A Secure and Reusable Artificial Intelligence Platform for Edge Computing in Beyond 5G Networks

D3.1 Initial report of systems and methods for AI@EDGE platform automation

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 10101592

D3.1 Initial report on system and methods for AI@EDGE platform automation	
WP	WP3 – AI@EDGE platform for network automation
Responsible partner	RISE Research Institutes of Sweden AB (RISE)
Version	1.0
Editor	Bengt Ahlgren (RISE)
Authors	Flávio Brito (EAB), Wei JIANG (DFKI), Neiva LINDER (EAB), Daniel RETI (DFKI), Roberto RIGGIO (RISE/UNIVPM), Akhila Rao (RISE), Anders Lindgren (RISE), Nimish SORATHIYA (DFKI), Cristina CERVELLÓ-PASTOR (UPC), Estefanía Coronado (i2CAT), Jérôme François (INRIA), Omar Anser (INRIA), Jonathan Proietto-Stallone (INRIA), Antoine Petit (INRIA), Salah BIN RUBA (CNAM), Chi-Dung PHUNG (CNAM), Alessio DIAMANTI (CNAM), Stefano SECCI (CNAM), Miguel CATALAN-CID (i2CAT), Daniel F. Perez-Ramirez (RISE), Cristina Costa (FBK), Arfan Wahla (FBK) Rasoul Behravesh (FBK), Nicola DI PIETRO (ATH)
Reviewers	Françoise SAILHAN (CNAM) Javier MELIAN HERNANDEZ (ATOS) Luigi GIRLETTI (ATOS) Jovanka ADZIC (TIM) Roberto Riggio (RISE/UNIVPM)
Deliverable Type	R
Dissemination Level	PU
Due date of delivery	31/12/2021
Submission date	31/01/2022

Version History				
Version	Date	Authors	Partners	Description
0.1	30/06/2021	Roberto RIGGIO	RISE	First internal draft for milestone MS3.1
0.2	29/09/2021	Bengt Ahlgren	RISE	Second internal draft for Milestones 3.2 and 3.3 (15 and 16)
1.0	31/01/2022	Bengt Ahlgren	RISE	Final version
1.0	31/01/2022	Irene Facchin	FBK	Final review

Table of Contents

List of Tables

List of Figures

Glossary	
AI	Artificial Intelligence
AIF	Artificial Intelligence Function
AMF	Access and Mobility management Function
CL	Closed Loop
CP	Control Plane
CPU	Central Processing Unit
CU	Centralized Unit
DL	Deep Learning
DNN	Data Network Name
DU	Distributed Unit
FaaS	Function as a Service
FL	Federated Learning
GPU	Graphic Processing Unit
ICT	Information and Communication Technology
IIoT	Industrial Internet of Things
IoT	Internet of Things
KPI	Key Performance Indicator
MEC	Multi-access Edge Computing
ML	Machine Learning
NFV	Network Function Virtualization
NFVO	Network Function Virtualization Orchestrator

NSAP	Network and Service Automation Platform
ORAN	O-RAN Alliance, developing open RAN standards
QoS	Quality of Service
RAN	Radio Access Network
rAPP	Microservice on the non-RT RIC in ORAN
RIC	RAN Intelligent Controller
RSRP	Reference Signal Received Power
SBA	Service-Based Architecture
SDN	Software-Defined Networking
SMF	Session Management Function
UC	Use Case
UL	Uplink
UPF	User Plane Function
VM	Virtual Machine
xAPP	Microservice on the near-RT RIC in ORAN
4G, 5G, 6G	Fourth, Fifth, Sixth Generation of cellular networks

Executive Summary

An initial report is given on the systems and methods developed for the automation of the AI@EDGE connect-compute platform. The reported achievements are the progress towards project overall Objective 3 on a general-purpose network automation framework.

The Network and Service Automation Platform (NSAP) hosts the network automation functionality within the overall AI@EDGE system architecture. The NSAP makes use of the concept of closed-loop network intelligence, where the goal is to relieve the human operator as far as possible from the need to take manual action to operate the system. The concept of closed-loop control is described by a number of steps that form the loop, including sensor, monitor, aggregator, training/inferring, decision making, action enforcer, orchestrator and actuator.

The concept of closed loops are being studied in several standardisation bodies, including ETSI, O-RAN, ONAP, TM-Forum, and ITU-T. The project takes a closer look at the designs and activities developed within TM-Forum and O-RAN.

A structured approach to providing data to the data-driven methods and algorithms for network automation is central to the developed Network and Service Automation Platform. This is manifested in the concept of data pipelines that provide preprocessed system monitoring and other data to multiple functions that need data from the same source. A set of entities in the NSAP support the data pipeline system: data source, data collector, data repository, and data analytics & insights catalogue. As system monitoring often has to be done by parsing of text log files, automated methods for parsing and extracting the essential information is studied.

A number of methods for automation and learning in MEC and cloud systems are developed. A group of methods provides predictive resource monitoring for service placement, often involving forecasting or prediction of performance metrics in the near future. Another group of methods provides advanced support for AI-enabled applications, for example, detection of anomalous events, methods involving federated learning, distributed and collaborative service placement, and data augmentation to increase robustness.

The data-driven methods and algorithms do not work without the appropriate data sources. Relevant identified data sources include container-level data, physical server-level data, eNodeB and UE level data, WiFi AP-level data, application server-level data and network performance real-time monitoring data.

1. Introduction

This deliverable is an initial report on the systems and methods developed in the project for the automation of the AI@EDGE connect-compute platform. It covers the identification of technical and architectural requirements of a framework for closed-loop network automation, systems aspects of automated network operations and management with scalable data exchange models, and methods for data-driven local and global AI/ML models for performance prediction, resource management, and orchestration of application components.

The achievements reported in this deliverable are the progress towards fulfilling the project's overall Objective 3 on designing and implementing a general-purpose network automation framework, capable of supporting flexible and reusable pipelines for the end-to-end creation, utilisation, and adaptation of secure and privacy-preserving AI/ML models. This objective is detailed in the four work-package objectives: (i) defining the technical requirements specific to the methods and systems mechanisms planned for development relative to the overall AI@EDGE concepts and architecture; (ii) defining the systems-oriented methods for developing reusable data models, scalable data propagation and information-exchange, container deployment strategies and processes, as well as adaptive security approaches implementing data protection and service isolation; (iii) designing the learning methods for secure and resilient infrastructure management and performance prediction, largely based on AI/ML for the purpose of supporting automated and adaptive deployment processes and resource allocation in line with application specific requirements; and, (iv) realizing the tools facilitating development and operation for stakeholder users (application developers, service providers/network operators) at the application level.

The progress on the work-package objective (i) is largely reported in Section 2 on the AI@EDGE Network and Service Automation Platform (NSAP). The NSAP is a framework for closed-loop automated network management that provides an environment for data-driven, intelligent, methods that support decision making. The progress on objective (ii) is reported in Sections 3 and 5, on scalable and secure data pipelining and on identified data sources. The data pipeline is an important system component of the NSAP that provides the data needed by the respective decision making method. The progress on objective (iii) is reported in Section 4 on intelligent methods for automation and learning for network management purposes. These methods are modelled in the project as AI functions, or AIFs, as described in more detail in project deliverables 2.1 [D2.1] and 4.1 [D4.1]. The AIFs can serve different roles, for example, providing a network automation function, providing a data pipeline function, or providing an application function. The methods and algorithms described in this deliverable form an initial set of AIFs that is anticipated to be extended. In addition to the ones described in this deliverable, AIFs are also described as part of the development of the connect-compute platform (WP4) and of the use cases (WP5). The research addressing objective (iv) will start in project month 13, and will thus be reported on in future deliverables.

2. AI@EDGE Network and Service Automation Platform

This section presents the proposed AI@EDGE system architecture and its relation to the Network and Service Automation Platform (NSAP) for data-driven intelligent automated network management as a basis for the rest of the WP3 work in this D3.1 deliverable. The brief description in this section will be referenced in the following sections. As an example, in Section 4, several methods for automation and learning in MEC system and for AI-enabled applications are presented, along with how these methods can be related to the described AI@EDGE system architecture shown in this section. Furthermore, in Section 3, the data pipeline system is presented and it is explained how it integrates with the description given in Section 2.1.

This section is organized as follows: Section 2.1 presents in a general approach the AI@EDGE system architecture and describes the NSAP and the Connect Compute Platform. Section 2.2 describes the closed-loop network intelligence design and describes each type of data relevant to the closed loop: sensory data, monitor data and aggregated data; and each function of the proposed general closed loop: tactic, reaction, and performance/reward. In Section 2.3, a description of how such concepts are applied in the AI@EDGE project is described. Section 2.4 deals with how closed loops are defined in standardization projects such as O-RAN ZSM architecture and TM forum, while Section 2.5 explains how the AI@EDGE system relates specifically to the O-RAN architecture.

2.1 AI@EDGE System Architecture

This section describes the overall AI@EDGE system architecture defined in WP2. It will be used as a reference for the rest of the deliverable D3.1 and for elaborating the WP3 work to showcase how it fits into the overall architecture. Figure 1 is a block diagram of the proposed system architecture for AI@EDGE, which is composed of two main components: the Connect-Compute Platform and the NSAP. The former is addressed by WP4 and the latter is addressed by WP3.

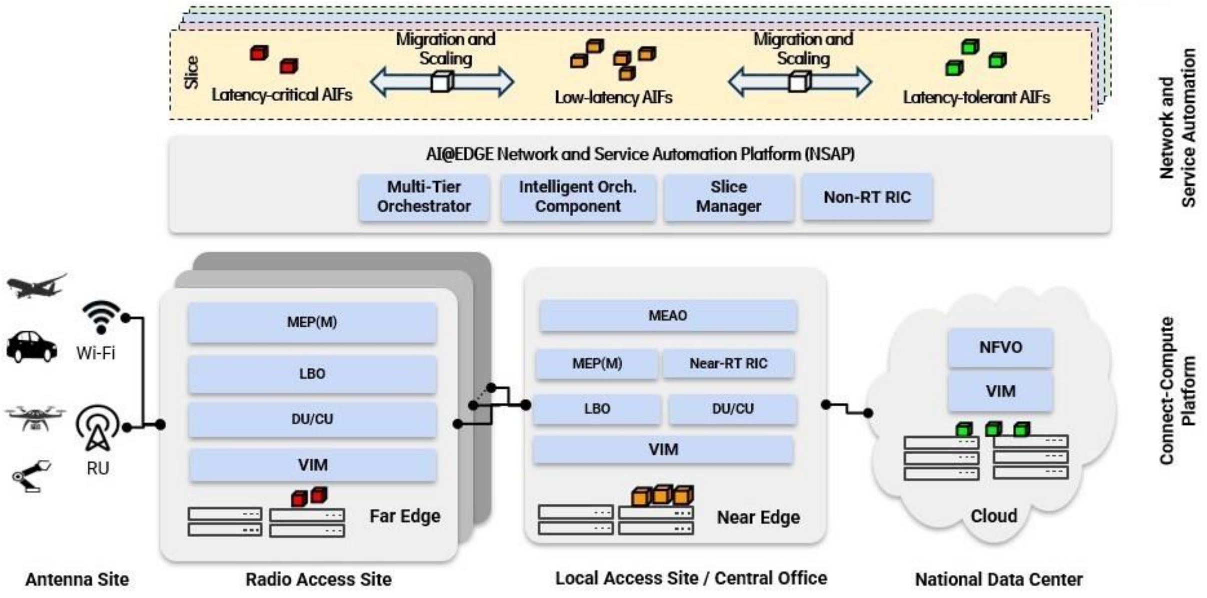

Figure 1 AI@Edge system architecture design.

