WIP: Impact of AI/ML Model Adaptation on RAN Control Loop Response Time

V. R. Chintapalli#, V. Gudepu$, K. Kondepu$, A. Sgambelluri•,
A. Franklin#, B. R. Tamma#, P. Castoldi•, L. Valcarenghi•
#Indian Institute of Technology Hyderabad, India
$Indian Institute of Technology Dharwad, India
•Scuola Superiore Sant'Anna, Pisa, Italy
e-mail: cs17resch01007@iith.ac.in, 212011003@iitdh.ac.in

***Abstract*—The advent of Open Radio Access Network (O-RAN) technology enables intelligent edge solutions for base stations in beyond 5G (B5G) networks. O-RAN Working Group 2 (WG2) focuses on the architecture and specifications of AI/ML workflows, allowing AI/ML applications in O-RAN environments to meet different QoS requirements for different use cases over varying time periods. This study shows the technical challenges in mapping AI/ML functionalities at Near-Real Time (RT) RAN Intelligence Controller (RIC) and/or Non-RT RIC for closed loop control-based resource adaptation in O-RAN. We also present a drift-based solution to avoid performance violations if there is decay in prediction accuracy. Results show that drift-based solution outperforms offline models.**

***Index Terms*—O-RAN, Beyond 5G services, RIC control loops, AI/ML, Drift-assistance.**

I. Introduction

Several consortia, such as Operator Defined Open and Intelligent Radio Access Networks (O-RAN), Telecom Infra Project (TIP), Open RAN Policy Coalition, to name a few, are promoting disaggregated architectures for the Radio Access Network (RAN) for 5G and beyond [1]. These architectures are expected to bring many benefits such as enabling more market competition and customer choice, lower equipment costs, and improved network performance. They are based on two fundamental principles: *openness* and *intelligence*. *Openness* allows multivendor interoperability in RAN technologies, specifically by creating open interfaces between O-RAN components: central unit (O-CU), distributed unit (O-DU), and radio unit (O-RU). *Intelligence* is becoming a key component in next-generation networks deployment to ease the complex network configuration for meeting the dynamic service demands.

In O-RAN, two RAN intelligent controllers (RIC) are defined: the Near-Real Time (Near-RT) and the Non-RT RIC [2]. RICs enable RAN autonomous optimization by introducing three closed control loops operating at different timescales depending on the position of Machine Learning (ML) model inference: (i) control loop at O-DU ($<$ 10ms — loop 1); (ii) control loop at Near-RT RIC ($\geq$ 10ms and $<$ 1s — loop 2); (iii) control loop at Non-RT RIC ($\geq$ 1s — loop 3) [2]. In RICs, prediction models, exploiting AI/ML inference, are used to predict future traffic behaviour for triggering an autonomous network reconfiguration, if needed. Thus, the performance of the prediction models plays a major role in the control loop, particularly when the prediction models exhibit drift or decay in the performance, requiring switching to an improved model or retraining the models. Moreover, the considered prediction model and the resources allocated to run it may have an impact on the control loop. To the best of our knowledge, no prior work investigated impact of AI/ML model adaptation on RAN control loop response time.

This paper focuses on a method for determining whether to retrain or switch to a different trained model based on the nature of the drift. In general, the drift is classified into different categories such as: (i) *sudden*; (ii) *gradual*; (iii) *incremental*; and (iv) *reoccurring* drift [3], [4]. A *sudden* drift happens whenever there is an abrupt change in the data distribution within a short period of time. A *gradual* drift is an extension of *sudden* drift where we can observe another data distribution over a long period of time and performance decay occurs gradually. An *incremental* drift occurs whenever model performance decays and grows as the time progresses. A *reoccurring* drift happens when the performance decay pattern repeats itself (i.e., repeated behaviour) over time.

This paper presents a drift-based solution to avoid performance violations of the application whenever there is a decay in the prediction accuracy of AI/ML inference. In addition, it experimentally evaluates the impact of the AI/ML model adaptation and of the resources available for AI/ML based inference and on the control loop response time.

