GAN-Based Drift and Anomaly Detection for Open Radio Access Networks

Venkateswarlu Gudepu*, Bhargav Chirumamilla*, Venkatarami Reddy Chintapalli†, Piero Castoldi‡,
Luca Valcarenghi‡, Bheemarjuna Reddy Tamma§, Deepak Kataria¶, 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
¶ IP Junction, USA
Email: 212011003@iitdh.ac.in, k.kondepu@iitdh.ac.in

***Abstract*—Next-Generation Radio Access Networks (NG-RANs) aim to facilitate high data rates, low-latency applications, and dense mobile connectivity — benefit from the integration of Artificial Intelligence and Machine Learning (AI/ML) to enhance performance and efficiency. Nevertheless, the dynamic service demands within NG-RAN (namely Open RAN) lead to AI/ML performance degradation known as *drift*, resulting in violations of Service Level Agreements (SLA) and issues like over- or under-provisioning of resources. Detecting and adapting to drift becomes crucial to meet the diverse requirements of intelligent networks. Due to frequent retraining, the existing threshold and classifier-based approaches have potential disadvantages such as SLA violations and resource inefficiency. This paper introduces a novel approach that exploits the Generative Adversarial Network (GAN) architecture to determine the drift and anomaly. The proposed approach is evaluated for a throughput prediction use case over a real-time dataset and compared to the threshold and classifier-based approaches. The results show that the proposed approach outperforms the threshold and classifier-based approaches.**

***Index Terms*—AI/ML Model, Open RAN, Generative Adversarial Networks (GANs).**

I. Introduction

Mobile Network Operators (MNOs) are incorporating Artificial Intelligence / Machine Learning (AI/ML) techniques to meet various use cases of Beyond 5G (B5G) networks. AI/ML capabilities are increasingly becoming a pivotal component in mobile networks, driving efficiency and introducing intelligence and automation across various domains of Open Radio Access Networks (Open RANs) — Management Data Analytics (MDA) in orchestration, Network Data Analytics Function (NWDAF) in the 5G core network, and Next Generation RAN (NG-RAN), including RAN intelligence as defined in 3rd Generation Partnership Project Technical Specification (3GPP TS) 38.300 and 3GPP TS 38.401 [1].

The emergence of AI/ML has transformed various aspects of Open RAN, focusing on Network Planning, Network Diagnostics/Insights, and Network Optimization and Control [2]. In Network Planning, AI/ML techniques optimize base station placement and dimensioning for Centralized RAN clusters. Network Diagnostics utilize AI/ML for forecasting network characteristics, user localization, and identifying security incidents. Network Optimization and Control includes various RAN components, including radio access, transport/fronthaul/backhaul, virtualization infrastructure, end-to-end network slicing, security, and application functions.

Alongside, to enhance *intelligence* in Open RAN, consortia like Operator Defined Open and Intelligent Radio Access Networks (O-RAN), Telecom Infra Project (TIP), and Open RAN Policy Coalition are actively working [3]. Here, the O-RAN consortium introduced RAN Intelligence Controllers (RICs) to improve RAN performance. However, one of the main challenges in this network is maintaining the AI/ML model performance consistency. AI/ML model drift is a common issue in these networks due to dynamic changes in incoming user data traffic, which impacts AI/ML applications. Model drift can provide false insights and inaccurate decision-making, potentially leading to severe service interruptions. Moreover, model drift can result in inefficient resource allocation and utilization with over/under-provisioning, leading to network performance degradation and service disruptions. Drift occurrence can also impact the Service Level Agreements (SLAs) between service providers and users [4].

In [5], authors focus on anomaly detection, which identifies the unusual user traffic points (or outliers) that deviate from the normal (or observed) behavior of the network. [6] focuses on employing an Auto-Encoder consisting of both encoder and decoder to determine the anomaly. Anomaly detection with the above approaches often triggers False Positives (FPs) and struggles to adapt to dynamic network conditions. Whereas, determining the drift includes several anomaly points and user traffic changes over time. Drift detection can reduce FPs, prevent model performance degradation, and efficient resource utilization. In [7], drift detection is based on statistical features — *Root Mean Square Error (RMSE)*, *Mean Absolute Error (MAE)*, *fisher score*. However, determining the appropriate threshold values for each of the statistical features poses significant challenges — too low may result in excessive computational costs due to frequent retraining — too high may lead to poor model performance and violate SLA.