The Connect-Compute Platform is composed of the Cloud, Far Edge, and Near Edge. Each component is responsible for running AIFs with different latency tolerances: high latency tolerance, low latency tolerance, and latency-critical tolerance respectively. The cloud can use heavy virtualization technology such as OpenStack and VMs since the applications running in that component are latency tolerant. However, such virtualization technologies are not expected to run in Near Edge and Far Edge. Therefore, more lightweight virtualization methods should be used such as containers and FaaS. Also, for orchestration, Kubernetes or microk8s[1] are expected to be deployed in these domains.

Another important concept in the AI@EDGE system architecture is the closed-loop used to manage network functionality on different timescales. The loop is composed of an Autonomic Manager responsible for the management of the closed-loop and the definition of the decision criteria used for the various processes. The decision taken is sent to the Orchestrator, which implements in the system architecture the decision made by the Autonomic Manager onto the connect-compute fabric. Closing the loop is a block with a monitor responsible for collecting the data and an aggregator that collects the data from several monitors and aggregates it in one entity. All this aggregated data is used as input for artificial intelligence algorithms to determine which decision should be taken to improve network performance or to fix specific problems that were detected. The closed-loop concept can run in multiple instances and in multiple locations in the architecture, depending on latency and other requirements. The closed loop functionality and components that implement it are shown in Figure 2 and are described in more detail in Section 2.2.

To provide computational resources to the NSAP, the slice manager block will provide slices that cover all the domains of the Connect-Compute Platform (e.g, from Far Edge to the Cloud). Also, the NSAP component is responsible for the integration of four components: Non-RT RIC, Slice Manager, Intelligent Orchestration and Multi-tier Orchestration. The intelligent orchestration and multi-tier orchestration are implemented through the use of AIFs in the different closed-loop components to make orchestration decisions across the different domains with relevant underlying orchestration frameworks as described above. The non-RT RIC and its relation to the AI@EDGE architecture are described in more detail in Section 2.5 and in deliverable D4.1.

2.2 Closed-loop network intelligence

In this section, we will look into the role of closed loop control in the context of the AI@EDGE NSAP. A closed-loop control for network intelligence starts from the monitor collecting information from network sensors and terminates at performing the actions by means of, e.g., deploying AIFs, SDN/NFV actuators, etc. When an intelligent network task is launched, it runs as a continuous closed loop. If from the analysis of the network and/or service data, a network problem or anomaly is detected or predicted, actions to deal with that can be triggered before the task goes back to monitoring the system again.

[1] https://microk8s.io/

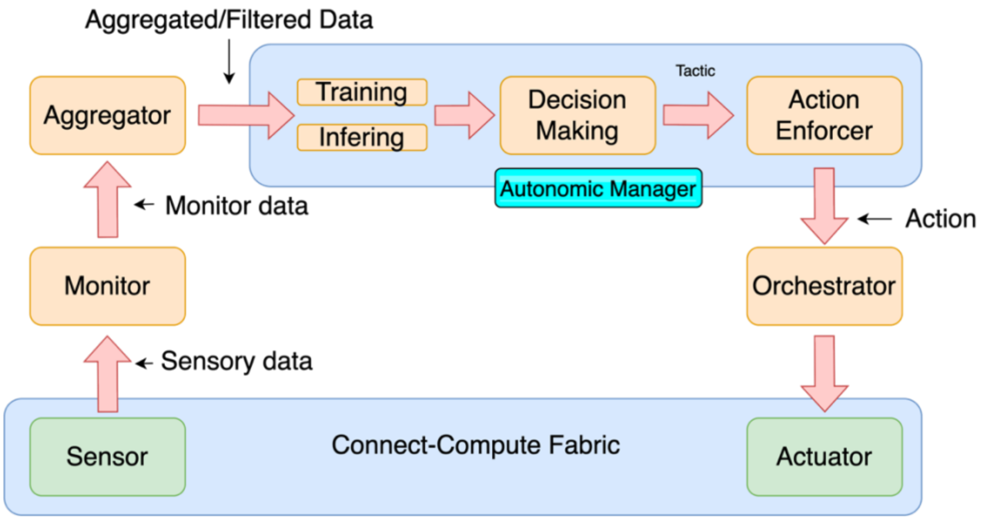

Figure 2 Network automation closed-loop control loop.

The involved functional blocks through the whole intelligence control loop, as well as its information flow, are illustrated briefly in Figure 2. Each functional block receives the required information from its previous block, derives a higher level of information and passes to the next functional block. The information processed by each functional module is defined as follows:

- **Sensor Data**: Different data sources can be identified. Data retrieved from physical devices, AIF modules, data plane, SDN controller, SDN/NFV network function models, and Virtual Infrastructure Manager (VIM) can be referred to as sensor data. The Monitor is the corresponding module charge of collecting sensor data from underlying infrastructures. Some sensor data is periodically reported, while others are occasionally collected triggered by some network events or situations.
- **Monitor Data**: The Monitor regularly collects the sensor data, filters out redundant information, and reports the necessary monitor data to the Aggregator, where the raw data from sensors are aggregated and correlated in order to provide high-level metrics. Some sensor data are periodically generated and collected at high volumes during normal operation of the network. Such data has much redundancy, so the Monitor needs to filter some redundant information and extract the informative data for the aggregation.
- **Aggregated Data**: The monitor data related to a network problem/task/anomaly may be retrieved from a set of sensors, rather than a single one. For example, in the case of distributed Denial of Service attacks, the source and destination are distributed. The raw information contained in monitor data should be processed to produce aggregated and correlated information, which can be called Aggregated Data. The Aggregator performs aggregation and correlation of low-level monitor data provided by the Monitor. This step may involve performing different processing such as data normalization, verification or correlation. In order to facilitate the later processing stages and simplify the workflow, redundant information would also be discarded.
    - The function of the Training and Inferring blocks are to train AI models on the aggregated data and use those models to infer and identify network behaviours and patterns required by the decision-making framework or infer the required actions directly. For this purpose, a comprehensive analysis of the received data is performed, which includes functions such as AI/ML-based knowledge acquisition and reasoning.

- **Tactic**: When the root cause/pattern/behaviour of network problem/anomaly is determined and reported to the Decision Maker, one countermeasure/configuration/command can be decided to solve the problem. The tactic is a high-level description of such a countermeasure. Although a tactic is not fully detailed, it contains the necessary metadata for the action enforcement, such as the target locations, the effective period of the action, high-level configuration information etc.
- **Action**: It can be regarded as an implementable version of the tactic requested by the Decision Maker. The target location of the action should be passed to the Orchestrator by the Action Enforcer. The location contained in metadata of action might be more detailed and specific than that of tactic, for example, the IP address of the device. The type and ID of AIFs, SDN APPs, SDN controller, or NFV APPs might be provided by the Action Enforcer to the Orchestrator, while the Orchestrator could decide implementation of the APP type in terms of the available resources. Besides, the lifecycle of the AIFs/APPs/network functions such as the time to start and stop, the duration, as well as the configuration should be provided by the Action Enforcer.

The network intelligence not only decides an action for a network problem/task to be executed upon the underlying infrastructure but may also need to receive the feedback of the network responses afterwards. The response of the network can be regarded as an achieved performance or reward. The motivation of collecting the achieved performance/reward is three-fold: i) if an action achieves a worse performance, which degrades the performance rather than alleviating the impact of the problem, a roll-back mechanism could be triggered to recover the network status to some of the status points before the action was performed; ii) the achieved performance can be regarded as the evaluation/result of a selected action, whose records with operational data could be used to train the network intelligence; iii) some intelligence algorithms, for example, the reinforcement learning needs the reward of each possible action to learn and find out the optimal solution. In order to get the performance/reward information, the Monitor should continuously observe the underlying networks. After an action is carried out, the Monitor collects and reports the performance achieved by the action to the decision-making framework.

2.3 Closed Loops and AI@EDGE System Design

AI@EDGE will incorporate closed-loops into its architecture as an enabler for the automation of network management processes. The proposed closed-loop system suggests three different closed-loops: the Resource CL, NSAP CL and Cross-Domain CL. Each CL differs based on its domain. The Resource CL only manages and control aspects regarding each Far Edge site while the NSAP CL can manage only the NSAP domain. As these different CLs do not interact with each other (Resource CL does not interact with NSAP CL), the Cross-Domain CL was proposed to be able to collect information regarding all domains (NSAP, Cloud, Near Edge and Far Edge). This CL can be used to control in a higher level the aspects regarding all domains of the AI@EDGE system architecture and can interact and manage with NSAP CL and Resource CL by giving inputs or instruction to change the current workflow of these CL. Figure 3 describes the location of operation of each CL. In red is the Resource CL, in blue is the NSAP CL and in orange is the Cross-Domain CL. It is possible to note that only the Cross-Domain CL can interact with each Resource CL and NSAP CL.

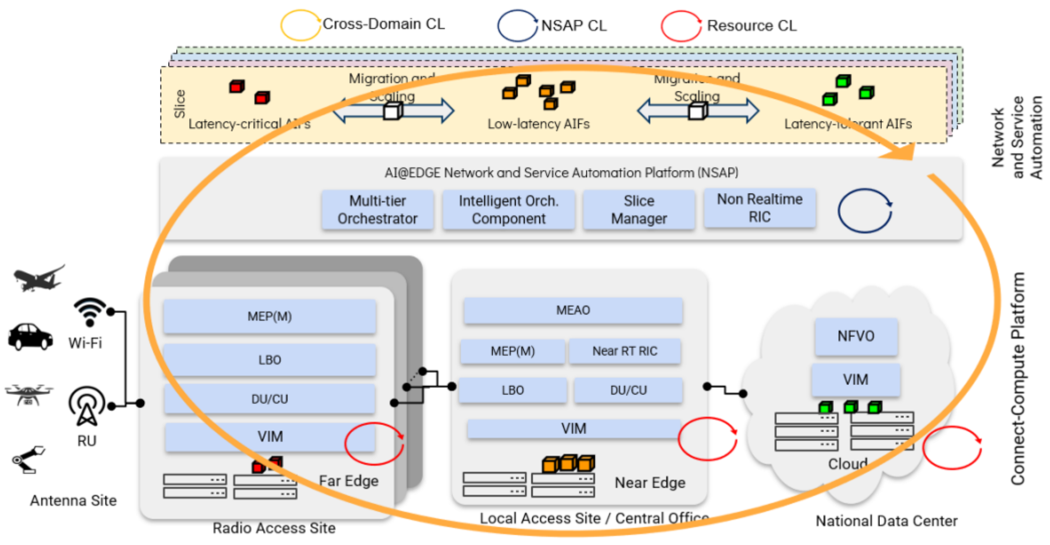

Figure 3 AI@EDGE system architecture include the Closed-Loops.

In Section 2.5, we will describe how CL is proposed in the RAN domain in detail. This can be seen as a type of Resource CL, where the term “Resource CL” is a more general CL and can contain different types of CL such as these ones that will be deployed in the RAN domain. The project will not actively work on these RAN CLs, but will monitor their progress in standardization and how it relates to the AI@EDGE architecture.

2.4 Closed control loops in standardization

Autonomous Networks architecture and the associated closed control loops have gained a lot of traction recently across standardization bodies, such as ETSI, O-RAN, ONAP, TM-Forum, and ITU-T. For the AI@EDGE project, this is an opportunity to learn together with industry and certainly drive research on key solutions that will enable the autonomous networks realization.

In the context of Autonomous Networks, a closed loop is defined in the TM Forum by “a framework in which outputs of a system, workflow, or process are circled back and used as inputs in a way to improve the next output. There are FAST and SLOW closed-loops, and the difference between FAST versus SLOW closed-loops is that FAST Closed-loops function on real-time inputs/data, while slow closed-loops function on historical data and/or user feedback” [Boasman-Patel]. When applied to telecommunication networks, different control loops can be defined as shown in the Autonomous Networks Framework, also proposed by the TM Forum [Boasman-Patel].

