Demo: A Drift-Handling Automation in AI/ML Framework Integrated O-RAN

Venkatesh Gani
IIT Dharwad, India
cs25dp001@iitdh.ac.in

Dhruv
IIT Dharwad, India
cs25dp004@iitdh.ac.in

Yaswanth Kumar L.S.
IIT Hyderabad, India
cs24resch11007@iith.ac.in

Ashit Subudhi
IIT Dharwad, India
ashit.subudhi.21@iitdh.ac.in

Majid K
IIT Dharwad, India
cs24dp006@iitdh.ac.in

Abhishek Bhattacharyya
IIT Dharwad, India
CS21MS001.alum24@iitdh.ac.in

Venkateswarlu Gudepu
IIT Dharwad, India
CS21DP003.alum24@iitdh.ac.in

Bheemarjuna Reddy Tamma
IIT Hyderabad, India
tbr@cse.iith.ac.in

Koteswararao Kondepu
IIT Dharwad, India
k.kondepu@iitdh.ac.in

Abstract

Beyond 5G (B5G) networks increasingly leverage open and disaggregated Radio Access Networks (O-RAN) to enable high-speed, low-latency, and flexible services across diverse use cases. However, the dynamic nature of user traffic and network conditions causes Artificial Intelligence (AI)/ Machine Learning (ML) models in RAN Intelligent Controllers (RICs) to experience performance drift – leading to inefficient radio resource allocation and potential service level agreements (SLA) violations. This work demonstrates a real-time *drift detection*, *analysis*, and *adaptation* by leveraging an AI/ML framework integrated O-RAN. This is achieved with real-time data collection from open interfaces which would be used by intelligent controllers (i.e., Non-RT RIC) for training and inference purposes of AI/ML models. The proposed drift-handling automation monitors network performance metrics to identify *model drift* with minimal data and computation overheads and then adapts to current network conditions by either *retraining* the model or *replacing* it with an available pre-trained model. This approach improves reliability, agility, and automation in multi-vendor B5G deployments by means of drift detection and resolution.

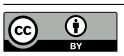
CCS Concepts

• **Networks** → **Network measurement**; **Network experimentation**.

Keywords

Open RAN, Beyond 5G Networks, AI/ML Models, Network Traffic Monitoring, Drift-Handling

ACM Reference Format:
Venkatesh Gani, Dhruv, Yaswanth Kumar L.S., Ashit Subudhi, Majid K, Abhishek Bhattacharyya, Venkateswarlu Gudepu, Bheemarjuna Reddy Tamma, and Koteswararao Kondepu. 2025. Demo: A Drift-Handling Automation in AI/ML Framework Integrated O-RAN. In *2nd ACM Workshop on Open and AI RAN (Open & AI RAN 25), November 4–8, 2025, Hong Kong, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3737900.3770176



OpenRan '25, November 4-8, 2025, Hong Kong, China

ACM ISBN 979-8-4007-1977-6/25/11
https://doi.org/10.1145/3737900.3770176

1 Introduction

In Beyond 5G (B5G) networks, mobile operators aim to support a broader range of services that require higher data rates, ultra-low latency, and greater reliability for billions of connected devices, while simultaneously reducing capital and operational expenditures (CAPEX/OPEX) [7]. Open and disaggregated RAN (O-RAN) plays a central role in this vision by enabling interoperability, programmability, and cost efficiency through open interfaces, virtualization, and AI/ML-driven network automation.

AI/ML models offer a promising approach in O-RAN, as they can effectively handle complex network protocols and make intelligent decisions for network optimization and management [3]. Traditionally, these models are trained on historical or observed data, making their performance highly sensitive to the characteristics of the training dataset. However, in real-world deployments, network traffic evolves continuously due to addition of new devices and roll out of new applications and services. This mismatch between the training data and the current traffic often leads to AI/ML model's performance degradation, a phenomenon commonly referred to as *model or concept drift*.

