

O-RAN AI/ML Workflow Architecture and Framework

Andrej Čop, Carolina Fortuna, Blaž Bertalaníč

Department of Communication Systems, Jozef Stefan Institute
ac8560@student.uni-lj.si

Abstract

Modern 5G and 6G networks increasingly rely on intelligent AI/ML services for network control and operations. The efficient management of these services requires standardized AI/ML workflow specifications. This paper analyzes an open architecture and an existing solution for such a system proposed by O-RAN Alliance and O-RAN Software Community. We perform experiments using an example workflow, evaluating metrics such as resource utilization and workflow execution time with varying dataset sizes. Our analysis shows a moderate resource utilization with less than 20% of resources being utilized in an idle state, however it identifies some limitations, with data preparation tools failing at larger dataset sizes.

1 Introduction

Radio Access Networks (RAN) are evolving towards more intelligent architectures and increasingly rely on intelligent AI/ML services supporting the adaptability and flexibility of the network control [1]. For integration of artificial intelligence and machine learning (AI/ML) capabilities to RAN, automation of AI is needed, which is already present in cloud systems, referred to as machine learning operations (MLOps) [2]. To support this, the O-RAN Alliance defines specifications for AI/ML workflows which encompasses components that cover the entire life-cycle of AI/ML models in RAN environments, from data pre-processing to model retraining and inference [3]. In addition, the O-RAN Software Community proposes a solution for AI/ML workflows - AI/ML Framework, aligned with the O-RAN architecture and specifications [4].

This paper analyses the architecture of AI/ML workflows as defined by the O-RAN Alliance and examines the AI/ML Framework solution. We conduct a series of experiments using O-RAN SC's example workflow for Quality of Experience (QoE) prediction [4], evaluating key metrics such as deployment time, resource utilization, workflow execution time with varying dataset sizes, and inference latency under different loads. Through this analysis, we aim to provide insights into the practical performance, capabilities, and potential areas for improvement in the current O-RAN AI/ML Framework implementation, such as data preparation tools failing at larger dataset sizes.

2 O-RAN AI/ML Workflow Specifications

In the domain of AI/ML workflows, O-RAN Alliance's Work Group 2 defined components and requirements for workflows on RAN. The latest release of AI/ML specifications was released in July 2021 [3].

Fig. 1 depicts the specified architecture of a general AI/ML workflow as defined by the specifications. These specifications outline a system consisting of five components: data preparation, model training, model management, model inference, and continuous operation. Combined in a workflow, components present AI/ML functionalities with input and output artifacts connecting them.

In O-RAN, these components can be deployed on the Non-RT RIC (Non-Real Time RAN Intelligent Controller) or the Near-RT RIC (Near-Real Time RAN Intelligent Controller). The RIC is an element of the Service Management and Orchestration (SMO) framework and controls essential RAN operations. The Near-RT RIC, situated at the network edge, provides lower latency for network actions [5].

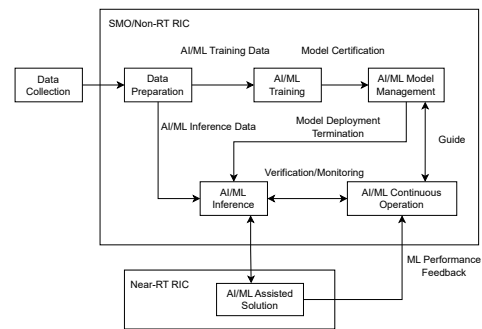


Figure 1: Architecture diagram of an AI/ML workflow, adapted from [3].

Deployed models form an AI/ML assisted solution, which is a service for network functionalities and is commonly deployed on Near-RT RIC for lower response time, as depicted in Fig. 1. RIC, in cooperation with AI/ML assisted solutions, provides control of RAN elements and optimizes resource usage.

A common workflow use case is deploying intelligent models for controlling RAN slicing while retraining the ML models as the network changes. The workflow is

collecting metrics and data from the O-RAN interfaces and storing the datasets with the data collection and data preparation components depicted in Fig. 1. The model is trained or retrained on the latest dataset and deployed as a service on the Near-RT RIC. RIC is performing slicing control using model prediction from key performance metrics of the network, such as latency and throughput. In case any model inefficiencies occur, such as model prediction drift, the continuous operation component triggers the retraining of the model on new data to mitigate the problem [5].