To illustrate the importance of closed-loops in standardizations, Figure 4 shows a description of the block diagram of the proposed autonomous network framework proposed by TM forum. There is a closed loop for each autonomous domain represented here as autonomous domain X and Y. In addition, there are the Service Closed Loop and Business Closed Loop and a main closed loop which covers the entire framework. Service closed-loop runs in the service application layer. Business closed-loop works on the business level. Besides all these closed-loops, a larger one covers all the domains of the TM Forum reference system architecture.

The idea proposed by TM forum is to have Autonomous Domains that are self-contained, self-managing, self-optimizing, and self-healing. One key requirement is that the Autonomous Domain and the closed loop operate across different scopes [Boasman-Patel]. This is shown in Figure 4. Therefore, closed loops are a key technique to reach the autonomous network scenario as proposed by TM forum.

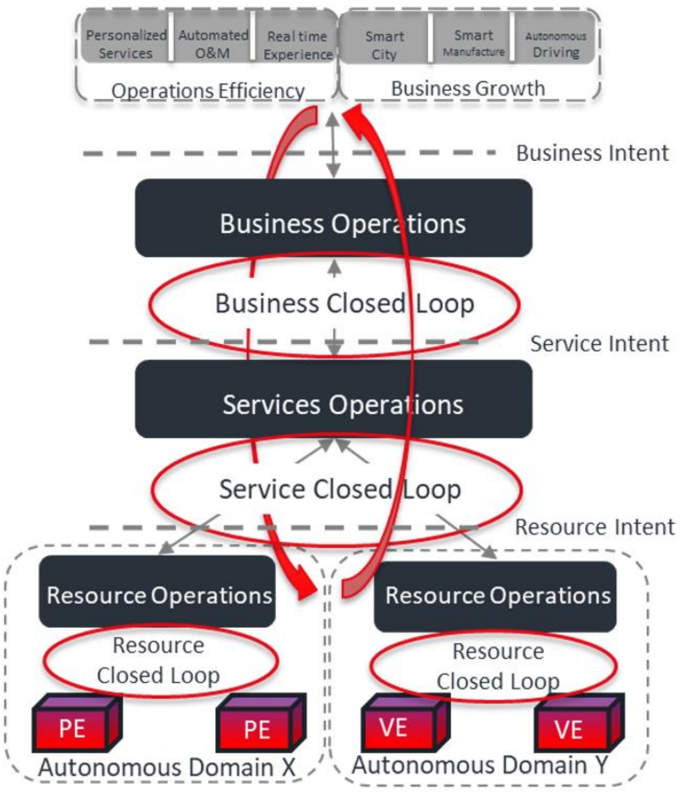

Figure 4 Autonomous Networks Framework from TM Forum [Boasman-Patel]

Figure 5 describes the TM Forum Autonomous network architecture target. It is possible to note the closed-loops on the left side of the picture.

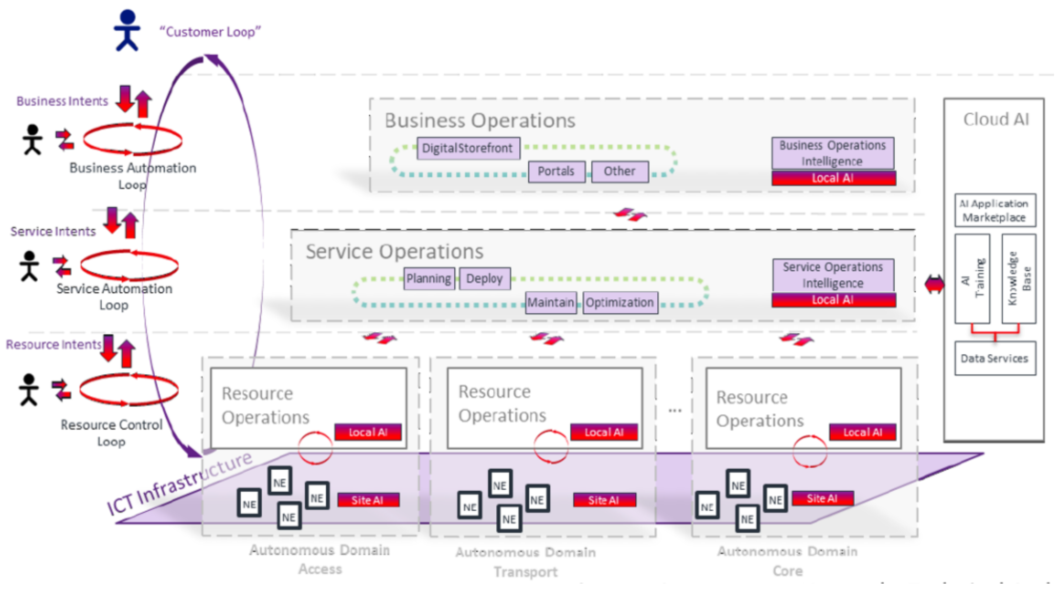

Figure 5 TM Forum Autonomous network architecture target [Boasman-Patel]

Another standardization in which the closed loops appear as an important aspect of the system is in the Zero-touch network and Service Management (ETSI ZSM). The closed control loop is an important part of the architecture and is described in more details in Figure 6. At the left side of the figure is described the architecture proposed by ETSI ZSM and the red arrows are the closed loops. The right side of Figure 6 describes all the steps of the closed-loop, which is divided into five steps:

- **Monitoring:** responsible for detecting anomalies of the entity
- **Orientation:** identify the root cause of the problem
- **Decision:** define the action plans
- **Execution:** fix the problem

The output of the execution step is used as input of the **Managed Entity** that is responsible for managing a specific resource. The Monitoring block collects the output of the managed entity to detect an anomaly, closing the loop [ETSI-ZSM].

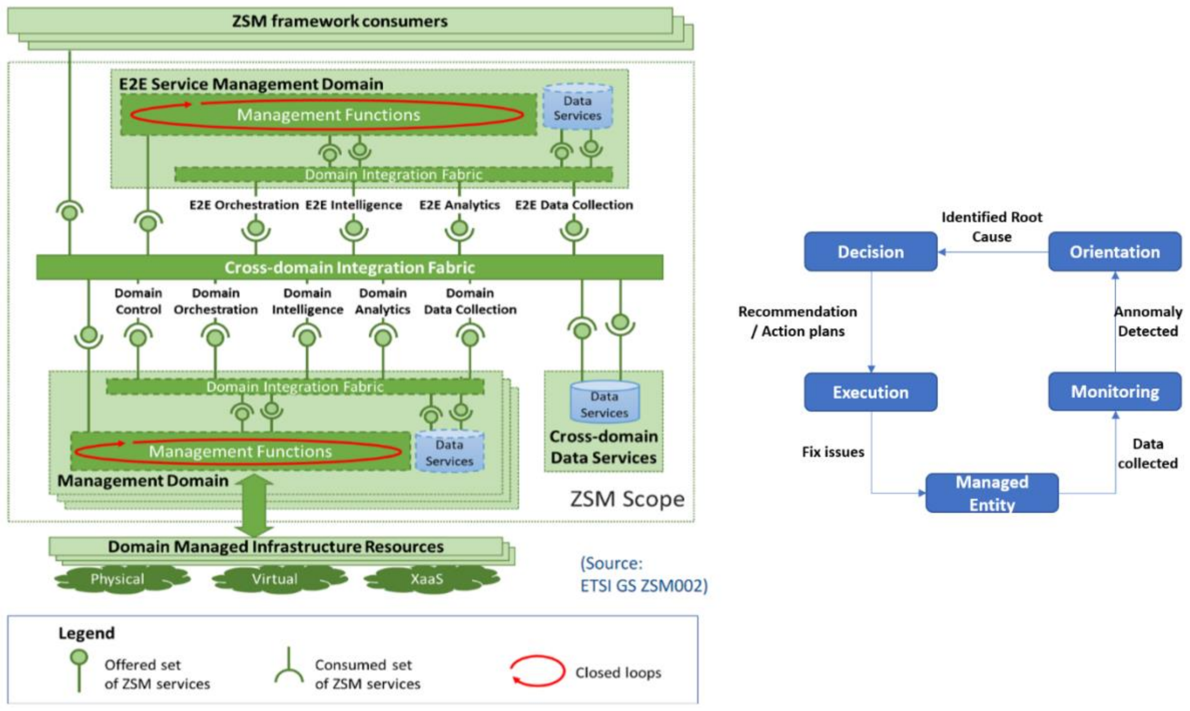

Figure 6 Closed loops in ZSM architecture. Source: ZSM001 specifications [ETSI-ZSM]

ESTI ZSM also presents some active documents specifically for the closed loops in the ZSM network:

- ETSI GS ZSM 009-1: Closed-loop automation: Enablers [GR ZSM 009-1]
- ETSI GS ZSM 009-2: Closed-loop automation: Solutions [GR ZSM 009-2]
- ETSI GR ZSM 009-3: Closed-loop automation: Advanced topics [GR ZSM 009-3]

The third use of closed-loop in standards is in the O-RAN architecture. Figure 7 shows the closed control loop usage in O-RAN. There are three types of closed-loops in O-RAN architecture: Non-real-time control loop, near real-time control loop and real-time control loop. The main difference between each type is the timescale of each one. The non-real-time control loop operates in a timescale of at least 1 second. Examples of application of non-real-time loop include the instantiation and orchestration of network slices. In the Near real-time control, the loop operates in a timescale between 10 milliseconds and 1 second. In these scenarios, an external machine learning-based algorithm is implemented as xAPPs and is responsible for services like inference and classification, and predictions. These xAPPs are deployed at near real-time RIC [Balasubramanian2021].

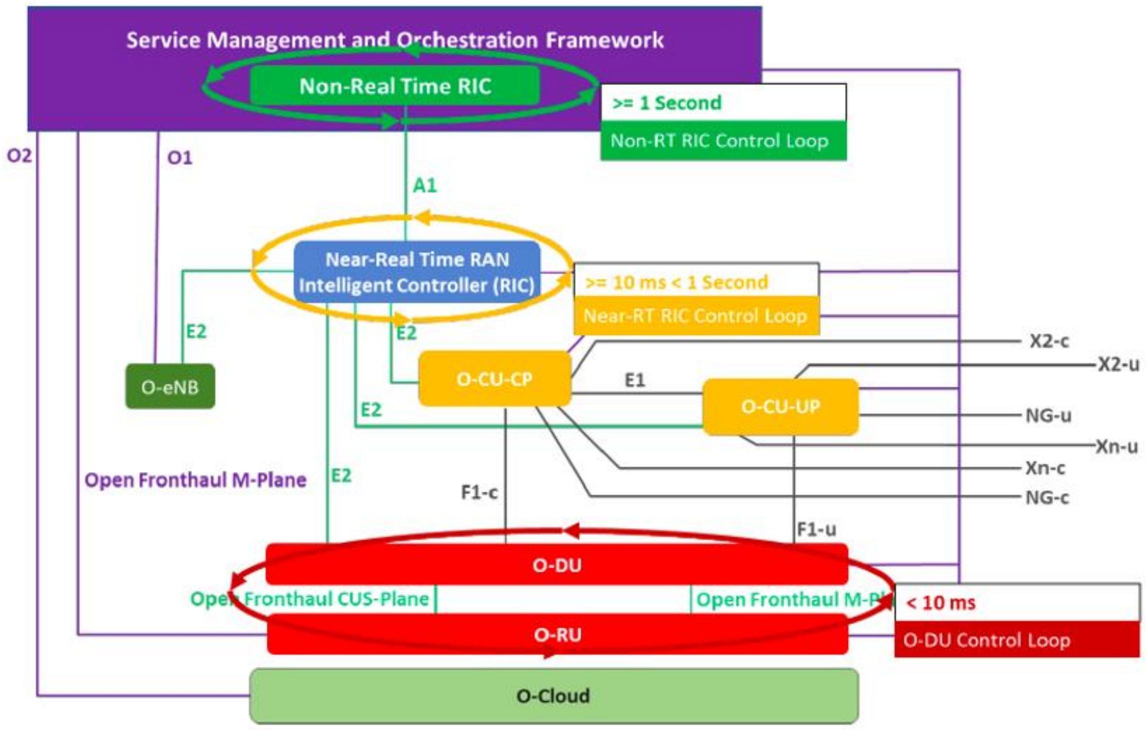

Figure 7 O-ran reference system architecture including the closed-loops (CL in picture) indicated by black arrows. [Balasubramanian2021]

Concluding, this section described some of the closed-loops in standardization such as ETSI-ZSM, TM FORUM, and O-RAN. As AI@EDGE will implement some O-RAN capabilities (e.g, Near-RT RIC and Non-RT RIC), the study and usage of O-RAN closed-loop will be an important aspect of the AI@EDGE project.