To address the limitations of statistical measures to detect both the anomaly and drift, the proposed work employs the Generative Adversarial Networks (GANs) due to their ability to capture the underlying distribution of the user traffic and adaptability to the dynamic network conditions. GANs are advanced AI/ML models designed to generate new user data that closely resembles the observed (or trained) data. The applications of GAN in the RAN include generating synthetic data, resource optimization, network planning, and QoS management [8]. More details on GANs can be found in [9]. The proposed approach determines anomaly and drift to increase the efficiency of computational resources and network performance.

The main contributions of this paper are summarized as follows:

- A novel approach to detect drift using GANs.
- Evaluating the proposed approach by considering the Throughput prediction use case.
- Performance of the proposed approach is compared with state-of-the-art approaches such as the classifier approach [Local Outlier Factor (LOF)] and a threshold approach.

II. System Model and Proposed Approach

This section describes the system model and the proposed approach to detect the drift using GANs.

A. System Model

Figure 1 illustrates the O-RAN architecture, defining two RAN intelligent controllers (RIC)– Near-Real Time (Near-RT) and Non-Real Time (Non-RT), which enables the *intelligence*. These RICs enable autonomous optimization of Open RAN by functioning at different timescales, depending on the AI/ML model inference position [10].

The O-RAN architecture provides multiple interfaces, including O1, A1, and E2, to facilitate 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* collects input data from all components (O-CU, O-DU, and O-RU) and transmits model deployment/termination information from Non-RT to Near-RT RIC. The *A1-interface* facilitates policy-based guidance, AI/ML performance feedback, verification, and monitoring information exchange between Non-RT RIC and Near-RT RIC. The *E2-interface* controls RAN functions via E2 control messages.

The AI/ML model management block within the Non-RT RIC is crucial in detecting model drift utilizing the proposed method, which leverages the GANs. In addition, the AI/ML model management block interfaces with the AI/ML training block to trigger retraining as needed by ensuring model accuracy and effectiveness.

B. Proposed Approach

The proposed approach exploits GANs to determine the drift. GANs generate data similar to observed data and then employ the Kolmogorov-Smirnov (KS) test to compare the generated and Long-Short-Term Memory (LSTM) forecasted data to determine the drift occurrence. If changes in user traffic are introduced, then the proposed approach triggers the drift. Algorithm 1 outlines the proposed approach and Table I shows the list of variables used in it.

Table I: Description of variables used in Algorithm.

Acronym	Referring to / Definition
ds	Incoming data stream
$\mathcal{LSTM}$	LSTM to forecast incoming user traffic
$\mathcal{D}$	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) [11] 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 Gaussian Noise is used throughout this study
$Lead\_Time$	The number of previous incoming user data samples to use as input variable for $\mathcal{LSTM}$ to predict the next sample
$LSTM_{forecast}$	Stores the $\mathcal{LSTM}$ forecast data in a list
$Y_{predict}$	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 — 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
P_{value}	Value determined by KS Test for each window of length WS by comparing both generated and $LSTM_{forecast}$

Algorithm 1 takes ds, $\mathcal{LSTM}$, $\mathcal{D}$, $\mathcal{G}$, P_{kstest}, $\mathcal{D}_{score}$, WS, $Lead\_Time$ and a $Noise$ as inputs and triggers the drift. The incoming data stream ds divides into consecutive chunks of data with a length of WS. The $\mathcal{LSTM}$ takes the chunks of ds and forecasts for a given $Lead\_Time$ (i.e., $LSTM_{forecast}$). The discriminator (i.e., $\mathcal{D}$) determines whether the $LSTM_{forecast}$ belongs to the observed data or not (see lines 4-6).

The $\mathcal{D}$ evaluates each data sample from the $LSTM_{forecast}$, assigning a score between 0 and 1 (i.e., $Y_{predict}[j]$). If $Y_{predict}[j]$ falls within the pre-defined range of $\mathcal{D}_{score}$, no action is taken (see lines 7-9). However, if $Y_{predict}[j]$ exceeds $\mathcal{D}_{score}$, the proposed approach triggers an alert zone (i.e., anomaly), signaling that changes in user data have been detected in the incoming traffic (see lines 10-11). Data collection for model retraining begins from the onset of the alert zone until the actual initiation of model retraining.