2.1 Data Preparation Component

According to the specifications [3], the data preparation component, shown in Fig. 1, is an ETL (extract-transform-load) process. It consists of storage and extraction, transformation, and loading of data. The loading step involves splitting data into training and testing datasets. Data quality assessment occurs in the analysis stage, while further validation and cleaning can be part of the preparation pipeline. Data can be stored in storage buckets, generic storage volumes, or data lakes [3].

2.2 Model Training Component

The model training component, depicted as AI/ML training in Fig. 1, consists of procedures for the selection and training of an ML model in relation to an AI/ML assisted solution. The model has to be trained, evaluated, and validated. The process of training ML models can be enhanced with hyperparameter optimization, which is an automated process of selecting the best model training parameters.

Model retraining can be triggered by the continuous operation component to retrain the deployed model according to the model performance feedback as depicted in Fig. 1. In the event of needing to retrain, models are fetched from the model catalog, retrained on new data and a new version is published to the model catalog with the model management component [3].

2.3 Model Management Component

The model management component, depicted as AI/ML model management in Fig. 1, is a service responsible for storing the models in the model catalog and deploying machine learning models on the inference host.

Models are packaged with relevant artifacts and meta-data information. In coordination with the model inference component and aligned with AI/ML assisted solutions, model management selects the appropriate AI/ML model for deployment. The model management component can change the model based on performance feedback and terminate the model in situations where degradation of the model performance is detected [3].

2.4 Model Inference Component

The model inference component, depicted as AI/ML inference in Fig. 1, consists of a model inference engine, model deployment phase, and performance monitoring. Model deployment differs for image-based and file-based deployments. In image-based deployments, the model

is packaged as containerized xApp and the inference engine is situated inside the deployed container. File-based deployed model is decoupled from the xApp and is deployed on the inference platform running inside the xApp. In both cases, the inference host can be a Non-RT or Near-RT RIC, depending on the deployment scenario. The deployed model can report Key Performance Indicators (KPI) so that the performance of the models can be monitored for reselection or retraining of the model if necessary.

2.5 Continuous Operation Component

The specifications define the continuous operation component as a set of functionalities for the continuous improvement of AI/ML models in the entire AI/ML life cycle. The continuous operation component positioned in AI/ML workflow architecture is depicted in Fig. 1. This component is responsible for the operation of all other components by monitoring the workflow and model performance. It collects feedback and triggers corresponding events such as retraining and redeployment of the models accordingly.

3 O-RAN SC AI/ML Framework

O-RAN SC (O-RAN Software Community) is developing open-source software for the RAN in collaboration with the Linux Foundation and according to the specifications of O-RAN Alliance. Following the AI/ML workflow specifications, a solution called the *AI/ML Framework* was developed to support AI/ML workflows in O-RAN [4]. Based on definitions from Section 2, their solution consists of multiple interfaces and open-source component modules. At the time of writing this paper, version *1.2.0 I Release* is officially released [6].

Table 1: Open-Source Tools Utilized by AI/ML Framework

AI/ML Component	AI/ML FW Tools
Data Preparation	InfluxDB, Cassandra
Training	Kubeflow
Model Management	Leofs
Model Inference	KServe
Continuous Operation	Kubeflow

Table 1 lists open-source tools utilized by O-RAN SC AI/ML Framework to provide a solution for all components in a workflow. Further, they developed interfaces to connect the components and provide a dashboard user interface for controlling the workflows [6]. The developed software and open-source tools are deployed on a Kubernetes cluster.

For data preparation, they utilize InfluxDB as a data lake, and for storing features a Feature Store SDK interface with Apache Cassandra. Model training is realized with a Training Manager interface and Kubeflow. Training jobs can be created and controlled on the Dashboard Portal. For model management they developed Model Store and Model Management Service interfaces with Le-

ofs as a storage solution. Model Management can register models and retrieve a trained model by model ID. It stores the models as zip files and makes them accessible through the URL address. For model inference they utilize KServe as an inference platform, however, deployment of the models is a manual process. The solution can be integrated with O-RAN SC implementation of Near-RT RIC for the model to be deployed directly as an xApp. The continuous operation component is supported through different interfaces and the Dashboard Portal which provides the creation of training workflows and management of training jobs.