II. System Model and Proposed Solution

Fig. 1 shows the considered O-RAN architecture. O-RAN provides different interfaces (O1, A1 and E2) for the functional block communication. The *O1-interface* obtains the input data from all the components (O-RU, O-DU, and O-CU), 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 used between near-RT RIC, O-CU, and O-DU. For what concerns the fronthaul-based transport network, multiple Transport Units (TUs) can connect to Transport Node (TN), and TN can allocate the bandwidth to TUs. As shown in Fig. 1, different ML assisted control loops are envisioned

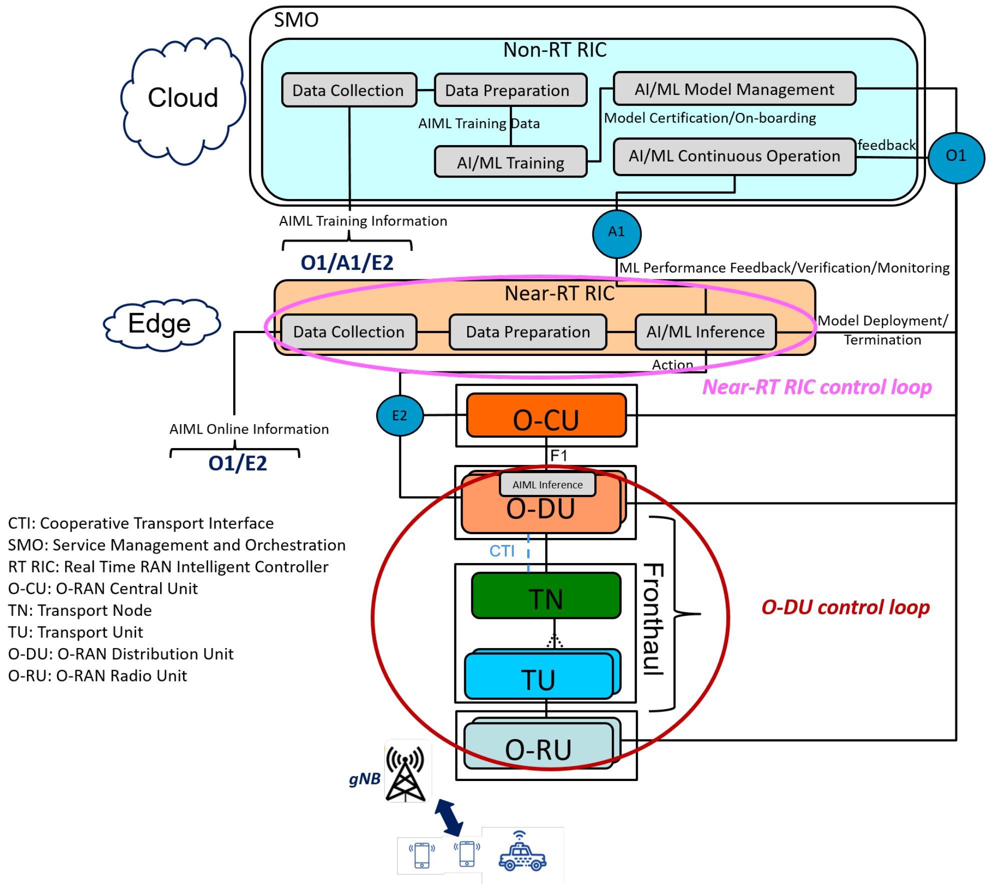

Figure 1. O-RAN architecture realised by fronthaul transport networks.

depending on what is being controlled (i.e., radio resources and policy scheduling, handover, and models). The AI/ML model management block is primarily responsible to determine when to retrain or switch the model for improved performance based on feedback from the AI/ML continuous operation block. In addition, AI/ML model management directs the AI/ML training block when the model needs to be retrained. Non-RT RIC runs at the cloud, whereas Near-RT RIC runs at an edge in the considered scenario. Despite the advantages of the edge over the cloud, compute and storage resources on edge servers are constrained and less powerful when compared to cloud servers. The resources allocated to AI/ML inference at the edge may have an impact on prediction time (e.g., the lower the resources are available, the longer the prediction time).