The generator (i.e., $\mathcal{G}$) utilizes $Noise$ as input to generate data g_{data} of a specified length WS (see line 14-15). The

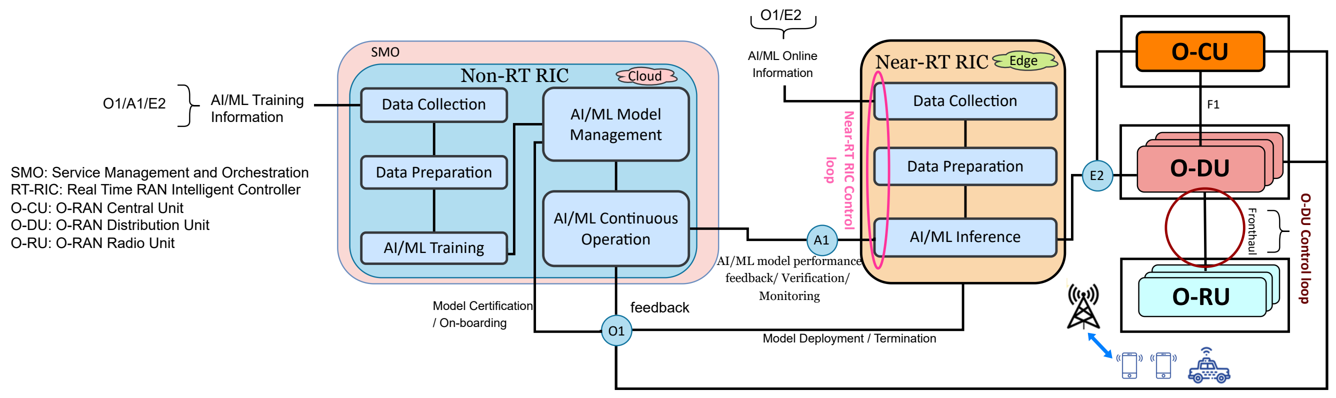

Figure 1: O-RAN architecture.

Algorithm 1 Drift and Anomaly Detection using GANs.

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

KS Test is performed over both g_{data} and $LSTM_{forecast}$ to determine drift occurrence. Here, the KS Test determines the maximum difference between the cumulative distribution functions (CDFs) of g_{data} and $LSTM_{forecast}$ and yields a P_{value}, which signifies the likelihood that $LSTM_{forecast}$ and g_{data} belong to the same distribution as the observed data. A lower P_{value} indicates significant deviations between both $LSTM_{forecast}$ and g_{data}. If the P_{value} lies within P_{kstest}, then the proposed approach triggers drift which indicates that the incoming user traffic differs from the observed data distribution (see 16-19). The previously deployed AI/ML model weights are updated during the retraining and deployed as an xApp (i.e., AI/ML model deployed at the Near-RT RIC) to predict further.

Also, Figure 2 depicts the proposed approach to determine the drift. Initially, forecasts the incoming user traffic with an LSTM network. On the other hand, the $\mathcal{G}$ produces the data closer to the observed. A KS Test is performed over both LSTM predicted traffic and data generated from $\mathcal{G}$, to determine the drift occurrence.

III. Experimental Setup and Results

We evaluate the proposed approach for throughput prediction using a real-time dataset.

A. Throughput Prediction Use Case

The increasing user traffic and evolving service requirements in B5G networks are expected to influence traffic patterns and bandwidth needs [12]. MNOs must manage this complex architecture to meet user expectations. Utilizing AI/ML to predict network conditions enables MNOs to implement user-centric strategies, facilitating network service management and adaptable policy configurations. However, AI/ML implementation introduces challenges, such as drift caused by dynamic user traffic demands, which can result in resource allocation issues and SLA violations. To address these challenges, the proposed approach detects drift and adapts to changes in incoming user traffic.