For future research, new closed-loops designed specifically for AI@EDGE can be designed in this project. Such closed-loops could be based on the ETSI-ZSM and TM Forum closed-loops definitions, justifying the study of such standards in this report.

2.5 High level reference to closed loops over O-RAN in the AI@EDGE System Design

At the RAN domain, AI@EDGE adopts the O-RAN architecture to provide closed-loop automation. Figure 8 shows the interaction between the main components and interfaces involved in the three closed loops envisioned by O-RAN. While Real-Time (RT) control loops involving E2 nodes are out of the scope of the project (e.g., optimizations on DU radio scheduling, beamforming, etc.), near- and non-RT closed-loops are perfect examples of the application of the different procedures and functional blocks introduced in Section 2.2.

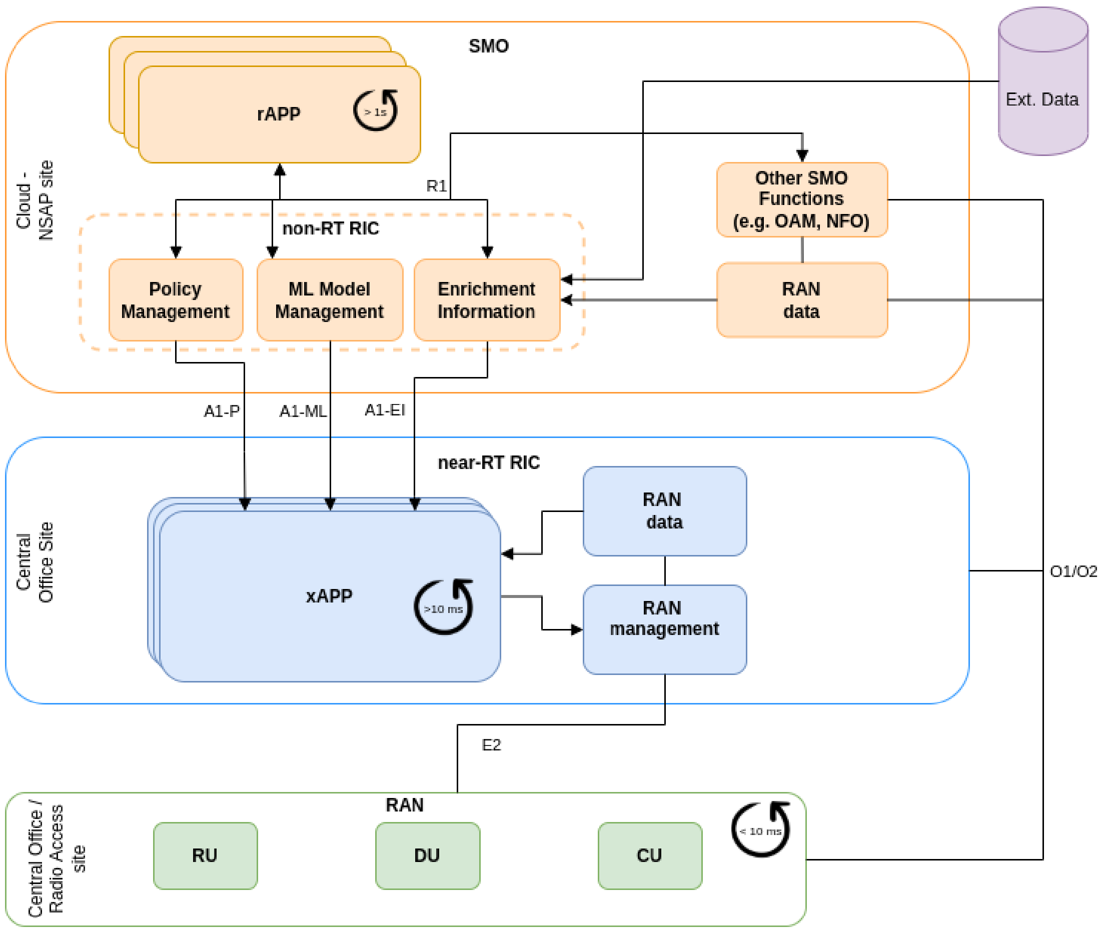

Figure 8 Closed loops in O-RAN.

In the near-RT control-loop, the decisions are taken by the near-RT RIC (RAN Intelligent Controller) and, in particular, by its xAPPs. The xAPPs are applications onboarded on the near-RT RIC, which consist of one or more microservices executing RAN optimizations using E2 interface based on available monitoring data [ORAN1]. This data includes RAN telemetry available at the near-RT RIC level, but could also include aggregated data from external sources such as other RAN domains or even applications, obtained via the Enrichment Information service of the A1 interface (A1-EI) provided by the non-RT RIC. The non-RT RIC can also control xAPP decisions applying policies through the A1-P interface, which are usually related to improving the fulfillment by the RAN of the SLA for all or a class of users in a given area over a period of time [ORAN3]. These policies can be updated during the xAPPs execution by the rAPPs, as part of the non-RT control loop. Similarly, the rAPPs are modular applications present at the SMO/NSAP level. They can execute decisions using the functions and interfaces exposed by the non-RT RIC and the SMO through the R1 interface, like the aforementioned policy creation and update via A1-P interface, but also RAN optimizations or slice management via the O1 interface or NF orchestration via O2 interface [ORAN2]. rAPPs can also access the aggregated data from different monitoring sources present at the SMO, like RAN or application-related telemetry.

Note that both AI-enabled rAPPs and xAPPs could be considered AIFs as defined in the AI@EDGE project. In such a case, they could be managed by the MTO plus the Intelligent Orchestration Controller, implementing some functionalities inherent to ORAN's SMO layer. The different O-RAN interfaces (i.e., O1, O2, A1 and E2) could easily be linked to the AIF's interfaces (i.e. IF1, IF2, IF3 and IF4) introduced in D2.1 [D2.1], further described in Section 4. The O-RAN alliance has started working on the specification of AI/ML workflows and requirements for its architecture [ORAN4]. The on-going work envisions different scenarios (e.g. supervised, unsupervised, reinforcement and federated learning), where the non-RT and near-RT RICs would be capable of acting both as AI/ML training host or AI/ML Model Host/Actor. The A1-ML and O1 interfaces would be used for model management (e.g. model distribution, model update, etc.). As the standardization work is still under development, the AI@EDGE project will not actively work on it, but will monitor its progress to see how it fits in with the project architecture and how it can be utilized.

3. Data Pipelines and Deployment Mechanisms

This section introduces data pipelines architecture considerations in the scope of AI@EDGE and discusses deployment mechanisms needed for secure service offerings and possible solutions. As the amount of data transferred through the network increases, the data pipeline becomes a crucial part of such a complex system, needed to deliver to the automation and learning methods that will be described in Section 4 at the necessary granularity. In the standard approach, when an application requires some data, they collect such data from the data source. However, the same data can be required several times and a redundant stream of this data can be detected (e.g, the same data is transferred twice or more). Since applications in the AI@EDGE architecture can also be deployed in the far edge, using the standard data pipeline approach to deliver the necessary data with the necessary speed and granularity will not be possible. Therefore, a smart data pipeline should be designed. In Section 3.1, the data pipeline solution, its architecture and data-driven operation aspects are presented. Section 3.2 presents some solutions for automated parsing and structure detection of software components logs. This is necessary because the logs can be used to detect problems in the data pipeline that can occurs. Section 3.3 presents the mechanism for secure service isolation, necessary as the security of a data pipeline is important to protect the network. This section is concluded with Section 3.4 which presents the deployment mechanisms.

3.1 Data Pipelines: Architecture and Data Driven Operation Aspects

AI@EDGE will host of multiple AI applications such as end-user AI applications and AI for closed loop control. Each application would require data to process and lead to a decision, and each application need to fit with some specific KPIs. As an example, it is important to meet latency requirements and, if the data is not accessible at the desired time, the application latency will be affected. Also, it is important to avoid the traffic of unnecessary duplicated data in the network. Therefore, a data pipeline system needs to be designed to bring data to those who need it and minimize the traffic of data into the network.

Each AI/ML application that will be developed in the project will require data to process and can use this data to train the model to improve its evaluated metrics (e.g. accuracy, mean Average Precision, QoS, and QoE). However, there are situations where multiple end user applications require data from the same data source. One possible approach is to design a specific data pipeline that will collect, preprocess, and send the preprocessed data to the application that has required it. However, this approach has the drawback that multiple transmissions of the same data can occur, resulting in inefficient usage of the network. One improvement in this design is to collect data once and then preprocess it in different ways according to each application's requirement. This approach is shown in Figure 9. In this figure, it is described as one data pipeline structure called harmonized data ingestion architecture. In this approach, the data pipeline is composed basically of two main structures: The Data ingestion architecture and the Data refinement functionality. The former is responsible for collecting the data and making sure that the data is collected only once. The latter is responsible for preprocessing the data for each different application suite. Each different preprocessed set of data is described by the green box where the collected data is represented by the yellow box. It is important to highlight that one collection of data represented by one yellow square can lead to *n* different preprocessed data sets, one for each application that requires that data in this processed form.

The data pipeline approach of collecting data once from the data source, preprocessing them and sending them to the application that required them can reduce the OPEX and CAPEX if compared to the traditional approach of designing one data pipeline for each application. In the AI@EDGE project, each application can be mapped to one or more AI functions requiring data from one or more data sources. The data pipeline will collect the data and preprocess it before sending it on to the AI function to use it. Figure 9 shows an example of a data pipeline architecture with several AI functions as the application components using the data.

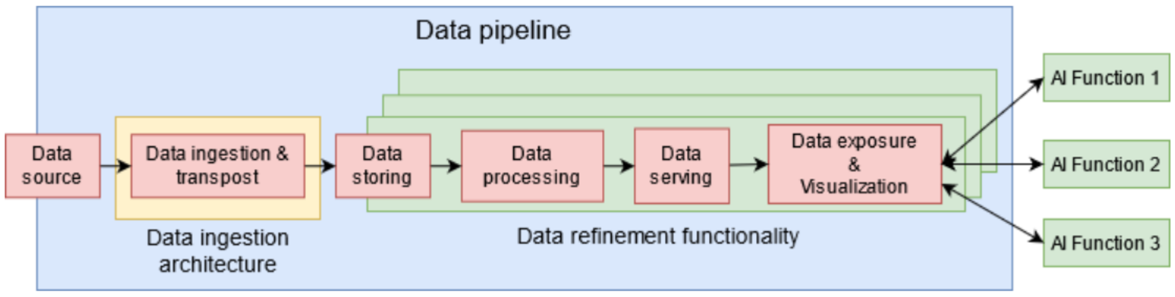
Data pipeline
Data source
Data ingestion & transport
Data storing
Data processing
Data serving
Data exposure & Visualization
AI Function 1
AI Function 2
AI Function 3
Data ingestion architecture
Data refinement functionality

Figure 9 Data pipeline using harmonized data structure in the AI@Edge context.

