the generator improves its output to generate more realistic data samples that can deceive the discriminator. Simultaneously, the discriminator enhances its ability to distinguish between the observed and the generated data samples. This adversarial interplay between the generator and the discriminator leads to the refinement of both networks.

The generator network attempts to model a noise vector z to fit the input user data traffic used for training, whereas the discriminator attempts to accurately classify the generated data from the observed data. Loss convergence of the generator and discriminator terminates the training period. Essentially, both the generator and discriminator networks are jointly involved in a 2 – player min-max game, as shown in Equation 1, until

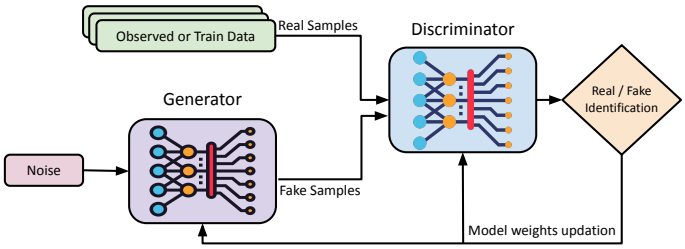

Figure 1: GAN architecture.

the discriminator fails to distinguish between the observed and generated data [11].

$$\min_G \max_D \; \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1-\log D(G(z))] \quad (1)$$

where x is the user input data, p_{data} is the observed data distribution, $p_z(z)$ is the noise distribution, $log(D(x))$ is the predicted output of the discriminator for x, and $log(D(G(z))$ is the output of the discriminator on the GAN generated data $G(z)$. The aim is to maximize the ability of the discriminator to identify observed data from generated data (i.e., $\max_D$), whereas the generator part of the equation tries to minimize the discriminator's ability to classify observed and generated data correctly (i.e., $\min_G$).

The applications of GANs in the radio access network (RAN) include the generation of synthetic data, resource optimization, network planning, and QoS management [11]. However, we leverage the GANs to determine when to retrain an AI/ML model, which is detailed in the following sections.

III. System Model and Proposed Approach

This section describes the system model and the proposed approach to predict when to retrain an AI/ML model.

A. System Model

Figure 2 shows the O-RAN architecture in which two RAN intelligent controllers (RIC) are defined: the Near-Real Time (Near-RT) and the Non-Real Time (Non-RT) [2] to employ the *intelligence*. These RICs enable B5G network autonomous optimization by operating at different timescales depending on the position of the AI/ML model inference [12].

The O-RAN architecture provides various interfaces, including O1, A1, and E2, to enable data collection and communication among the RAN components (i.e., central unit (O-CU), distributed unit (O-DU), and radio unit (O-RU)). The *O1-interface* obtains the input data from all the components (O-CU, O-DU, and O-RU) and model deployment/termination information from Non-RT to Near-RT RIC. The *A1-interface* publishes policy-based guidance, AI/ML performance feedback, verification, and monitoring information between Non-RT RIC and Near-RT RIC. The *E2-interface* is to control the RAN functions through the E2 control messages.

Within the Non-RT RIC, the AI/ML model management block plays a crucial role in determining when to retrain the model using the proposed predictive approach, which leverages the GANs to predict retraining requirement in advance. The

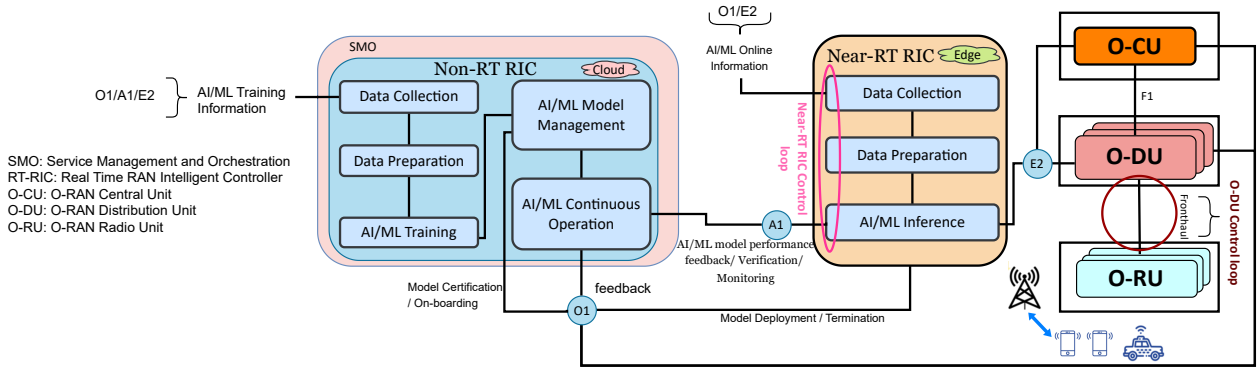

Figure 2: O-RAN architecture.