In [1], model drift is identified when AI/ML model's performance metrics like *accuracy*, *F1-score*, *recall*, and *precision* exceed a predefined threshold. Various drift detection techniques, including Drift Detection Method (DDM) and Early Drift Detection Method (EDDM) are reviewed in [6]. In [8], the fisher score—a statistical measure capturing variation in specific features—is used for drift detection instead of relying on the model's online error rate. In addition, [6] detects drift by monitoring shifts in the incoming user traffic using descriptive statistics such as data range, mean, mode, median, standard deviation, and variance. The authors of [5] introduced a hybrid approach that tracks both error rate as well as changes in the incoming data. All these methods require large training dataset and high computational resources, and also prone to false positives/negatives causing over- or under-provisioning of resources and thereby leading to SLA violations.

This demonstration presents an advanced drift-handling framework – *drift detection*, *drift analysis*, and *drift adaptation* – designed

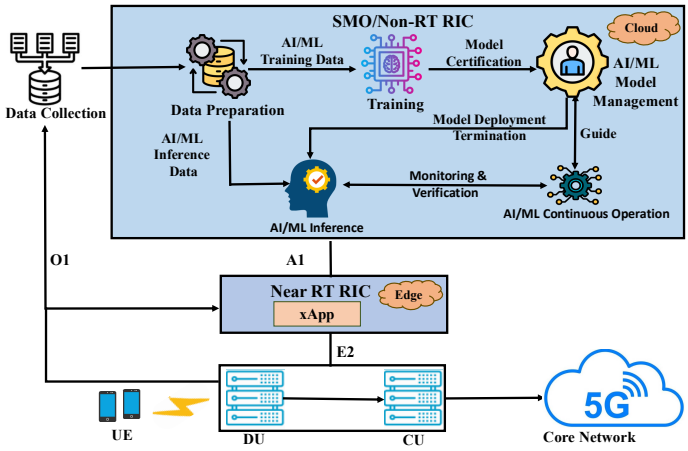


Figure 1: A Drift-Handling Framework in O-RAN

for O-RAN environments. Leveraging automated real-time data collection from open interfaces and AI/ML model training and inference at the RAN intelligent controller (Non-RT RIC), the proposed framework swiftly detects the model drift. Once drift is detected, the framework dynamically selects and implements the most appropriate adaptation strategy: rapidly retrains the model if it is the case of *sudden drift* occurrence or automatically replaces the trained model with an alternative model if it is the case of *incremental drift*. Through this dynamic approach, we demonstrate how the integration of real-time drift detection along with automated adaptation can significantly improve the resilience and efficiency of O-RAN, thereby helping the mobile operators to maintain optimal performance despite evolving user traffic and dynamic network conditions.

2 Drift-Handling Framework

Figure 1 shows the proposed draft-handling framework in O-RAN. RICs provide more granular control and optimization of resource allocation at various operational levels by managing the RAN through Non-RT Applications (*rApps*) and Near-RT Applications (*xApps*). The Non-RT RIC, part of the Service Management and Orchestration (SMO) framework, is responsible for the AI/ML model life-cycle management: data collection, data preparation, model training, model management, model inference, and continuous operation [2], as well as continuous monitoring of *x/rApps* for detecting model drift. The Near-RT RIC provides near-real-time control, optimization and monitoring of O-CU and O-DU nodes via *xApps* [7].

Data collection is a process by which telemetry data (e.g., Uplink/Downlink user throughput) is gathered from various O-RAN entities through open interfaces (E2, O1/O2) and stored inside a time-series database like influxDB. *Data preparation* occurs in an ETL (extract-transform-load) process which comprises of extracting data sources stored in storage repositories such as buckets, volumes, and data lakes, transforming it by cleaning, structuring, and loading the data into target environments.

During the model training, the chosen ML model (e.g., throughput prediction model) is trained, evaluated, and validated to guarantee accuracy and reliability. This itself can be enhanced by optimizing the hyper-parameters of the model. *Model management* is a service where trained models are stored in a centralized model catalog and used for inference purposes. The models are packaged with the appropriate artifacts, metadata, and all supporting information needed to deploy, replicate, and update in the future.