The traditional AI models performance always depends on adopted training or observed datasets. Whenever the model receives unseen data than the observed one, then resulting in the performance decay [3], [4]. Thus, to maintain the performance guarantee, it is required to either switch to alternative trained models or retrain the existing models. Fig. 2 shows the proposed drift-based solution: (i) AI/ML inference module refers to an instance of prediction model; (ii) Upon decay in the prediction accuracy, determine the type of drift based on the behavior of the prediction values; a window based approach is used to detect the kind of drift by employing a continuous observation of error-rate for each window and helps to trace the drift pattern. (iii) A model retraining is required when the prediction values are sudden or gradual compared to the actual values; (iv) select suitable (pre)trained model when the prediction values are incremental or reoccurring; and (v) the selected or retrained model is replaced at AI/ML inference, and the procedure repeats. During this process, the control loops are influenced by different delays in AI/ML lifecycle management.

Moreover, the prediction time also depends on the weight of the model (i.e., classical models may take prediction time in terms of microseconds and ML methods take more time). Thus, the trade-off between resource usage at the edge and prediction time of the model could be maintained. Continuous observation of the incoming data is required to avoid the performance decay in the model prediction. This could be handled by performing periodic retraining (due to unseen data) [5], however, this may not be appropriate when the user demands changes dynamically [6].

The total time required for AL/ML retraining ($Total_{rt}$) is computed as:

$$Total_{rt} = T_p + T_{ico} + T_{co} + T_{mm} + T_{rt} + T_{mmi} \quad (1)$$

where T_p is AI/ML inference time (i.e., prediction time),

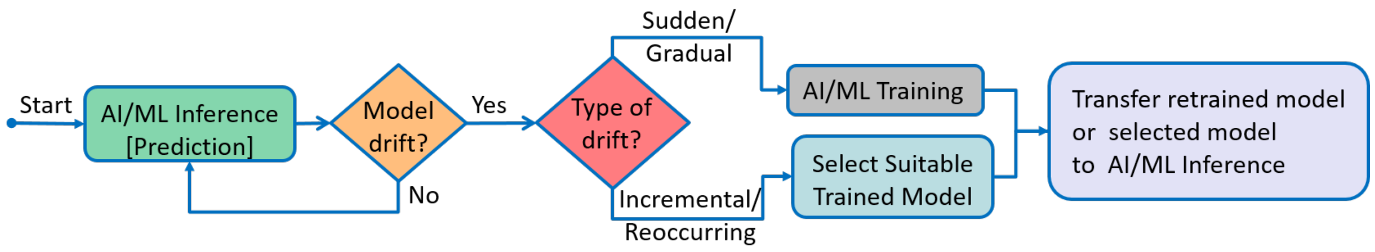
Start
AI/ML Inference [Prediction]
Model drift?
Yes
No
Type of drift?
Sudden/ Gradual
Incremental/ Reoccurring
AI/ML Training
Select Suitable Trained Model
Transfer retrained model or selected model to AI/ML Inference

Figure 2. Drift process to retrain the model or switched to best suitable trained model.

T_{ico} is the delay between AI/ML inference and continuous operation modules, T_{co} is the time spent at the AI/ML continuous operation (T_{co}) module for providing the feedback to the AI/ML model management module, T_{mm} is the time spent on AI/ML model management module to determine whether or not retraining is required, T_{rt} is the time spent on retraining the model at AI/ML training module, and T_{mmi} is the time taken for transferring the retrained model from AI/ML model management to AI/ML inference location. Note that the delays between modules at Non-RT RIC are ignored. All of the aforementioned delays are taken into account, even if switching to an improved model, with the exception of T_{rt}. Prediction time (T_p) plays a key role in meeting control loop even if no change is required to the model. Thus, the following section shows how the resource allocated to run the AI/ML inference affects T_p.

III. Performance Evaluation Results

For evaluation purpose, both traditional time series analysis prediction algorithms such as: double exponential smoothing (DES), triple exponential smoothing (TES), and AI/ML based prediction algorithms such as long short term memory (LSTM) and Transformer (TS) are used to estimate the future traffic [7]. Mathematical description of distinct time series analysis and ML-based techniques are well exploited in [7], [8]. Furthermore, the state-of-the-art AI/ML mechanism, *transformer* is explained in [9]. The dynamic nature of user traffic dataset [10] is considered, and reprocessed to obtain *1ms* granularity. We generate synthetic dataset for analyzing the impact of adopting drift-based approach. The considered performance parameter is the prediction accuracy, represented by the *RMSE* (eq. 2) and the optimal hyper-parameters are selected for each technique.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \qquad (2)$$

where, y_i is real output and $\hat{y}_i$ is the predicted output from respective model.