The proposed approach utilized a real-time dataset from the Colosseum testbed [13] to evaluate the proposed approach and state-of-the-art. The Colosseum dataset comprises a mixed-slice, containing three slices: enhanced Mobile Broadband (eMBB), Machine-type Communications (MTC), and ultra-reliable and low-latency communication (URLLC). Among the three slices, multiple user equipment (UEs) are distributed randomly across static and moving conditions (with a speed of 3m/s). The eMBB slice user traffic under static and slow conditions are considered to evaluate the proposed approach along with other approaches. The AI/ML model is trained with the eMBB slice user traffic (static) and employed to forecast for the given $Lead\_Time$. To introduce the drift scenario, an eMBB slice user traffic with moving conditions is used. The proposed approach and other approaches are employed to determine drift and retrain the model to adapt to user traffic changes.

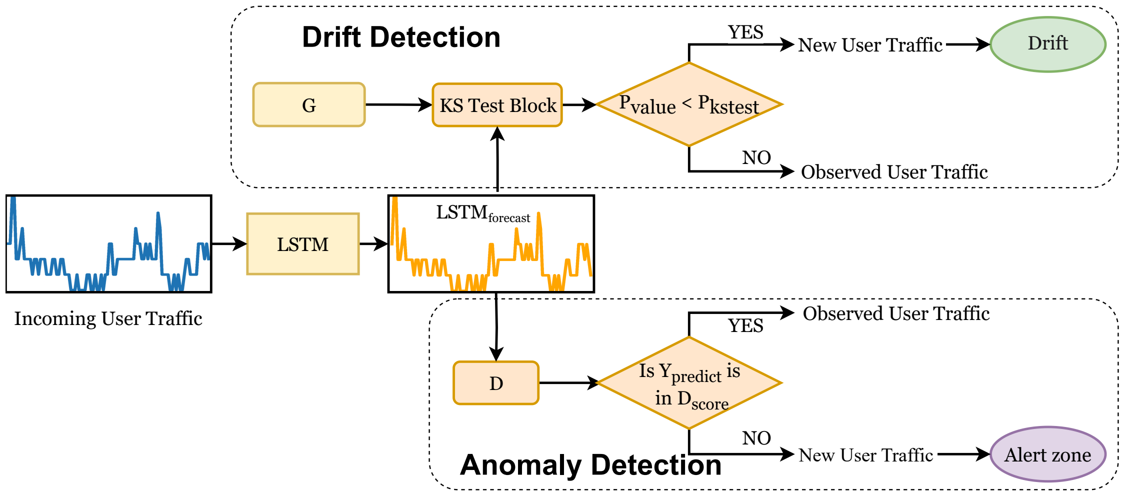

Figure 2: Proposed approach architecture.

Table II: GAN and LSTM Model Description

LSTM		Architecture	3 LSTM layers with 100 hidden units
		Activation function	Rectified Linear Unit (ReLU)
		Loss Function	RMSE
		Learning Rate	0.001
GAN	Generator	Architecture	LSTM layer with 10 hidden units + 3 Fully Connected (FC) layers
		Activation function	ReLU
		Loss Function	Binary Cross-Entropy
		Learning Rate	0.001
	Discriminator	Architecture	3 FC layers
		Activation function	ReLU
		Loss Function	Binary Cross-Entropy
		Learning Rate	0.001

B. Performance Comparison

The performance of the proposed approach is evaluated by comparing with the other two approaches. The working of the approaches considered for comparison are as follows:

- **Classifier Approach [LOF] [14]:** Uses an unsupervised classifier to determine the drift. LOF is one of the unsupervised classifiers that calculates the local density deviation for each incoming user data sample, distinguishing between observed and new data. Suppose new user data arrive over a certain number of windows of WS size, 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}). Classifier-based approach triggers drift whenever newly arrived user traffic is observed for $\frac{T_{ds}}{T_{e2e}}$ number of windows.
- **Threshold Approach [10]:** Triggers the drift occurrence whenever the considered AI/ML model performance metric exceeds the pre-defined threshold. The RMSE over each data chunk of size WS serves as the performance metric for the threshold approach.

The performance metric considered to evaluate these approaches is how promptly they can determine drift occurrence and adapt to the user traffic changes by triggering model retraining [15]. *TP Ratio, FP Ratio, FN Ratio, Accuracy, Precision,* and *F1-score* are also considered for performance evaluation of the considered approaches [4].