O-RAN SC provided an example workflow for training a QoE (Quality of Experience) model for networks [4]. They provided a dataset example of time series measurements of network metrics such as the amount of uploaded and downloaded *pdcp* bytes, throughput, and network cell identity [4]. Further, they provided model training code for training a deep neural network for QoE prediction using the Keras framework. The example workflow includes data collection from InfluxDB, feature creation, and training of a model for a QoE prediction service. They also provided configurations for deploying the model as a service using KServe and sample data for testing inference service over the API.

4 Methodology

To evaluate the AI/ML Framework we utilized O-RAN SC's example workflow for QoE prediction [4]. The experiments were performed on a Ubuntu Linux virtual machine following the minimum requirements defined in the AI/ML Framework documentation. The virtual machine consists of 16 x86_64 architecture CPU cores and 32 GB of RAM. VM processor is an Intel Xeon 2.00 GHz CPU with one thread per core.

For model training, we used the default hyperparameters provided by O-RAN SC, which are *batch size 10* and *epoch 1*.

The deployment time of the AI/ML Framework in a new environment was measured, which in the context of network services, relates to service instantiating time [7]. Container images were built in advance and the installation scripts were modified to fully automate the deployment. The experiment was repeated 5 times.

After the deployment CPU and RAM requests and limits were collected. These metrics specify resource allocation for containers and are used for scheduling the deployments. Further, the utilization of these resources on the VM shows the overhead of the system in an idle state. Metrics were collected using *kubectl* CLI and *free -m* tools. Additionally, resource utilization was measured at 1s intervals during the workflow execution.

To evaluate the framework under different loads, we scaled the dataset 10 and 100 times. We measured the entire workflow execution time as well as the time to complete data preparation and model training stages. All measured times were collected from the dashboard user interface provided by O-RAN SC.

After the model was deployed using KServe, requests

with sample data were sent to the API endpoint for a QoE model to perform prediction service. To evaluate the inference performance of KServe, a worst-case scenario was tested using parallel request latency testing software and sending up to a thousand concurrent requests to the endpoint. The average end-to-end delay was measured and plotted.

5 Results

Evaluation results of the AI/ML Framework developed by O-RAN SC are presented and discussed in this section according to the methodology presented in Section 4.

On average the system deployed in 1577s with a standard deviation of 269s. Significant variations in the deployment time are expected as software installation and downloading container images is a major part of the deployment procedure.

After the deployment, the Kubernetes resource allocation metrics and utilization of VM resources were collected and are presented in Table 2.

Table 2: Requested and utilized resources of AI/ML Framework after the deployment in an idle state.

	CPU /16 cores	RAM /32 GB
Requests	2.8	5.2
Limits	8	8.7
Utilization	3.2	5.3

We can see that utilization of CPU in an idle state is 2.8 out of 16 cores which is $\approx 20\%$ of the entire capacity while the Kubernetes limits for all deployments are set at half of the capacity, leaving enough free resources for training scheduling and inference services on the rest of the cluster. Similarly, RAM utilization is 5.3 GB out of 32 GB, which is $\approx 17\%$.

In Fig. 2 are presented execution time for data preparation and model training on different dataset sizes when running the example workflow on the AI/ML Framework.

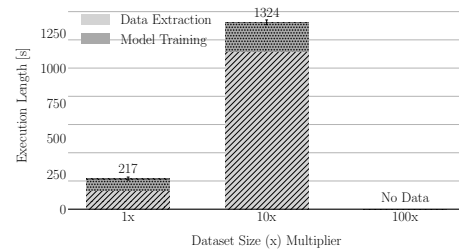


Figure 2: Execution length for different stages of the QoE prediction workflow.

Workflow execution time increases with larger dataset sizes. For original dataset size it takes 217s, for 10 times the original dataset size it takes 1324s and for 100 times the original dataset size, the workflow execution fails, presumably due to timeouts in the data extraction stage,

which were observed in the logs. The data extraction component takes $\approx 70\%$ longer than model training, and for larger datasets data extraction component is more than 4 times slower than model training. This indicates a potential bottleneck in the data extraction component, which is inefficient as the dataset size increases.

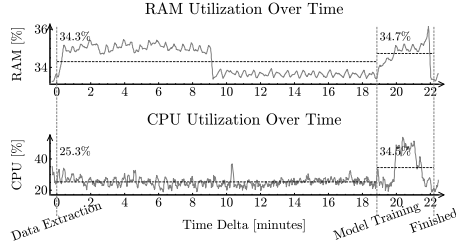


Figure 3: Resource utilization during QoE prediction workflow execution.