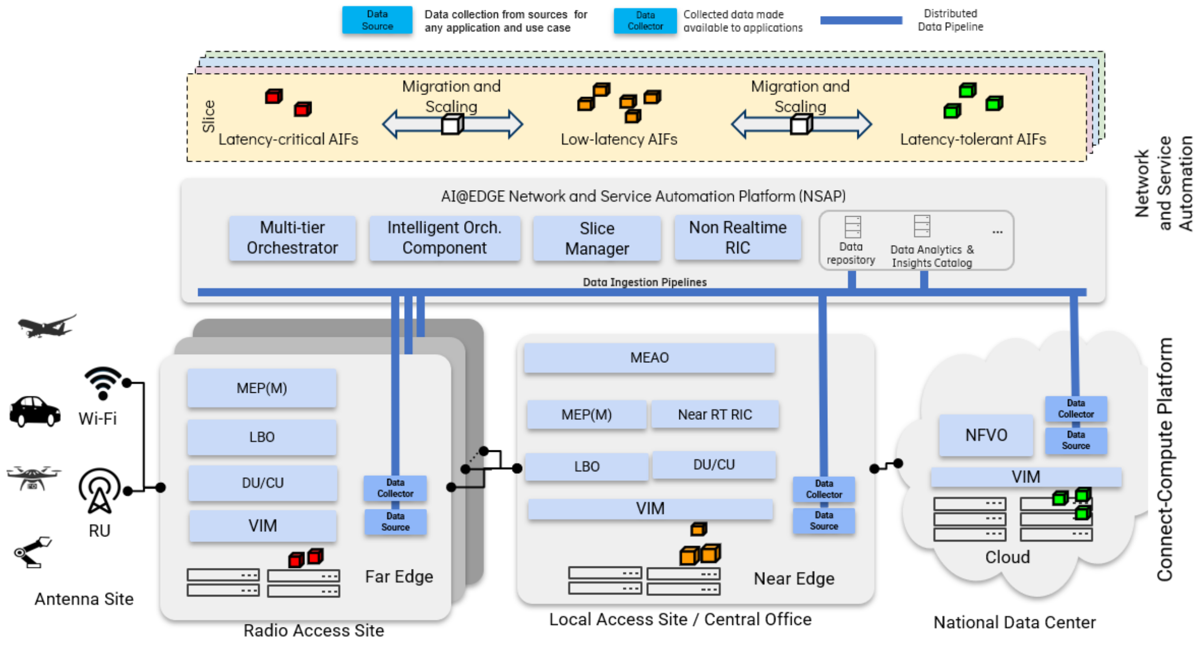

Figure 10 AI@Edge using data ingestion pipeline.

Figure 10 illustrates the AI@Edge conceptual architecture including the proposed data pipeline system. This system is based on the described data pipeline in Figure 9. The blocks named “Data Source” and “Data Collector” are in each domain in the system architecture (e.g, Cloud, Near Edge and Far Edge) and are described in the following:

- **Data source:** is a representation of the element that contains the raw data to be collected and it may be different in the Cloud, Near Edge and Far Edge. ¨
- **Data collector:** is the entity responsible for collecting the data from the data source and may be different depending on the data source. The collected data is going to be inserted in the data ingestion pipeline which is a data bus that connects all the data pipeline elements.

In the NSAP domain, there are two elements of the data pipeline proposed system: Data repository and Data Analytics & Insights Catalog. They are described in the following:

- **Data repository:** is the element that will stores the raw data collected from the Data Source by the Data Collector entity.

- **Data Analytics & Insights Catalog:** are the elements responsible for storing the preprocessing data or any insights about these data that can be useful for the applications.

In this proposed data pipeline system, before collecting the data from the sources, the application would check first if the necessary data has not already been collected and stored in the Data repository (if it is raw data) or in the Data Analytics & Insights Catalog (if the data necessary is a preprocessed data).

3.2 Automated parsing and structure detection of software components logs

Software component logs play an essential role in the management and maintenance of systems. Operating systems and virtualization systems record detailed system runtime information to support comprehensive system understanding and track down problems that may arise. Despite containing rich information, effectively analyzing logs remains a major challenge due to log size, possibly unstructured data nature, and frequency of software updates (thus the frequent update of logging statements).

For AI/ML algorithm processing, and particularly within anomaly detection applications, the log can indeed have the form of gauge values/metrics, cumulative metrics, or natural language strings. To enable log analysis, the first and foremost step is log parsing, in that the raw log records are transformed into a stream of reformatted and structured events. Commonly, a log record includes two parts: header and content. The header is often structured and can be extracted relatively easily. For example, a typical header usually has some basic information such as timestamps, verbosity level (e.g., Error/Information/Bug), and name of events. In contrast, the content has, on the other hand, to be processed. For instance, it needs to be transformed to a suitable scale or data format. Moreover, the content can be hard to structure due to the inclusion of free-text messages written by developers. Generally, the content includes constants and variables. Constants are the fixed text written by the developers and represent a system event, while variables carry dynamic information. The object of log parsing is to separate constants and variables and build the event template from all the constants.

There are many approaches to do log parsing, including, but not limited to:

- Manual with handcrafted regular expressions or grok patterns: Although simplistic, writing special rules for parsing large amounts of logs is time-consuming and error-prone. In particular, these rules will need to be modified manually every time the logging code in the software system is updated.
- Auto learning patterns from log data and generating a recurrent event template:
    - **Frequent Pattern Mining** [Vaa03]: This method considers the event templates as a set of constant tokens that frequently occur in the logs, so frequent pattern mining can be applied to automated log parsing. The log data is crossed to build the frequent words (a word is considered frequent if it appears more than N times in the log, where N is the user-specified value). These frequent words then are used to find out associated frequent words. If a log message contained a pattern of associated frequent words, these words would be regarded as constants and employed to generate event templates. Otherwise, the log message would be put into an outlier group.
    - **Clustering** [Vaa15, Fu09]: The log parsing is defined as a clustering problem of log messages. A log message will be added to an existing cluster if it is successfully matched, otherwise, a new log cluster will be created. Then, the corresponding event template will be extracted from each cluster. [Vaa15] is based on the [Vaa03] but it considers each word without its position in the event log line so it is not sensitive to shifts in word positions and it can able to detect patterns with wildcard tails. [Fu09] adopted a hierarchical clustering algorithm with a customized weighted edit distance metric. It based on the string edit distance (a metric to represent the similarity between

two strings (word sequences) and equals to the number of edit operations to transform from one string to another) but added position-based weighing of words. Additionally, the clusters were further partitioned by heuristic rules.

- **Heuristics** [Mak09, ZAPG]: The log messages have some unique characteristics like message length, token position, so the heuristics-based log parsing methods can apply. For example, in [ZAPG], for all the pairs like "word=value," the "value" is considered as a variable and is replaced with a "$v" symbol.

There are many research studies focused on automated log analysis. Meanwhile, many logs management tools have been developed. For example:

- **Fluentd** [FLU21]: is an open-source data collector, which allows unifying data collection and consumption. It unifies all facets of processing log data: collecting, filtering, buffering, and outputting logs across multiple sources and destinations.
- **GrayLog** [GRAY21]: a centralized log collection and real-time analysis tool.
- **GoAccess** [GOA21]: is an open-source real-time web log analyzer and interactive viewer that runs in a terminal in *nix systems or through your browser. It provides fast and valuable HTTP statistics for system administrators that require a visual server report on the fly.
- **LogParser** [LOGP21]: is a set of open-source toolkits and benchmarks for automated log parsing. It automatically learns event templates from unstructured logs and convert raw log messages into a sequence of structured events.

In data pipelining and parsing, when it comes to using AI/ML algorithms, additional data processing can be required for internal communication within the algorithmic fabric. For instance, models are trained at the device level in federated learning (an approach under consideration in AI@EDGE for distributed anomaly detection). The models are sent to the data sources or devices for training. Then, the models (i.e., models' updates) are sent back to the main server for aggregating. At the end of each round, a consolidated model is sent back to the devices for further refinement. In such a paradigm only the training parameters (i.e., weights and biases) are exchanged between FL entities over communication networks, which requires us to think about applying a different approach for data piping from that of the traditional ML ones.

Due to the large numbers of devices that could participate in a FL training, and complicated ML models, DNN-related for example, the number of training parameters could be up to hundreds of millions (weights and biases) [He16]. This requires incorporating a "stage" in the pipeline to apply techniques for reducing the impact of the model size on the communication means (i.e., model compression [KB], essential update [TZQ], etc.). Also, we might need to incorporate more different stages to the pipeline to handle some of the challenges introduced by FL in terms of security and privacy, for instance. In addition, with multiple FL aggregation servers (in a hierarchical setting), the pipeline should include the operations and communications between the aggregating servers.

We are currently evaluating PySyft framework. It is an open-source library built for Federated Learning and Privacy-Preserving. It allows performing private and secure Deep Learning. PySyft is built as an extension of other DL libraries, such as PyTorch (an open-source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab). We are also using Duet, which is part of the Syft family. It enables creating machine learning models with familiar tools like Jupyter notebooks and the PyTorch API; while allowing training over a remote session, on data you cannot see, anywhere in the world.

3.3 Mechanisms for secure service isolation

The data, which is being processed in the AI@Edge data pipeline, is assumed to be confidential and sensitive. To ensure a secure transmission of the data, the services in the data pipeline have to be secured from external access through encryption and isolation on a system level and the transmission. Therefore, this section deals with mechanisms for secure service isolation.

The first mechanism listed here for secure service isolation is the concept of slice. In 5G networks, isolation can be thought of as a built-in characteristic of a slice, as it is a logical network with certain capabilities that prevents illegal data exchange. To deploy the required functions, the 5G network uses resources from several domains as building elements, such as radio, cloud, nodes, and transport links. 5G networks are planned to use automatic resource allocation algorithms by default. The orchestration and policy mechanisms are anticipated to maintain resource allocation with the desired level of isolation. Taking into account the existing models for describing resources in 5G networks, one can conclude that there is presently no uniform description of isolation capabilities that might be utilized to automate its implementation. Due to the fact that network slicing is a relatively new technology, academics have discovered a variety of implementation and security concerns. Kotulski [KoNoSeTu] goes into great detail on the different difficulties that the technique faces. Security and the implementation of the radio access network (RAN) are the main challenges. A problem that can appear is how to manage the access to the same data to different entities. This scenario is called the multi-tenant hosting and various privacy and security concerns with multi-tenant hosting must be addressed such as Risk Governance, Isolation Failure, Security Incidents, and Data Protection [Yur].

The division of the network into slices in order to deliver numerous separate sets of resources is one of the main principles in 5G. Isolation of resources is critical for offering a network as a service for Service Operators with predetermined Service Level Agreements (SLAs) and quality metrics, such as QoS or QoE. For years, isolation has been a well-known security countermeasure. Isolation strategies are classified based on how deeply they are buried in the informatics infrastructure and which technological tools are employed to offer it [KoZb].

There are two types of systems for which isolation can be defined: individual (server) and networked. Individual isolation techniques can be further categorized as Language-based isolation, Sandbox-based isolation, Virtual Machine based isolation, OS-kernel based isolation and Hardware-based isolation. [ViArNe] Language-based isolation refers to the isolation offered by programming languages, compilers, or code interpreters. Programmers are forced to write code in a way that enforces isolation between implemented programs or isolation of one program from the others by the languages [ScMoHa]. Sandboxing is the process of encapsulating untrusted code in software to prevent it from escaping into fault domains. It is based on the inclusion of additional instructions or checks around each store or jump operation in binaries, compiling the program to only leap into its own code segment and write data to its data region. Virtual Machine (VM) technology is used to provide computer systems with such tools as: emulation, optimization, translation, replication, and isolation. Hosted virtual machines (VMs) provide isolation between virtual operating systems, ensuring that processes running inside a VM do not affect processes running outside of it [Kong]. Isolation based on an operating system kernel takes advantage of the system's most trusted component to offer separation between applications executing on top of it. Its security is determined by the OS kernel's security [ZhYuGl]. Shu [Shu16] defined the hierarchical classification structure for isolation methods, grouping them according to several criteria. First, the authors consider two basic aspects of isolation which are: a mechanism used and a policy. Then, they describe the key properties of methods in each category.