C. Results

We considered a real-time dataset from the Colosseum testbed to train LSTM models for each approach under consideration. The architectural specifications for LSTM and GAN are provided in Table II. The parameters for LSTM and GAN architectures are carefully selected for dynamic network conditions and effective drift adaptation. With three LSTM layers of 100 hidden units, the model captures complex data dependencies crucial for drift detection. ReLU activation accelerates training and captures nonlinear relationships. RMSE serves as a vital metric for accurate drift detection. A moderate learning rate of 0.001 balances convergence and stability during training. The GAN integrates LSTM and fully connected layers to generate realistic data samples. ReLU activation fosters variation, while binary cross-entropy loss aligns generated data with trained data. A learning rate of 0.001 ensures stable convergence. These parameter choices are rooted in both application specific considerations and established practices in the Open RAN environment [2].

Also, the input parameters for all the approaches considered are outlined in Table III to assess throughput prediction. The P_{kstest} value was determined during the training phase. Other parameters like T_{ds} and T_{e2e} are specific to use case [16], while

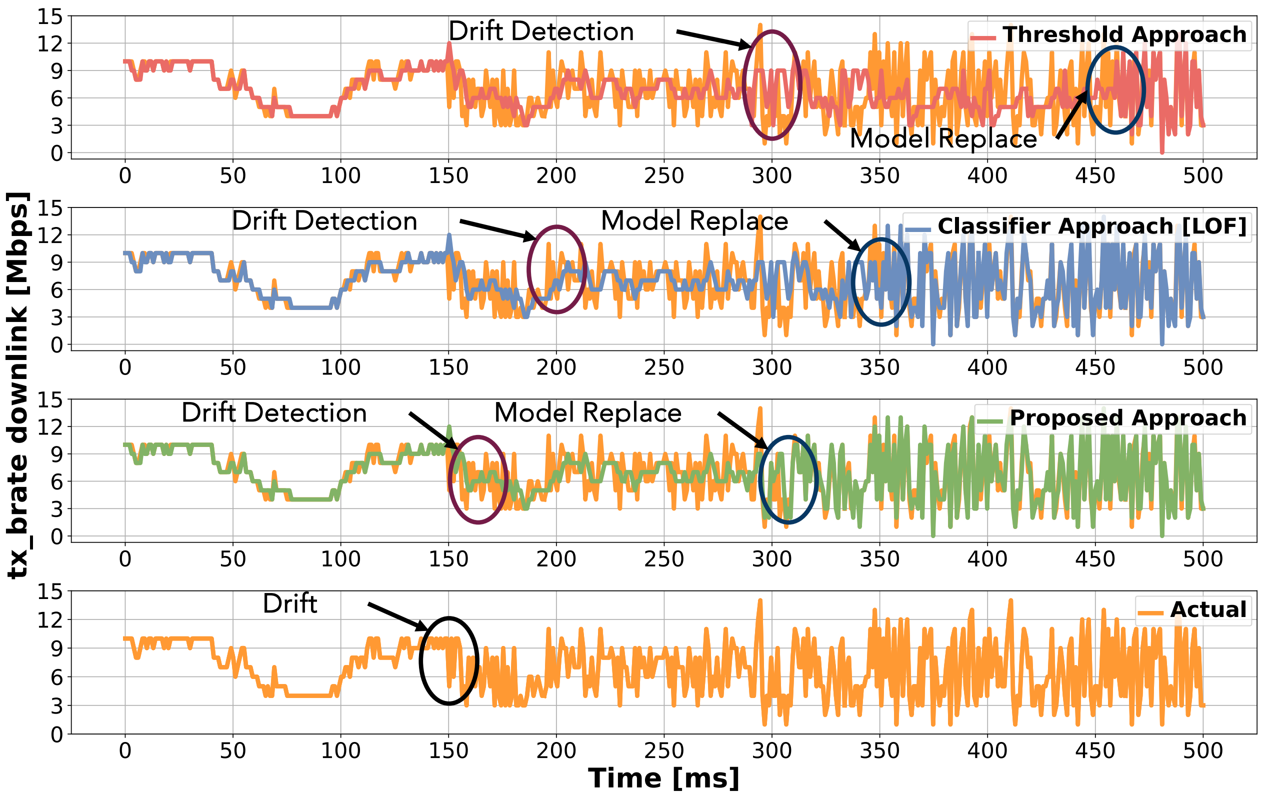

Figure 3: Evaluation of the considered approaches over a real-time user traffic dataset.