Fig. 3 shows resource utilization during the workflow execution, with timestamps indicating the start of each workflow stage and an average utilization during data extraction and model training. Average RAM utilization is 34.3% and 34.7% during data extraction and model training respectively.

There is a clear distinction between RAM utilization in the two parts of the data extraction component at ≈ 9 minute mark. O-RAN’s system uses InfluxDB as a data lake and Cassandra as a feature store, as discussed in Section 3. The first part of the data extraction RAM utilization graph in Fig. 3 is related to InfluxDB and the second part to storing and extracting features from Cassandra. This also indicates that Cassandra doesn’t utilize a measurable amount of RAM in this setup. By relying only on disk I/O operations Cassandra is slowing down the data extraction phase.

As it can be seen in Fig. 3 the average CPU utilization during data extraction utilization is 25.3% and during model training is 34.5%. CPU utilization increases for model training, but is well below the capacity of the VM.

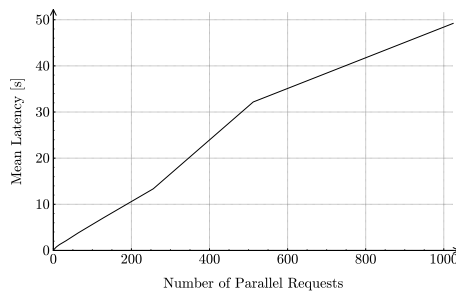


Figure 4: Mean latency for concurrent inference requests.

Fig. 4 depicts mean end-to-end latencies for parallel inference requests on the deployed QoE prediction model. By increasing the number of concurrent requests sent to the API endpoint, the mean latency increases. At

200 concurrent requests the latency is $\approx 10s$ and 500 concurrent requests the average latency is around $\approx 32s$.

The latency increases approximately linearly with the number of parallel requests. The system can handle 1000 concurrent requests without errors, while the mean end-to-end latency does not exceed 50 seconds.

6 Conclusion

In this paper, we analyze the O-RAN AI/ML workflow architecture and its open-source implementation by the O-RAN Software Community. Our evaluation of the AI/ML Framework, using a QoE prediction example, revealed strengths and areas for improvement. The Framework showed reasonable deployment times, averaging 1577 seconds, and moderate resource utilization in an idle state, with CPU and RAM utilization at 20% and 17% respectively. However, we identified limitations in the data extraction phase, which took up to 4 times longer comparing to model training for 10 times the example dataset size and failing to process data 100 times the original size. The model inference, handled up to 1000 concurrent requests with linearly increasing latency, reaching up to 50 seconds for the highest load.

In our future work, we will focus on an alternative solution for an AI/ML workflow system according to the workflow architecture presented in this paper and to address the shortcomings of the existing solution.

Acknowledgment

This work was supported by the Slovenian Research Agency (P2-0016) and the European Commission NANCY project (No. 101096456).

References

- [1] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. J. A. Zhang, “The Roadmap to 6G: AI Empowered Wireless Networks,” *IEEE Communications Magazine*, vol. 57, no. 8, 2019.
- [2] J. Diaz-De-Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, and A. Almeida, “A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey,” *ACM Computing Surveys*, vol. 56, no. 4, 2023.
- [3] O.-R. Alliance, “O-ran working group 2 ai/ml workflow description and requirements,” *ORAN-WG2. AIML. v01.03*, July 2021.
- [4] (2024) O-ran sc documentation. Accessed on January 29, 2024. [Online]. Available: <https://o-ran-sc.readthedocs.io>
- [5] M. Polese, L. Bonati, S. D’Oro, S. Basagni, and T. Melodia, “Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges,” *IEEE Communications Surveys and Tutorials*, vol. 25, no. 2, 2023.
- [6] (2023) Ai/ml framework. Accessed on January 29, 2024. [Online]. Available: <https://wiki.o-ran-sc.org/pages/viewpage.action?pageId=57377181>
- [7] H. Khalili, A. Papageorgiou, S. Siddiqui, C. Colman-Meixner, G. Carrozzo, R. Nejabati, and D. Simeonidou, “Network Slicing-aware NFV Orchestration for 5G Service Platforms,” in *2019 European Conference on Networks and Communications, EuCNC 2019*, 2019.