The *AI/ML continuous operation* component facilitates the continuous enhancement of AI/ML models over the AI/ML workflow life-cycle. It is the component that monitors the performance of all other workflow components by continuously monitoring on the processes and the models' performance. It gathers feedback of the system and performs the necessary steps including re-training and replacement of models on the basis of the drift observations, thus facilitating the adaptive and efficient incorporation of AI/ML models in O-RAN.

3 Live Demonstration

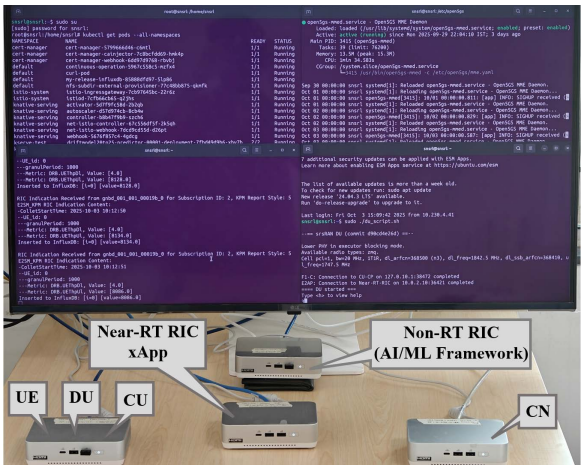


Figure 2: Testbed Setup

Figure 2 shows a snapshot of the testbed setup. This live demo incorporates an Open5GS instance (the core network) and an srsRAN deployment which includes one instance of srsCU, srsDU, and srsUE. The O-RAN Software Community (SC) Near-RT RIC (*i release*)

is deployed as a multi-container app following the O-RAN Alliance specs. The srsCU connects to the core network, and the srsDU interfaces with both the srsCU and the Near-RT RIC. The srsUE talks to the srsDU over-the-air using ZeroMQ (ZMQ), to emulate RF but without actual hardware. A continuous *iperf* session is used to generate the user traffic and the RIC makes use of the *KPM-MON xApp* to gather real-time performance related stats (i.e., downlink and uplink throughput) on a per-second interval over the E2 interface.

```
root@snsrl:/home/snsrl# kubectl get pods
NAME                                          READY   STATUS
continous-operation-bb6475c78-b7h68           1/1     Running
```

Figure 3: Status of Continuous Operation

The Non-RT RIC uses Kubernetes (K8s) for the AI/ML workflow, whose deployment was carried out on an Ubuntu 24.04 mini PC having 22 CPU cores, 64 GB RAM, and enough storage to handle the datasets used for training AI/ML models. It supports an AI/ML framework (*L release*) that allows managing the lifecycle of AI/ML models: training models, retraining, data ingestion, and drift detection. The data collected through O1/E2 interfaces are pre-processed and used to train an LSTM model for throughput prediction which is suited for sequential and time-series data. Once trained and validated, the model and its artifacts are packaged within the model management component, which stores it in a versioned, cataloged, and packaged format to be deployed or retrained in future.

3.1 Drift Detection Methodology

Once the setup is established, KPIs are continuously fed from *KPI-MON xApp* to the influxDB database pod. Inside the Non-RT RIC, the pre-trained deployed model processes the data streams and forecasts the future values, such as UL and DL *user throughputs*.

The *continuous operation pod* continuously monitors incoming KPIs and compares them against predicted values to detect drift in the AI/ML model employed. The main responsibilities of this pod include *drift detection*, *drift analysis*, and *drift adaptability*. Figure 3 represents the status of the continuous operation pod. For drift detection, this pod employs a Root Mean Square Error (RMSE)-based technique. The pod divides the real-time data into sliding windows and calculates RMSE for each window by comparing the model's predicted outputs to the corresponding real-time values reported through KPIs. When consecutive RMSE values exceed a dynamically tuned threshold – calibrated via Particle Swarm Optimization (PSO) to reduce false positives and maximize sensitivity – the pod flags the occurrence of model drift [4].