In the network isolation, isolation is provided considering three major methods. Firstly, Tunneling, where the network traffic is isolated from the outside network on its way between one fixed point (port, IP node, host, subnetwork, etc.) and another one. Secondly, Virtual Machine in which the network traffic is separated from the outside environment within a specified virtual subnetwork formed over a physical network. Multi-tenancy via Tunneling or Virtual Networks is the last, in which many tenants' streams of network traffic are isolated from one another via a tunnel or virtual network.

The AI@Edge connect compute platform will isolate its services using virtualization based on VMs and containers. For performance considerations, containers are preferred for near edge and far edge applications and VMs are only employed if required.

3.4 Deployment mechanisms

This section provides an overview of the mechanisms, processes or best practices selected on AI@EDGE for deploying an application through a variety of hosting models across the cloud/edge continuum, each hosting model refers to a specific configuration or environments parameters that defines the computational, storage and networking capacity of the deployment infrastructures. The most common deployment models around cloud function as a virtual computing environment with a different deployment architecture that varies depending on the deployment type selected, such as private, public, hybrid, community or multi-cloud. The deployment types vary depending on who controls the infrastructure, its topology and where it is located. Each company will have to evaluate its own list of unique requirements before it can decide the best deployment model for its business goals.

The selected deployment mechanism needs to ensure interoperability and accessibility through various service or operation technology providers and Hyperscale Cloud Providers. Although there is no industry standard agreed as yet covering all aspects of cloud/edge computing, there are many years of efforts behind different standardisation bodies and open source initiatives on the cloud field. In December 2012, the European Commission (EC) and the European Telecommunications Standards Institute (ETSI) launched the Cloud Standardization Coordination (CSC) initiative with the objective of identifying a detailed map of standards required in areas like interoperability, security, data portability and reversibility. On the community side, The Linux Foundation Edge seeks to facilitate harmonization across Edge projects creating an open source community across IoT, Telecom, Enterprise and Cloud ecosystems. At the same time, the Cloud Native Computing Foundation (CNCF), founded in 2015 as part of the nonprofit Linux Foundation, aims at promoting the evolution of container technology. It servers as a vendor-neutral home for many of the fastest-growing projects providing foundations for the development of cloud computing applications.

Consolidated projects under the umbrella of CNCF include tools and frameworks that have been considered as a part of the tool chain selected to build the AI@EDGE platform, such as Kubernetes (K8s), Helm charts or Prometheus. Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and managing containerised applications. In the AI@EDGE project, K8s performs the role of Virtualized Infrastructure Manager (VIM) responsible for controlling and managing the NFV infrastructure. The HELM charts could be used by AI@EDGE platform as descriptors for managing K8s packages and their operations. The chart is a collection of files that describes a related set of K8s resources in YAML format.

4. Methods for automation and learning in MEC and cloud systems

In Task 3.3, our ambitions are to develop methods to support the automated deployment of AI-enabled applications in cloud and MEC infrastructures. AI-enabled applications can be composed of AIF and non-AI components. So, there is a high coupling with heterogenous services: network services to ensure connectivity, telemetry services to acquire data, analytics to extract knowledge and orchestrators to react from the extracted knowledge, etc.

Distributed infrastructure, as considered in the project, will provide a powerful platform to run AI algorithms at scale by the use of distributed computational resources. Several approaches exist from simple data distribution over a set of workers or more advanced mechanisms to jointly learn and update a model with Federated Learning (FL). These recent approaches differs from the traditional centralized approaches relying on very performant machines in a cloud (and mostly homogeneous). The highly distributed infrastructure we consider in the project would provide multiple advantages: benefit from specialized hardware that might not be located in a data center, reduced latency by analyzing data closer to their sources, reduced privacy risks by distributing the only necessary data to certain location, etc.

However, the counterpart is the heterogeneity of the infrastructure to be handled to deploy our services. In usual datacenters, network and compute resources are quite fixed, which makes easier the allocation of resources. In our case, the environment will be more heterogeneous with different resources to be leveraged, particularly with specialized hardware, and more dynamic with mobile users or devices. The processing and analysis of data might need to be migrated or scaled up, depending on the changing conditions. Even better, our objective is also to forecast the evolution of the system in order to anticipate the changes. We should also ensure the connectivity between the resources. That is why our project considers a connect-compute layer on top of which network and services must be managed or orchestrated in an efficient manner.

Therefore, Section 4 will cover these challenges by providing:

1. Monitoring techniques to get and forecast relevant information to be used for allocating resources for AI-enabled applications and network services;
2. Techniques to improve the performance of ML tasks, notably the learning phase;
3. Methods for allocating resources in order to orchestrate and deploy the services and applications, including AI-enabled applications composed of AIFs.

In this first iteration, the main objective is to give an overview of the proposed techniques, and their positioning regarding the state-of-the-art. However, a complete decoupling of the different techniques mentioned below is not always possible. For example, some methods to allocate resources will be dedicated to federated learning tasks or a monitoring technique will be able to predict indicators or metrics specific to a particular task, such as allocating resources for NFV. Thus, each proposed technique cannot be categorized as one particular type.

However, the proposed techniques can be distinguished whether they aim at directly supporting the deployment of services in the connect-compute platform (our main objective) or they aim at enabling additional services to improve security or performance. In the first case, covered in the first subsection, the objective is to predict resource availability in the platform in order to deploy or migrate services according to them. Techniques will differ based on the type of resources to be monitored and predicted which directly depends on the type of environment the services will be placed in. In the second case, covered in the second subsection, proposed functions could be qualified as advanced techniques to support a higher security (anomaly detection) and/or improve performance of using AI for service placement by leveraging collaborative learning or data augmentation. Several of the proposed techniques are also relying on AI. In these cases, they are considered as AIFs. For example, we can have an AIF for forecasting available computational resources to be used to deploy another set of AIFs. So, when possible and applicable, the proposed techniques are mapped to the initial reference model for AIF of the project reminded in Figure 11 with the following interfaces (more details about the AIF model can be found in D2.1 [D2.1]):

- IF1: (re)configuration
- IF2: model parameters exchange
- IF3: data exchange
- IF4: reconfiguration of other entities

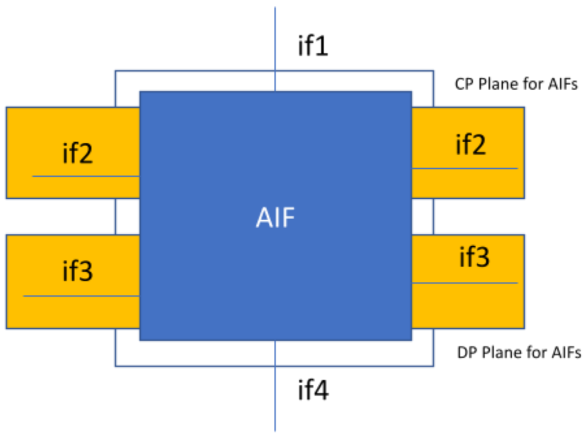

Figure 11 AIF model.

Thanks to this preliminary mapping, missing elements in the model such as interfaces will be identified and can serve as input for WP4.

4.1 Predictive resource monitoring for service placement

4.1.1 Predictive resource and service placement at the edge

This subsection describes the module in charge of the prediction of resources and service placement at the edge using ML techniques based on gathered data from the network infrastructure. The objectives of the module are described, followed by a brief state of the art and a description of the prediction method itself. Finally, the integration within the AI@EDGE architecture is described.

4.1.1.1. Objectives

When deploying resources at the edge of the network, and given the scarce resources that may be available, it is worthy to consider methods assisting service provisioning, resource availability and service placement. Resource provisioning and availability cover diverse aspects that can contribute to optimizing the use of the infrastructure allocated to specific services or network slices, and the configuration of the AIFs or applications running at the edge [Sal21]. AI/ML techniques can be applied to this task by leveraging, for instance, the data available from the radio access network [Cor21] and the monitoring data made available from different hardware and software components [Pol20]. Such resource availability can be of great help for performing smart placement of services in distributed environments [Emu20]. Notice that the complex nature of service placement is highly affected by the continuous changing environment and constraints from users (QoS) and service providers (SLA).

In view of this, the objective of the proposed method is to use telemetry data from the network, QoS and SLA constraints, to build an ML model able to estimate the resource availability and perform a smart service placement at the edge. Specifically, telemetry data is the network information obtained in numerous infrastructure layers, both logical and physical, used as an input for the ML process. Telemetry data could range, for example, from the status of the infrastructure itself in the form of available resources at the edge nodes, to radio access network data available at the edge through MEC services such as Radio Network Information Service (RNIS), and in general as any variable indicator that can be collected from any point in the network [ETS19]. In this regard, it will be analyzed how distributed learning methods can improve placement strategies, by identifying migration operations that keep the QoS constraints across time.

4.1.1.2. *State-of-the-art*

The topic of resource placement and orchestration has attracted considerable attention in the last years. In particular, related research can be looked at from the point of view of the platform components (i.e., cloud-based, edge/cloud-based, and multi-edge, and in the case with the involvement of IoT devices in any of these setups) and from the point of view of the application design (i.e., a monolithic application containing all the required functions, or a multi-element application with necessary or unnecessary communication among the subcomponents). When considering a multi-layer setting, the location of the placement logic becomes a key issue. Another aspect to consider is the temporal aspect of the placement decisions. In this respect, we could find in the literature (i) offline methods, which assume a batch placement of all tasks; (ii) online methods, solving the placement in sequential request arrival; (iii) online methods with migration, which extend the second one incorporating migration tasks for the requests already in place; and (iv) hybrid, which fully re-optimize the placement status periodically.

In either of the cases, a significant part of existing research assumes that services are initially placed in the cloud and that the MEC orchestrator must determine if and how to distribute this load across edge nodes [Mai19] [Zha17]. Alternatively, focusing on multi-edge scenarios, existing research usually focuses on a join optimization of radio association, service placement and resource allocation [Beh19] [Bad20]. Most of the works tackling this problem have as main optimization objective to minimize the latency of service completion [Mou21] or to maximize the number of satisfied requests with respect to delay QoS requirements and resource limits [Mse19]. Alternatively, cost minimization is widely applied to these problems, considering that the total deployment cost involves the wireless communication cost to reach the service and the consumption cost of the function placed to ensure the required QoS [Mah20] [Mai19]. Related to this, some works aim to predict the cost that future service migrations would have, based on users mobility, to estimate the initial placement [Wan17]. When the applications consist of multiple components, namely VMs, containers of functions, the methods adopted in the literature differ with respect to the previously mentioned works. A large part of the authors relies on replicas to increase the reliability and coverage of the service itself, and address latency constrained service access [Zha18]. This is especially important when users show mobility patterns, since taking them as input could help in proactively performing a better estimation of the replica’s locations [Bah20].

According to the location of the entity performing the placement logic, the approach could be centralized or distributed. The vast majority of papers assume the former case, and it is a dedicated orchestrator the one devoted on top to calculate the optimal placement and to enforce the appropriate deployment of service components [Mah20] [Mai19] [Zha17] [Beh19]. By contrast, the authors of [Cas19] and [Asc17] propose distributed algorithms based on voting and election procedures, thus demonstrating that these techniques can perform well in edge-cloud settings for workload management compared with centralized approaches. A similar pattern (centralized vs. Distributed) can be seen in approaches relying on ML tools. In the centralized case, a central entity gathers telemetry information from the underlying MEC platform(s) to decide on the placement. In this case, different types of supervised learning tools, and reinforcement tools can be observed in the literature [Mse19] [Mu21] [Fian20] [Kib21] [Emu21]. The authors of [Mse19] design a DRL agent that has the knowledge on resource availability and decides on resource allocation that maximizes the number of accepted requests, while performing migration processes when the response delay exceeds a threshold. A similar approach is also presented in [Mu21], although seeking to perform the elastic placement of VNFs while minimizing the total energy consumption. However, this paper only considers the requirements of cloud sites. The work in [Fian20] introduces a deep learning pipeline for traffic forecasting based on 3D CNNs, and uses this output to perform VNF scaling and proactive routing per eNB. The authors of [Kib21] take a step forward and propose a similar work to [Asc17] based on RL but consider the placement of stateful VNFs with the main goal of minimizing placement costs. From the centralized point of view, also ensembled VNF deployment strategies leveraging a set of CNNs and ANNs is envisioned in [Emu21], showing performance and accuracy differences w.r.t standalone deep learning models. However, no prior work tackles the problem looking at the telemetry data from orchestrators as well as from the radio access network, while considering SLA and specific requirements of the MEC nodes for the services to be placed, e.g., hardware acceleration capabilities, availability of certain MEC services, etc.