AI/ML model management block also communicates with the AI/ML training block to initiate the retraining when necessary.

B. Proposed Approach

The proposed approach predicts when to retrain the AI/ML model by exploiting GANs to generate data that is close to the observed and then performs Kolmogorov-Smirnov test (KS Test) [13] over both the generated and incoming user traffic data to determine changes in the user traffic. The proposed approach triggers retraining if there is any change in user traffic. Algorithm 1 depicts the proposed predictive approach, and Table I reports the definitions of the parameters/variables used in it.

Table I: Description of variables used in Algorithm.

Acronym	Referring to / Definition
ds	Incoming data stream.
$\mathcal{D}_{\text{MLP}}$	Discriminator built with multi-layer perceptron neural network to differentiate between the observed and generated data.
$\mathcal{G}$	Generator built with the LSTM architecture to generate data close to the observed data.
P_{kstest}	Value determined by Kolmogorov-Smirnov Test (KS Test) [13] between the generated and observed data during the training of GAN.
$\mathcal{D}_{score}$	Range of discriminator score (i.e., between 0 and 1) obtained for the observed data during training of GAN.
WS	Window Size
$Noise$	A source of variability or randomness injected into the GAN model to generate diverse and realistic data [9]. We used Gaussian noise throughout the study.
$Y_{predict}$	Stores the $\mathcal{D}_{\text{MLP}}$ prediction output in a list.
Z	Samples from the distribution considered in $Noise$.
g_{data}	Generated (or new) data from Generator $\mathcal{G}$.
$\mathcal{G}(z)$	Generator $\mathcal{G}$, taking a vector z of size WS as an input, which is sampled from Z.
KST	KS Test, which quantifies the distance between two distributions based on their empirical cumulative distribution functions (ECDF) and determines whether the incoming user traffic is following the observed data distribution or not [13].
P_{value}	Value determined by KS Test for each window of length WS by comparing both generated and incoming user data.

Algorithm 1 takes ds, $\mathcal{D}_{MLP}$, $\mathcal{G}$, P_{kstest}, $\mathcal{D}_{score}$, WS, and a $Noise$ as inputs and predicts whether to retrain an AI/ML model or not as an output. The incoming data stream ds divides into consecutive chunks of data with a length of WS, and the $\mathcal{D}_{MLP}$ predicts whether the incoming user traffic belongs to the observed or not over each data sample in the data chunk (see lines 4-6). The $\mathcal{D}_{MLP}$ assigns a score between 0 and 1 for each data sample, and if the score lies in the range of $\mathcal{D}_{score}$, no action will be taken (see lines 7-8). Otherwise, a warning zone is triggered to indicate that changes in user data are introduced in the incoming user traffic (see lines 9-10). We collect data for model retraining from the point where the warning zone starts and till the model retraining is triggered.

The generator ($\mathcal{G}$) takes the $Noise$ as input and generates data (i.e., g_{data}) of length WS (see lines 12-13). The g_{data} is compared with the data chunks from the incoming user traffic using the KS Test to determine changes in the user traffic. The KS Test computes the maximum distance between the cumulative distribution functions (CDFs) of the two distributions (i.e., g_{data} and ds) and calculates a P_{value}, which represents the probability that the ds and the g_{data} are from the

Algorithm 1 Predicting AI/ML Model Retraining Using GANs.