Table I shows variation in prediction time for different models running with various VM-flavors using *OpenStack* cloud platform. Results show that the resource allocated to the AI/ML inference module has a significant role in meeting control loop requirements. However, time series models exhibit negligible T_p compared to the ML models. Note that ML models require more resources to obtain better T_p. This implies that, due to resource constraints, traditional time series models can be used for O-DU AI/ML inference. However, trade-off between different models and their performance need to be taken into account. Table I also shows the prediction accuracy observed for the considered models. However, the performance of model is highly depends on the dataset provided and hyper-parameters used [7].

Table I
Prediction time and prediction accuracy

VM-Flavor	**DES [*ns*]**	**TES [*ns*]**	**LSTM [*ms*]**	**TS [*ms*]**
Small(1vCPU, 2GB RAM)	0.431	0.392	23.83	25.92
Medium(2vCPU, 4GB RAM)	0.392	0.231	14.31	17.41
Large(4vCPU, 8GB RAM)	0.201	0.101	5.12	7.53
XLarge(8vCPU, 16GB RAM)	0.082	0.085	1.3	1.1
Performance accuracy	**DES**	**TES**	**LSTM**	**TS**
RMSE	37	26	19	11

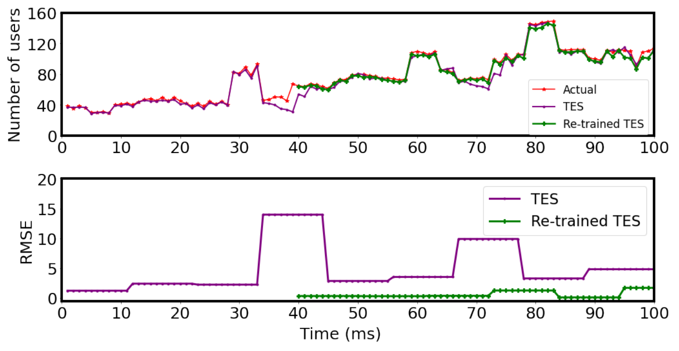

Figure 3. Sudden drift controlled by using retraining the model.

Figs. 3 and 4 depicts the impact of different drifts in performance. Two different prediction approaches are considered by using: (i) prediction without drift assistance (offline model), where no retraining is performed; (ii) prediction with drift assistance, where performance of the model is observed continuously and the required action is taken depending on the nature of the drift.

Fig. 3 depicts the effect of replacing the model with a retrained model in the case of sudden drift. The top plot in Fig. 3 depicts the actual and forecast the number of users of the considered models, while the bottom plot depicts the RMSE

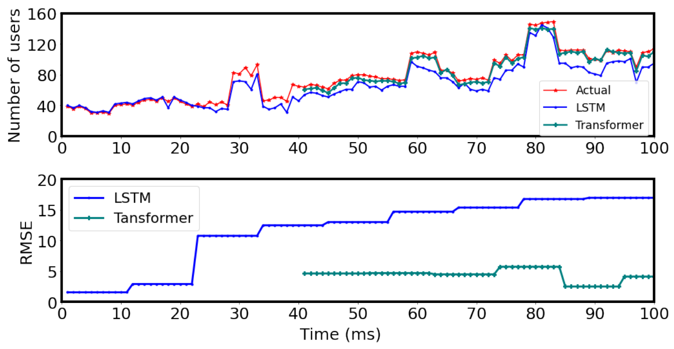

Figure 4. Incremental drift controlled by changing the model.

of the corresponding model. Initially, we chose the TES model based on the performance of the various models. When newly arriving data follows a different data distribution than the trained model distribution, it results in *sudden* drift. Re-training the model with newly arriving data increases model performance in order to overcome this type of drift.