4.1.1.3. Overview of the method

The problem described below is graphically represented in Figure 12. In particular, it can be seen how the potential data that can be used for the training process can be extracted from the virtual infrastructure manager (VIM), from MEC services available, or interchanged between orchestrators in case of distributed learning methods. Regardless of the final nature of the data, the ML/AI approach built must take care of properly managing and aggregating data from different sources, possibly using also different timestamps.

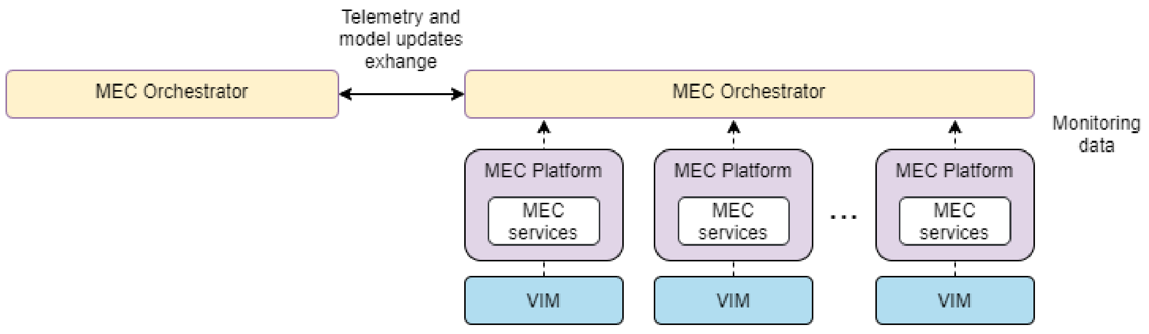

Figure 12 Overview of the predictive service placement approach.

This method could be divided into two approaches. The first approach would be based locally on a single domain (involving a single orchestration domain), and would be in charge of analyzing the monitoring and performance data available, and of forecasting the expected resource availability in terms of CPU and memory at the edge nodes to perform placement decisions on the aforementioned nodes. The second approach would work in a distributed manner involving agents at different orchestration domains and possibility deciding on service migration tasks if required due to a more accurate service placement or due to the degradation of QoS/SLA (if any).

4.1.1.4. Integration in the architecture

In the AI@EDGE architecture, different learning methods can be leveraged to introduce the required technical functionalities to implement the closed-loop operations. In this case, two types of learning models could be envisioned: centralized and distributed (including federated).

In the centralized case, a single AIF would be required per each entity where the model is trained and runs (i.e., this could mean having an AIF per MEC orchestrator). The AIF would be responsible for estimating the infrastructure resource availability (CPU, RAM, etc.) on the underlying MEC platforms (and VIMs) required to fulfill the SLA requested for the placement of a new service based on telemetry data and perform a placement decision. This AIF would have to also take care of the initial alignment of data sources (different timestamps, etc.) and their preprocessing. In this case, a data plane interface

(IF3) would be used for data exchange of the aforementioned metrics since no interaction would be required among AIFs. Moreover, an IF4 interface could be used to reconfigure the placement policies of the corresponding orchestrator.

In the distributed scenario, we could assume that each node with placement capabilities (i.e., a MEC or an NFV orchestrator) can have an AIF in charge of performing the training with local telemetry data from the underlying VIMs. An additional AIF could be located in such nodes for distributed learning purposes to interchange the weights of the model across nodes. This AIF could be optional since the same task could be carried out by a single AIF per node. The local AIFs would consider an IF2 interface (for data acquisition) and an IF3 interface (for weights and parameters aggregation). In the federated learning case, there could be an AIF located on a node with greater computational resources (e.g., on the cloud) whose task would be to compose a global model from the local AIFs, update the corresponding weights and take care of performing a continuous learning process. As in the previous case, the local AIFs will have an IF2 interface and an IF3 interface, which in this case would be used to request parameter aggregation and receive weights updates. The last interface shall also be present in the AIF on the cloud in charge of managing the federated learning operations, which would be used for receiving model weights from local AIFs and update each of them periodically. In both distributed and federated learning scenarios, the AIF would incorporate an IF4 interface to provide the service placement actions. The decision on the specific learning model to be used will be detailed in the next deliverables of this WP. Figure 13 showcases the described AIF as well as the interfaces defined in distributed scenarios.

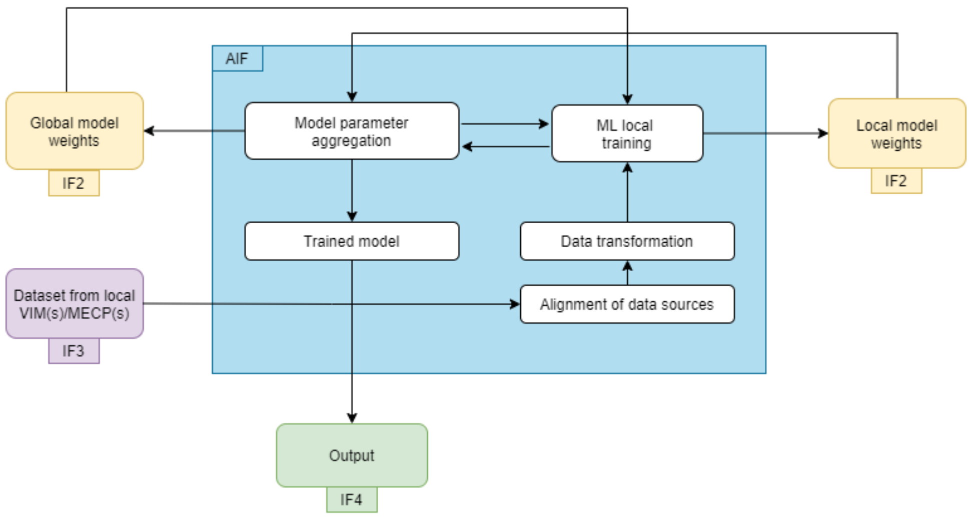

Figure 13 Layout of the AIF designed for service placement estimation operations.

4.1.2 Forecasting of measurable performance KPIs, capturing contextual RAN low-level and network layer data at the edge for user mobility

This module addresses the pivotal role that network function virtualization (NFV) technology plays in the realization of the 5G networks [ETSINFV2014, ETSINFV2017], as it decouples the legacy network functions (NFs) from purpose-built hardware and deploys them as platform-independent virtual network functions (VNFs). NFs in the 5G core service-based architecture (SBA) [3GPP5GS], such as the user plane function (UPF) and session management functions (SMF), are deployed as VNFs. In this way they provide unprecedented management flexibility while curtailing both capital expenditure (CapEx) and operational expenses (OpEx). NFV provides the opportunity to represent 5G network services and applications as a single VNF or multiple VNFs interconnected in a particular order forming service function chains (SFCs) [BEH21]. Multi-access edge computing (MEC) is expected to be widely adopted in the 5G network in order to satisfy the ultra-low latency requirement of certain applications and services while at the same time alleviating the transport network load. In conjunction with NFV, MEC enables both the 5G core VNFs and the application VNFs to be placed at the most appropriate location at the mobile network edge [ETSIMEC].

4.1.2.1. Objectives

In order to fully take advantage of the benefits of delivering services at the Edge, it is necessary to carefully consider the available resources that can be scarce and are shared between various competing services and applications. It is therefore necessary to adopt efficient methods capable of making intelligent use of resources [TANG]. Moreover, in the context of mobile networks, it is also necessary to consider the mobility of the users and their potentially frequent handovers that add dynamicity to the network configuration. In this context, providing service continuity is a challenge that involves VNFs migration and dynamic resource allocation [BEH21].

The objective of the proposed method is to employ the Radio Network Information Service (RNIS) available at the MEC hosts to predict the user location during mobility. These predictions, together with the forecasted performance KPIs based on data captured at RAN and Network Layer at the edge, are meant to be used for decision making regarding VNFs migration or re-instantiating. Measured KPIs could also be given as input to other modules for performing various actions by means of, e.g., SDN/NFV actuators, resources allocation etc. Through analyzing the network and/or service data, intelligence control loops can be triggered.

4.1.2.2. State-of-the-art

Service VNF migration has a great impact on both the QoE perceived by the end-user and the network performance. In this regard, a sizable body of research has been conducted on the problem of service VNF migration.

Xia et al. in [XIA] intend to minimize the VNF migration cost, where the cost is defined as the overall traffic served by the VNF. The problem is tackled using integer linear programming (ILP) techniques. Moreover, they propose a heuristic algorithm to minimize the migration cost and tackle the scalability issue of the proposed ILP method. Another study by Cho et al. [CHO] formulates the problem of VNF migration for latency stringent applications in a highly dynamic environment. They also propose a heuristic algorithm, which triggers VNF migration when SLA violations happen. Carpio et al. in [CARPIO] introduce a linear model to combat the problems of QoS degradation caused by service interruptions and improper load distribution among servers. They study the trade-off between VNF replication and migration of already deployed VNFs to balance the load on them and reduce the number of migrations. The study in [HAWILO] proposes a mixed-integer linear programming model to decide whether to migrate or instantiate the VNFs of the same service in case of failure or resource scaling, having the objective of minimizing service downtime and service latency. Behravesh et al. in [BEH21] considers both SFC placement and VNF migration into account and propose a MILP model and heuristic algorithm to tackle the problem.

Specifically, they propose an objective function to minimize the number of UEs that change their serving node. One way to achieve this goal is to minimize the number of VNF migrations and, upon an urgent need for a VNF migration, decide which VNF to migrate in order to ensure a minimal effect on the UEs served from that VNF.

Performing predictions at the RAN is challenging due to the continuous changes in the physical channel and the availability of different RATs. Recently, employing ML methods to predict the behavior of users has gained tremendous attention. Predicting user behavior is important since these predictions can be employed for taking actions that can result in better network performance and, at the same time, improve the user experience. Behravesh et al. in [BEH20] employ Random Forest and Gradient Boosting Trees to predict the base station association of the users and, based on that, to place the content at proper MEC servers that can lead to the highest satisfaction for the end-users. Amina et al. in [AMINA] propose an AI-empowered method for the migration of stateful VNFs. The objective of the work is to minimize the sum of operations cost and potential loss caused by outages. Takahiro et al., in [HIRAYAMA] propose an encoder-decoder Recurrent Neural Network for the problem of VNF migration to dynamically react to changes that happen in the substrate network.

4.1.2.3. Overview of the method

As mentioned above, the first problem that we are going to tackle is to predict the next base station that a user will be associated with. In this regard, we will employ data-driven Machine Learning (ML) methods to predict base station association of the user based on the radio metrics gathered from the RNIS service, which is available to the applications as a service by the MEC platform.

The prediction results gathered from the ML method will be provided to an Integer Linear Programming (ILP) model designed to embed the SFCs onto the substrate network. The ILP model makes decisions regarding the migration of the services, the state of the users, or the re-instantiation of a new service in the destination MEC host for the user. In this study, we consider different objectives to optimize the network, such as Quality of Service (QoS) and cost metrics.