1: $Input: ds, \mathcal{D}_{MLP}, \mathcal{G}, P_{kstest}, \mathcal{D}_{score}, WS, Noise$
2: $Output$: Retrain the AI/ML model or not
3: **while** data_available **do**
4: **for** $i \leftarrow 0$ to $\lfloor \frac{length(ds)}{WS} \rfloor$ **do**
5: $Y_{predict} \leftarrow \mathcal{D}_{MLP}([i * WS \ to \ (i+1) * WS])$
6: **for** $j \leftarrow 0 \ to \ WS$ **do**
7: **if** $Y_{predict}[j]$ *is in* $\mathcal{D}_{score}$ **then**
8: $Do\_Nothing$
9: **else**
10: $Warning \ Zone$
11: **end if**
12: $Z \leftarrow Noise$
13: $g_{data} \leftarrow \mathcal{G}(z)$
14: $P_{value} \leftarrow KST[g_{data}, ds[i * WS \ to \ (i+1) * WS]]$
15: **if** $P_{value} < P_{kstest}$ **then**
16: $Retrain \ the \ AI/ML \ model$
17: **end if**
18: **end for**
19: **end for**
20: **end while**

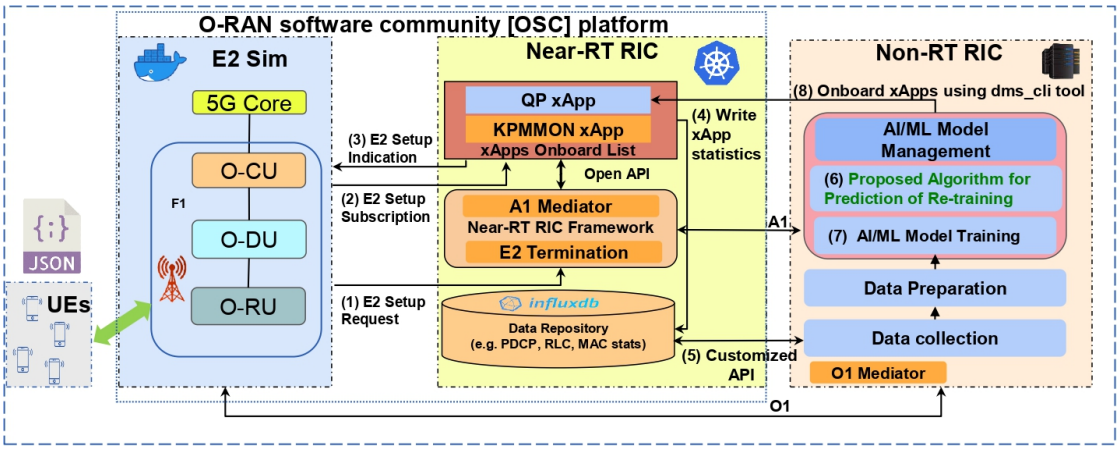

Figure 3: Experimental Setup.

observed data. A lower P_{value} indicates the arrival of significant changes in the incoming user traffic. Whenever P_{value} is lower than P_{kstest}, the proposed approach trigger AI/ML model retraining, indicating that the incoming user traffic is different from the underlying observed data (see lines 14-17). During the retraining, the previously trained AI/ML model weights are updated based on the newly available data and deployed as a xApp (i.e., AI/ML model deployed at the Near-RT RIC) to predict further.

IV. Experimental Setup and Results

This section describes the experimental setup used to evaluate the Quality of Service (QoS) prediction use case over the OSC RIC platform, followed by a discussion of the results.

A. Experimental Setup

The proposed approach is evaluated for *QoS prediction* use case, which focuses on predicting the service quality (i.e., throughput) provided by network operators to their customers [14]. Figure 3 depicts the experimental setup used to evaluate the QoS prediction use case, in which the proposed approach is deployed on the Non-RT RIC platform and integrated with the OSC RIC platform.

We deployed the OSC Near-RT RIC framework (F-release), an E2 simulator, and various open interfaces [10]. The Near-RT RIC framework is deployed as a Kubernetes pod, a collection of containers running inside a node of a Kubernetes cluster. 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). Open interfaces such as A1, E2, and O1 were also utilized, as shown in Figure 3. These components are deployed as microservices within the Kubernetes cluster. The AI/ML models run at the Non-RT and Near-RT RIC are referred to as rApps and xApps, respectively [2]. When a xApp is onboarded into the Near-RT RIC, it would write its statistics into the common database (i.e., InfluxDB).

Within the Near-RT RIC, a key performance metric monitoring (KPMMON) xApp is responsible for monitoring and storing the statistics of each layer in the RAN to an InfluxDB database. It periodically writes the statistics of the data units used for each layer, and an API function inside the database measures the cell throughput each time the data is updated. Furthermore, the QoS prediction (QP) xApp is also onboarded in the Near-RT RIC. The QP xApp predicts the QoS values and writes them into the InfluxDB database, and it utilizes a three-layer Long Short Term Memory (LSTM) model [15] with 100 hidden units in each layer, along with a rectified linear unit (ReLU) activation function.