The key algorithmic parameters used for the drift detection are as follows: the window size, the total number of windows, $RMSE_{th}$, and $flag_{vth}$ (flag violation threshold — a variable to indicate whether the window is exceeding $RMSE_{th}$ RMSE threshold) are set to 5,5,4.5,0.31, respectively. To optimizing these parameters, Particle Swarm Optimization (PSO) is employed with a swarm size of 20 particles iterating over 50 generations. The inertia weight is set to 0.9 to balance exploration and exploitation, while the cognitive and social coefficients are both set to 2.0, ensuring particles are influenced equally by their own best position and the swarm's global best. Convergence is declared when the improvement in the global best fitness value falls below 0.001 for five consecutive iterations, indicating that a near-optimal set of parameters has been found.

3.2 Adaptation Workflow

When consecutive RMSE violations are detected and the RMSE values are observed to be in a strictly increasing sequence, the proposed framework performs a model replacement using a pre-trained model from the catalog. If the RMSE values change abruptly without an increasing trend, this signals sudden drift and triggers model retraining. The model catalog contains pre-trained models that can be swapped at the granularity of 500 ms. The duration required for retraining depends on the volume of new data collected and the batch size used for training.

3.3 Performance Results

During the live demo, we will show four phases : (i) initial operation – where constant-rate iperf traffic yields stable model predictions; (ii) incremental drift – by changing iperf traffic from constant-rate to strictly increasing pattern; (iii) automatic adaptation – where the model swap happens with a pre-trained model; (iv) sudden drift – by changing iperf traffic from constant-rate to a burst pattern where the model automatically goes for the retraining for adapting to the new traffic pattern.

The proposed framework performs well by combining real-time drift monitoring with automatic adaptation. The considered AI/ML model achieves *precision* (0.9785), *recall* (1.0000), and *F1-score* (0.9891), thanks to the dynamic drift-handling mechanism.

Acknowledgments

This work has been supported by TTDF funded "SMART-RIC6G: Smart Drift-Handling Enabler for RAN Intelligent Controllers in 6G Networks" and DST funded "Synergy Taking Openness to the Next Level in 6G Networks" projects.

References

[1] Abu Hena Al Muktadir and Ved P Kafle. 2020. Prediction and dynamic adjustment of resources for latency-sensitive virtual network functions. In *IEEE Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 235–242.

[2] O-RAN Alliance. 2021. *O-RAN Working Group 2 AI/ML Workflow Description and Requirements.* Technical Report ORAN-WG2.AIML-v01.03. O-RAN Alliance.

[3] Ioannis A Bartsiokas, Panagiotis K Gkonis, Dimitra I Kaklamani, and Iakovos S Venieris. 2022. ML-based radio resource management in 5G and beyond networks: A survey. *IEEE Access* 10 (2022), 83507–83528.

[4] Venkateswarlu Gudepu, Venkatarami Reddy Chintapalli, Piero Castoldi, Luca Valcarenghi, Bheemarjuna Reddy Tamma, and Koteswararao Kondepu. 2024. The drift handling framework for open radio access networks: An experimental evaluation. *Computer Networks* 243 (2024), 110290.

[5] Meenal Jain, Gagandeep Kaur, and Vikas Saxena. 2022. A K-Means clustering and SVM based hybrid concept drift detection technique for network anomaly detection. *Expert Systems with Applications* 193 (2022), 116510.

[6] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering* 31, 12 (2018), 2346–2363.

[7] Michele Polese, Leonardo Bonati, Salvatore D'oro, Stefano Basagni, and Tommaso Melodia. 2023. Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1376–1411.

[8] Kungang Zhang, Anh T Bui, and Daniel W Apley. 2023. Concept drift monitoring and diagnostics of supervised learning models via score vectors. *Technometrics* 65, 2 (2023), 137–149.