As shown in Fig. 3, TES performance degrades abruptly over the intervals $[32-40]$ ms and $[65-75]$ ms. The type of drift is detected based on the average RMSE of each time window (time window size is assumed to be 10 ms). The sudden drift threshold is set to more than 10 in this case. However, the threshold value may differ based on the service/application under consideration. AI/ML model management re-trained the TES model in the cloud and employed it to predict from the 40 onwards and named as *Re-trained TES*. In addition, we have also shown the performance of previous model in Fig. 3 to show the impact of the retrained model over the previously running *TES* model. The new pattern is observed at time interval $[70-75]$ ms, and it is effectively predicted by the *Re-trained TES* model, whereas the earlier model (i.e., *TES*) failed to detect it.

Fig. 4 shows the impact of changing the model in case of *incremental drift*. Initially, we have chosen $LSTM$. The degradation of $LSTM$ performance starts at the interval of $[24-34]$ ms and grows over time. This type of performance degradation is referred as *incremental drift*. To deal with this type of drift, AI/ML model management module considers different other models in the cloud and found that the $transformer$ model performs better than $LSTM$ and replaces it instead (at 42 ms). Note that the *incremental* and *sudden* drift are emulated by continuously feeding new data.

We considered that the retrained or alternative model is readily available and delays involved in replacing the model is negligible. However, in reality, it takes significant amount of time which depends on the time taken for retraining the model in cloud and the communication delays between the modules for replacing the model. It can be overcome by periodically retrain the model and it is replace the existing model with the readily available retrained model as described in [5]. However, this may not be appropriate when user demands change on a regular basis, and it also depends on the periodic interval in consideration. Predicting when to retrain the model is an interesting open research problem to investigate and we would like to investigate as part of our future work.

IV. Conclusions and Future Directions

This paper studied technical challenges involved in AI/ML model adaptation on control loop response time in O-RAN. Initial study focused on evaluating the impact of prediction time as a function of resources allocated for both traditional time series and AI/ML based forecasting mechanisms. Results showed that the drift-based model could outperform whenever there is decay in performance. Based on this study, potential future directions of this work as follows: (i) *Emulation framework*: we are currently building an emulation platform to emulate drift based mechanism at RICs using open-source tools such as O-RAN and FlexRIC; (ii) *Predict when to Retrain*: retraining on the fly or periodic retraining approaches will not be suitable for time critical 5G and beyond services. To overcome this we would like to investigate to design a mechanism for predicting when to retrain; (iii) *Mathematical analysis:* analysing mathematically the impact of different delays involved in drift-based solutions.

Acknowledgment

This work has been partially supported by the project “Scheme for Promotion of Academic and Research Collaboration (SPARC)”, MHRD, Govt. of India and the EU Commission through the 5GROWTH project (grant agreement no. 856709).

References

[1] L. Bonati, M. Polese, S. D’Oro, S. Basagni, and T. Melodia, “Open, programmable, and virtualized 5g networks: State-of-the-art and the road ahead,” Computer Networks **182**, 107,516 (2020).

[2] O-RAN Working Group 2, “O-RAN AI/ML Workflow Description and Requirements - v1.01, Technical Specification, 2021,” .

[3] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, “Learning under concept drift: A review,” IEEE Transactions on Knowledge and Data Engineering **31**, 2346–2363 (2018).

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” ACM computing surveys (CSUR) **46**, 1–37 (2014).

[5] A. H. Al 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),” (IEEE, 2020), pp. 235–242.

[6] V. P. Kafle and A. H. Al Muktadir, “Intelligent and agile control of edge resources for latency-sensitive iot services,” IEEE Access **8**, 207,991–208,002 (2020).

[7] 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.

[8] J. Martín-Pérez, K. Kondepu, D. De Vleeschauwer, V. Reddy, C. Guimarães, A. Sgambelluri, L. Valcarenghi, C. Papagianni, and C. J. Bernardos, “Dimensioning of v2x services in 5g networks through forecast-based scaling,” IEEE Access (2022).

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in “Advances in neural information processing systems,” (2017), pp. 5998–6008.

[10] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, “A multi-source dataset of urban life in the city of milan and the province of trentino,” Scientific data **2**, 1–15 (2015).