An OSC E2 simulator consists of 5G core, O-CU, O-DU, and O-RU. These components establish communication with the Near-RT RIC through the E2 control messages such as (1) E2 setup request; (2) E2 setup subscription; and (3) E2 setup indication as shown in Figure 3. Additionally, the E2 simulator can be connected by multiple user equipment (UEs) with varying data rates configured through a customized JavaScript Object Notation (*json*) file.

Note that the current F-release of OSC does not include a complete implementation of the SMO (Service Management and Orchestration) and the Non-RT RIC frameworks. Therefore, to evaluate the QoS prediction use case using the proposed approach, the missing functionalities in the OSC platform are implemented and deployed on 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 using the dms_cli tool. Additionally, the proposed predictive approach is implemented as a REST API. It 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 a GAN architecture to determine changes in the incoming 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 it's deployment.

We leverage the deployed experimental setup to evaluate the performance of the proposed predictive approach [GAN] along with two other approaches, the predictive approach [LOF] [8] and the threshold approach [6] for determining the change in the user traffic and to adapt with it.

Initially, the proposed predictive approach [GAN] is employed with WS set to 10 for evaluating the QoS prediction usecase [5]. Whereas, the functionality of the other approaches that are considered for comparison with the proposed predictive

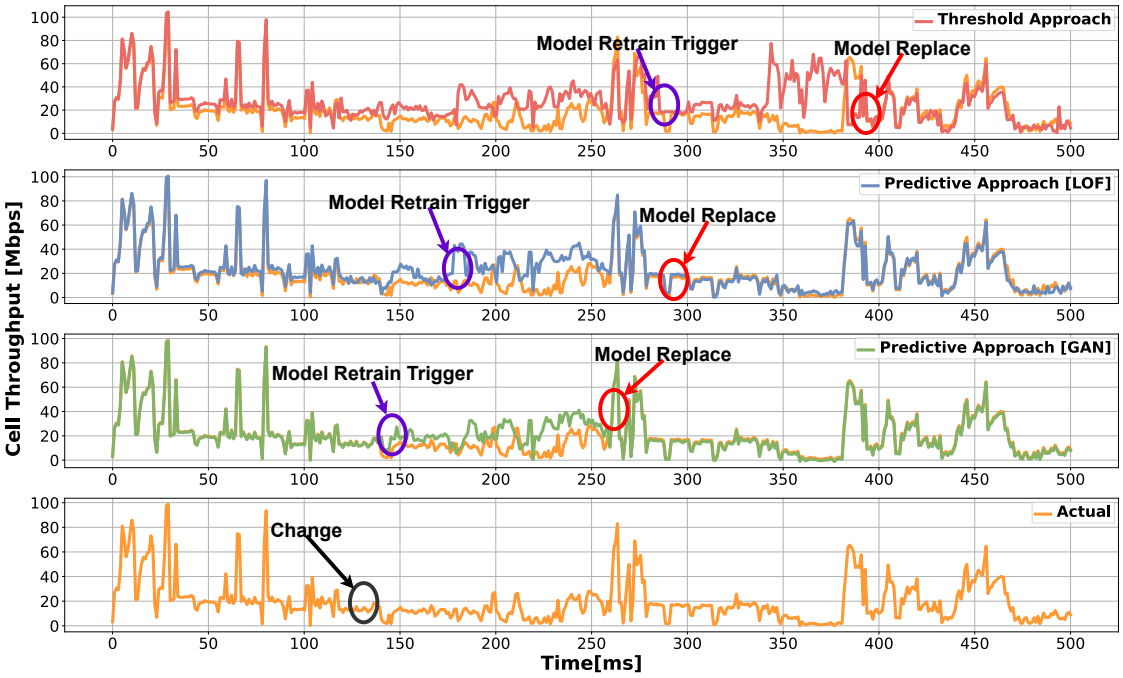

Figure 4: Evaluation of QP xApp using the proposed predictive approach along with the other approaches

approach [GAN] is as follows:

- **Predictive Approach [LOF]:** Predicts when to retrain the AI/ML model by exploiting a LOF classifier that calculates the local density deviation for each incoming user data sample and determines whether it belongs to the observed data or the new data. If the new data arrives over a certain period of time, which can be calculated as a function of both the time taken to transmit the considered data samples from the data stream (i.e., T_{ds}) and the end-to-end delay of the application under consideration (i.e., T_{e2e}), then the predictive approach [LOF] triggers retraining. T_{ds} and T_{e2e} are set to 20 ms and 5 ms, respectively to evaluate this approach [8].
- **Threshold Approach:** Triggers the model retraining whenever the considered AI/ML model performance metric exceeds the pre-defined threshold. The root mean square error (RMSE) over each data chunk of size WS is served as the performance metric for the threshold approach, and it is set to 15 [8].

The performance metric considered to evaluate these approaches is how promptly they can determine the changes in the incoming user traffic and adapt to it by triggering model retraining [16].

B. Results

We considered a QoS dataset collected from the InfluxDB database through an API inside the Non-RT RIC and used it to train the LSTM models for each of the considered approaches. The trained LSTM model of each approach is deployed as QP xApp to predict the throughput as shown in Figure 4.

Figure 4 depicts the performance of the considered approaches in predicting the throughput. We configured multiple UEs with different data rates to create changes in the incoming user traffic (i.e., the bottom plot of Figure 4), which can be observed from 140 ms onwards. The threshold approach detected changes in the user traffic between 270 ms and 280 ms. It triggered retraining at 280 ms when its RMSE exceeds the defined threshold, and the retrained LSTM model is replaced at 395 ms to predict the further throughput values. And the predictive approach [LOF] is employed over the same user traffic and it triggered retraining at 180 ms after detecting changes in user traffic over four consecutive data chunks between 140 ms and 180 ms, and is replaced at 295 ms. Whereas, the proposed predictive approach [GAN] triggers retraining at 150 ms after determining changes in the user traffic and is replaced with retrained LSTM model at 260 ms to predict the throughput values accurately. Figure 5 presents the model retraining trigger time, retraining duration, and its replacement time for each approach considered in Figure 4. From Figure 5, the early trigger for retraining, short duration of model retraining with newly arrived data, and quick model replacement by the proposed predictive approach [GAN] can adapt to the user traffic changes effectively. It is important to note that early detection of changes in the user traffic helps to retrain and replace the model quickly to predict the QoS accurately.

Furthermore, to evaluate the performance of the three approaches, we have analyzed the traffic rate over a duration of 3600 $seconds$, in which the variations in user traffic are introduced at 60 specific instances. Figure 6 presents a sampled probability mass function (PMF) that depicts the correlation between the time required to identify the occurrence of a change in user traffic and the corresponding frequency of such occurrences. On the Y-axis of Figure 6, occurrences represent the percentage of instances out of 60 in which each approach takes a specific time (e.g., 10 ms, 20 ms, etc.) to trigger retraining after the occurrence of an actual traffic change, whereas the X-axis is a discrete variable as the model retrain trigger time will be a factor of WS. The threshold approach exhibits a maximum

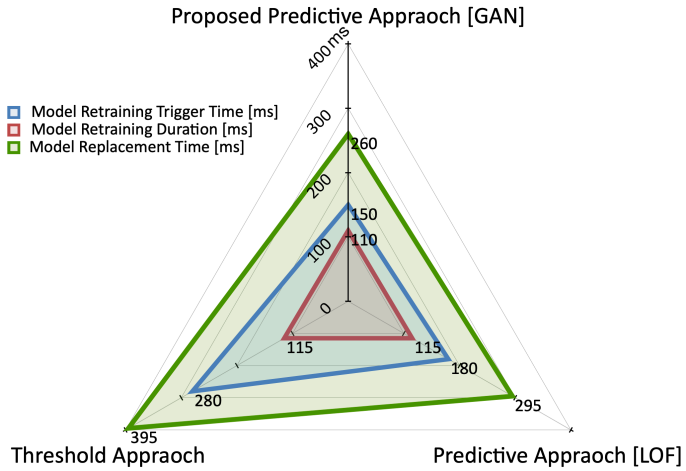

Figure 5: Performance comparison of the proposed predictive approach with the other approaches.

of 25% occurrences at $60\ ms$, while the predictive approach [LOF] shows approximately 85% occurrences at $40\ ms$. In contrast, the proposed predictive approach [GAN] achieves approximately 98% occurrences at $10\ ms$. Thus, experimental evidence supports the effective determination of user traffic changes by the proposed predictive approach [GAN].