Table III: Input Parameters List

Approach	Parameter	Value
Threshold Approach	RMSE	5
Classifier Approach [LOF]	T_{ds} T_{e2e}	20 ms 5 ms
Proposed Approach	WS P_{kstest}	10 0.0016

WS can be changed according to operator requirements. Here, the larger WS value provides more stability by handling short-term drift in user traffic, whereas a smaller WS value increases sensitivity to immediate drift in user traffic, allowing for a prompt response but potentially leading to frequent retraining.

Figure 3 depicts the performance of the considered approaches in predicting downlink *tx_brate*. Initially, from 0 ms to 150 ms UEs are static, and from 150 ms onwards UEs are moving under slow conditions (i.e., moving with 3 m/s). The threshold approach detected changes in user traffic between 290 ms and 300 ms, and determines drift at 300 ms when its RMSE exceeded the defined threshold. The replacement of the retrained LSTM is completed at 460 ms. In the case of the Classifier approach [LOF], it detects drift at 200 ms after identifying changes in user traffic over four consecutive data chunks between 160 ms and 200 ms and retrained LSTM model replaced at 350 ms. On the other hand, the proposed approach detects drift at 160 ms after determining changes in user traffic. The retrained LSTM model was replaced at 305 ms.

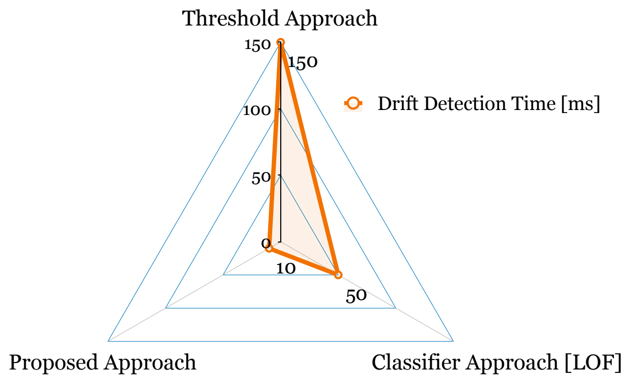

Figure 4: Performance comparison of the proposed predictive approach with the other approaches.

Figure 4 shows the *drift detection time* of considered approaches. The proposed approach exhibits a shorter drift detection time than other methods. This can be due to the following factors: (i) GANs have a high capability to adjust to evolving network conditions, changing traffic patterns, and dynamic user behavior; (ii) The Proposed method used the KS Test to assess distribution similarities, allowing for quickly detecting drift in user traffic.

In the threshold approach, the initial stage of forecasting the incoming new user traffic, the RMSE values are lower than the pre-defined threshold. While the RMSE values surpass the pre-defined thresholds, in the later stages of forecast. Thus, when compared with other approaches, the threshold approach exhibits a higher Drift detection time consistently. As previously mentioned, the Classifier approach [LOF] relies primarily on the number of windows with size WS that are determined. As a result, the approach must wait for the number of windows that are taken into consideration before drift detection, which results in a longer Drift detection time than the proposed approach.

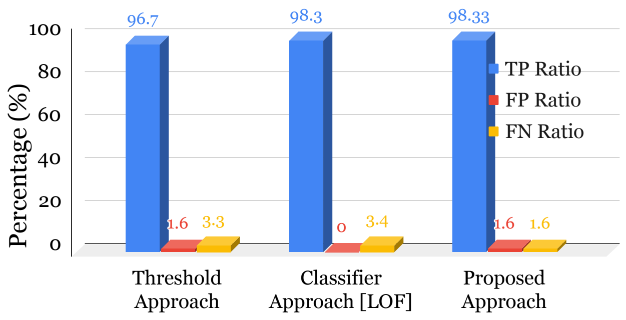

Figure 5: TP Ratio, FP Ratio and FN Ratio of the considered approaches.