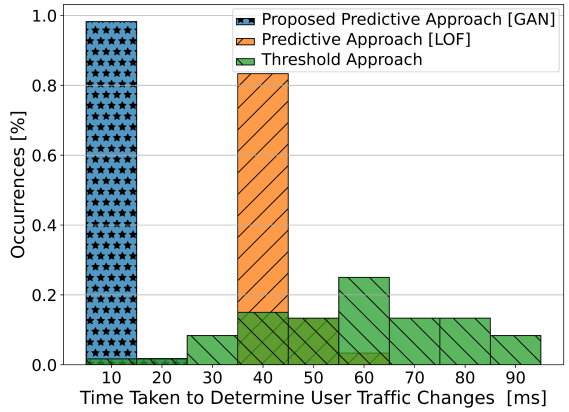

Figure 6: Sampled PMF of model retraining trigger time.

A performance comparison reveals that the changes in user traffic are effectively adapted by the proposed predictive approach [GAN] compared to the predictive approach [LOF] and the threshold approach due to its ability to capture the underlying user data distribution. Preparing 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. Although, training the GANs can be challenging due to certain challenges, such as model instability, where the generator fails to capture the full knowledge of the observed data. Various techniques, such as architectural modifications, hyperparameter adjustments, and regularization methods can be used to address this challenge.

V. 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. The proposed predictive approach leverages the GANs to predict when to retrain an AI/ML model. We compared the proposed predictive approach [GAN] with the QoS prediction use case using the Open RAN Software Community (OSC) platform with the existing predictive approach [LOF] and a threshold approach. The future work will explore Reinforcement Learning (RL) approaches to predict when to retrain an AI/ML model.

Acknowledgment

This study is partly supported by a DST SERB Startup Research Grant (SRG-2021-001522). The European Union partially supported this work 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] I. A. Bartsiokas, P. K. Gkonis, D. I. Kaklamani, and I. S. Venieris, "Ml-based radio resource management in 5g and beyond networks: A survey," *IEEE Access*, vol. 10, pp. 83 507–83 528, 2022.

[2] O-RAN Working Group 2, "O-RAN AI/ML Workflow Description and Requirements - v1.01, Technical Specification, 2023."

[3] N. Aryal, E. Bertin, and N. Crespi, "Open radio access network challenges for next generation mobile network," in *2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2023, pp. 90–94.

[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] V. Gudepu, V. R. Chintapalli, P. Castoldi, L. Valcarenghi, B. R. Tamma, and K. Kondepu, "Adaptive retraining of ai/ml model for beyond 5g networks: A predictive approach," *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, pp. 282–286, 2023.

[9] H. Navidan, P. F. Moshiri, M. Nabati, R. Shahbazian, S. A. Ghorashi, V. Shah-Mansouri, and D. Windridge, "Generative adversarial networks (gans) in networking: A comprehensive survey & evaluation," *Computer Networks*, vol. 194, p. 108149, 2021.

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

[11] A. Kaloxylos, A. Gavras, D. Camps, M. Ghoraishi, and H. Hrasnica, "Ai and ml–enablers for beyond 5g networks," 2021.

[12] V. R. Chintapalli, V. Gudepu, K. Kondepu, A. Sgambelluri, A. Franklin, B. R. Tamma, P. Castoldi, and L. Valcarenghi, "Wip: Impact of ai/ml model adaptation on ran control loop response time," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2022, pp. 181–184.

[13] G. Marsaglia, W. W. Tsang, and J. Wang, "Evaluating kolmogorov's distribution," *Journal of statistical software*, vol. 8, pp. 1–4, 2003.

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

[15] V. Reddy Chintapalli, K. Kondepu, A. Sgambelluri, A. Franklin A, B. Reddy Tamma, P. Castoldi, and L. Valcarenghi, "Orchestrating edge- and cloud-based predictive analytics services," in *Proc. of EuCNC*, 2020, pp. 214–218.

[16] Y. Wu, L. Liu, Y. Yu, G. Chen, and J. Hu, "Aewae: An efficient ensemble framework for concept drift adaptation in iot data stream," *arXiv preprint arXiv:2305.06638*, 2023.