Figure 5 depicts the *True Positive (TP), False Positive (FP),* and *False Negative (FN)* ratios of the proposed approach, classifier approach [LOF] and threshold approach. The proposed approach and classifier approach [LOF] have very close TP Ratios of 98.33% and 98.3% respectively, as shown in Figure 5. Whereas, the threshold approach has a slightly lower TP Ratio of 96.7% compared to other approaches. A higher TP ratio suggests the approach effectively detects drift while minimizing false alarms. The FP Ratio of the threshold approach, classifier approach [LOF], and proposed approach are 1.6%, 0%, and 1.6% respectively. The FN Ratio of the threshold approach, classifier approach [LOF], and proposed approach are 3.3%, 3.4%, and 1.6% respectively.

Other performance metrics such as *accuracy*, *precision* and *F1score* of the proposed approach, classifier approach [LOF], and threshold approach are listed in Table IV. As shown in the table IV the threshold approach has low accuracy, precision, and F1score compared to other approaches. Classifier approach [LOF] has a precision of 100% because, during training of LOF, it considers data distribution belongs to one class and during the testing phase it classifies unseen data as new user data. The classifier approach [LOF] and the proposed approach have very close accuracy and F1-score, but the drift detection time is less for the proposed approach than the classifier approach [LOF]. Hence, the proposed approach outperforms the classifier approach [LOF] and Threshold approaches.

Table IV: Drift detection performance comparison of each approach

Approach	**Accuracy (%)**	**Precision (%)**	**F1-score (%)**
Threshold Approach	97	96.7	97.47
Classifier Approach [LOF]	98	100	98.3
Proposed Approach	98	98.33	98.33

IV. Conclusions and Future Directions

This paper presented an approach to determine the drift occurrence that can prevent severe SLA violations and facilitate efficient resource provisioning. The proposed approach leverages the GANs to determine the drift. We compared the proposed approach with the throughput prediction use case with the existing Classifier approach [LOF] and a threshold approach. Future work will explore reinforcement learning (RL) approaches to predicting 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] 3GPP TS 38.401, “NG-RAN; Architecture description,” 2023.

[2] A. Kaloxylos, A. Gavras, D. Camps, M. Ghoraishi, and H. Hrasnica, “AI and ML – Enablers for Beyond 5G Networks,” 2021.

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

[4] V. Gudepu, V. R. Chintapalli, P. Castoldi, L. Valcarenghi, B. R. Tamma, and K. Kondepu, “The Drift Handling Framework for Open Radio Access Networks: An Experimental Evaluation,” *Computer Networks*, vol. 243, p. 110290, 2024.

[5] X. Lin, “Artificial Intelligence in 3GPP 5G-Advanced: A Survey,” *arXiv preprint arXiv:2305.05092*, 2023.

[6] A. Mekrache, K. Boutiba, and A. Ksentini, “Combining Network Data Analytics Function and Machine Learning for Abnormal Traffic Detection in Beyond 5G.”

[7] E. R. Faria, I. J. Gonçalves, A. C. de Carvalho, and J. Gama, “Novelty Detection in Data Streams,” *Artificial Intelligence Review*, vol. 45, pp. 235–269, 2016.

[8] V. Gudepu, B. Chirumallia, V. R. Chintapalli, P. Castoldi, L. Valcarenghi, and K. Kondepu, “Generative Adversarial Networks-Based AI/ML Model Adaptive Retraining for Beyond 5G Networks,” in *IEEE 28th European Wireless (EW) conference, 2024*.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Networks,” *Commun. ACM*, vol. 63, no. 11, p. 139–144, oct 2020. [Online]. Available: https://doi.org/10.1145/3422622

[10] 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.

[11] G. Marsaglia, W. W. Tsang, and J. Wang, “Evaluating Kolmogorov’s distribution,” *Journal of statistical software*, vol. 8, pp. 1–4, 2003.

[12] A. Alzahrani, T. H. Aldhyani, S. N. Alsubari, and A. D. Alghamdi, “Network Traffic Forecasting in Network Cybersecurity: Granular Computing Model.” *Security & Communication Networks*, 2022.

[13] L. Bonati, S. D’Oro, M. Polese, S. Basagni, and T. Melodia, “Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks,” *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, 2021.

[14] 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.

[15] 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.

[16] TM forum, “5G for Vertical Industries,” Aug, 2020.