4.1.2.4. Integration in Architecture

The ML method gets information on the radio metrics gathered from the RNIS service, which is available to the applications as a service by the MEC platform, the output is the prediction on the location of the user (during mobility), as illustrated in Figure 14. the design of the AIF for user mobility prediction consists of ML model that predicts User base station association from low-level RAN data received via IF3 process. The ML predictions on the user base station association are provided to an ILP model via IF2 process, which takes the decision regarding the migration of services or new service instantiation in the predicted MEC host.

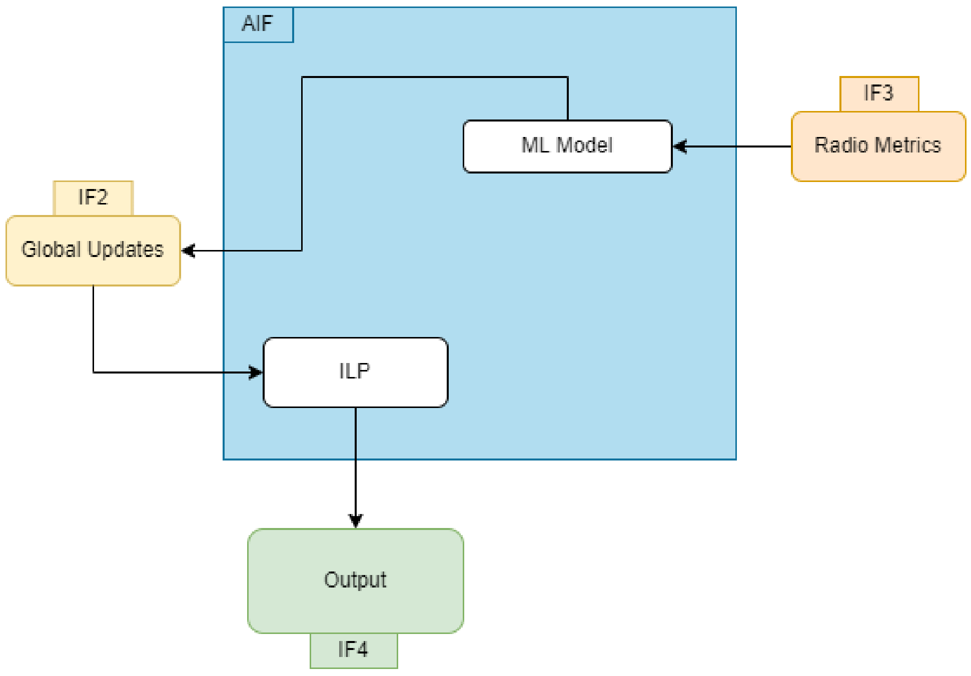

Figure 14 Outlines the AIF interfaces of the Proposed schemes.

4.1.3 *Predicting general and application specific user performance from the mobile edge perspective*

4.1.3.1. Objectives

Resources in a MEC enabled mobile network consist of radio, storage and compute resources. Allocation and management of these resources efficiently in highly dynamic scenarios requires proactive actions based on predicted requirements of users in the network using high granular metrics that capture the network state. Applications hosted at the edge require predictions of user performance so that they can effectively satisfy the demands of the users and provide performance guarantees to them as well as effectively use the shared edge resources they have. All intelligence decisions made at the edge require predictions of user performance to act on. This work will predict performance at the application level e.g. for video streaming applications to predict bitrate of user requests [DIA20][ZHO17], and at the network level, e.g. predicting delay performance at the user using base station metrics [BEN94] [SAT17].

4.1.3.2. State-of-the-art

Performing predictions in radio access networks (RANs)is especially challenging due to the continually changing conditions of the physical channel and the availability of different radio access technologies [KARIM]. Employing ML to predict specific metrics (e.g., channel throughput) for RAN has gained importance in the past years [YUE], [SAMBA][KARIM][RACA2]. Within the context of DASH, previous work employing ML focused mainly on bandwidth estimation at the client, which constitutes an input to most ABR algorithms. To this extent, Raca et al. [RACA3] demonstrate that integrating throughput prediction in the client can increase QoE regardless of the employed ABR algorithm. This idea is further explored by Raca et al. [RACA], where the authors used an RF algorithm at the client to predict the expected average throughput over a time horizon. Moreover, Mao et al. [MAO] developed a reinforcement learning method for directly obtaining the bitrate for the next video chunk. The model employs an Actor-Critic neural network model at the client, whose input includes historical throughput information, buffer state, and next chunk sizes. Liang et al. [LIANG] demonstrate and motivate the benefits of predictive pre-fetching. They consider streaming over a wired network wherein the rate of change of segment bitrate is very low, justifying their assumption that the next bitrate requested can be the same as the previous bitrate requested. Compared to the literature, we consider a more current realistic scenario that specifically addresses the challenges of pre-fetching in a highly dynamic wireless environment by using ML-driven prediction. Our approach performs the predictions at the MEC server, and its main aim is to assist the overall caching process at the core. To this end, it employs an end-to-end ML solution that uses the RAN metrics and past history of segments served to a client to predict the number of segments requested over a time horizon and the mode of these segments' qualities per client.

Applying ML for service assurance in 5G networks is now a thriving research topic. In [MAMUN], anomalous outliers are pre-directed using ML in a testbed network. In [ANGUS], three different ML algorithms were tested to find anomalies at the cell level, that is, cases where an entire cell experiences poor QoE due to a malfunctioning eNodeB. The authors of [SUND] focus on anomaly detection in functional behavior using 5G testbed system logs, and the authors of [MAIM] develop a system to detect cyber threats using a neural network model. The authors in [BOLD] present a proactive approach that predicts future base station alarms. Also, the authors of [YANG] present a distributed ML architecture for wireless network management, allowing tasks such as model training and inference drawing to be performed across the entire network. In these studies, the focus is on predicting anomalies that, over a large scale, affect at least an entire cell or even a wider area over a large timescale. Our work aims to provide one step ahead prediction of performance and tie it to the application-level QoS on UEs in the high-granular time scale of milliseconds. The authors of [NARA] present Lumos5G, ML framework, used for predicting throughput as perceived by applications running on the UEs. Through extensive experiments and statistical analysis, they identify the key UE-side factors that affect 5G performance, as well as quantify to what extent the throughput can be predicted. This work uses data from the UE for performance prediction, while our work extracts base station data for the performance prediction modeling, which is less studied and has more impact on network providers.

4.1.3.3. Overview of method

We are developing an approach to use deep neural networks to predict the video segment bitrate of a DASH video stream through supervised learning from an ns-3 simulator dataset, using RAN and DASH application metrics as features. This prediction of segment bitrate shall then be used to prefetch and cache these segments resulting in reduced access delay for the content and potentially to reduce backhaul bandwidth utilization for shared content.

We shall also investigate the use of deep neural networks for predicting delay violations of time critical traffic in 5G RAN. The approach shall investigate the classification of delay into classes, which can then trigger resource diversion/management actions to mitigate upcoming high delay events that could violate delay constraints.

4.1.3.3.1. The prediction problem formulation

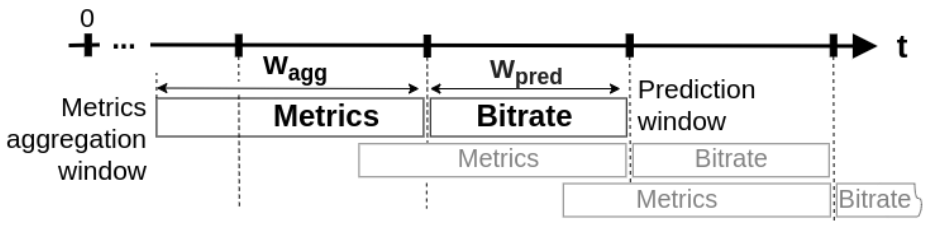

Figure 15 Formulation of the prediction problem

Table 1 Input features to the learning model and the output feature to be predicted.

Input features	Description
Last #seg.	number of requests sent out in the previous window
Last bitrate	bitrate of the previous segment request
Seg. throughput	throughput over the last downloaded segment
Window throughput	throughput over the last 10 s
Buffered Bytes	bytes in the video playback buffer
Buffered #seg.	number of segments in the video playback buffer
DL RSRP	downlink RSRP
DL SINR	downlink SINR
DL MCS	downlink modulation and coding scheme
DL throughput	downlink MAC throughput
Output value	**Description**

Next bitrate	predicted bitrate of the segment to be requested next

During DASH video streaming, segments are requested sequentially by the clients at the bitrate chosen by the ABR. Once a segment is downloaded or fetched, the ABR decides when and at which bitrate the next segment should be requested. Due to limited buffers at the clients, segments are requested as and when they are consumed by the video player, or, in the case of live streaming, as and when they are created. The prefetching of segments is also done on a need basis in sync with the aggregated clients' requests. We aggregate predicted bitrates of requests of clients over prediction windows, W_{pred}, and prefetch them periodically. The input features are created from metrics aggregated over an aggregation window, W_{agg}. Figure 15 shows the time line of how metrics are aggregated and predictions are made.

The metrics used as the input features and the output predicted metric (output feature) are listed and described in Table 1. The input features can be categorized as low-level RAN metrics obtained using the RNIS function from each radio base station, and high-level metrics obtained from the DASH application. Note that it also contains the last segment history of the bitrate which is the previous requested bitrate seen in W_{agg}. To summarize, the prediction problem formulation is the prediction of the *next bitrate* for each client in W_{pred} using the features in W_{agg} as input.

During inference, the prediction algorithm is run periodically to predict the requests expected in the next W_{pred} window. The overall complexity of the prediction task scales linearly with the number of clients in the network. However, since the prediction task for each client can be run independently, it can be parallelized to run over distributed compute resources.

4.1.3.3.2. Data preprocessing

All the metrics we consider as input are time series quantities observed with different periodicities. Some are sample metrics (e.g. RSRP), wherein we have a measured value for every observation, while others like MAC throughput can only be measured over windows. To reconcile these differences in how metrics are observed, we aggregate them over time windows (W_{agg}, metric's aggregation window) to generate a structured tabular dataset.

Since the data is a time series, there is a time correlation between samples of input metrics. Since the prediction quantity is the bitrate of a segment of 8 seconds video, small time correlations do not impact the output much and can be averaged over as we have done using W_{agg}. The choice of W_{agg} size should be informed by the rate of change of state in the radio network as well as the segment duration (which influences the rate of segment request) of the streaming video. The trade-off here is between too little information in small windows and too stale information in large windows. It is reasonable to pick window sizes that are roughly in the range of the size of segments since this is the time frame over which network variation has an impact on segment bitrates.

Table 2 Prediction accuracy using RF over varying aggregation and prediction window sizes.

	Aggregation window size (s)						
		4	**8**	**12**	**16**	**20**	**24**
Prediction window size (s)	**2**	0.859	0.883	0.901	0.903	0.897	0.891
	4	0.843	0.874	0.890	0.892	0.889	0.882
	6	0.800	0.855	0.876	0.887	0.885	0.878

When selecting the prediction window size, W_{pred}, larger windows mean that there are potentially multiple segment requests sent out in the window resulting in additional complexity to the problem, requiring estimating the number of segments. A two-step approach would be required, with separate predictions for the number of segments requested and the bitrate of each of those segments. In our previous work, we considered such a formulation [BEH20]. W_{pred} also increases the waiting time from when a request was sent and when the next one is predicted and prefetched. However, if it is made too small, then predictions will be made very frequently, resulting in a high computation cost to run the prediction and prefetching algorithms. This is evaluated in Table 2. Note that several of these W_{pred} windows will be empty since we choose them to be smaller than the typical time between segment requests. These empty windows just result in empty rows in our structured dataset and are ignored during learning. The results of our empirical evaluation of these window sizes are presented in Table 2. We compared the sizes using the prediction accuracy obtained using a random forest model and chose W_{agg} to be 12 seconds and W_{pred} to be 2 seconds. We obtained a similar order of results for all models we explored and have hence presented the results from one model.

4.1.3.3.3. *Proposed solution using ML*

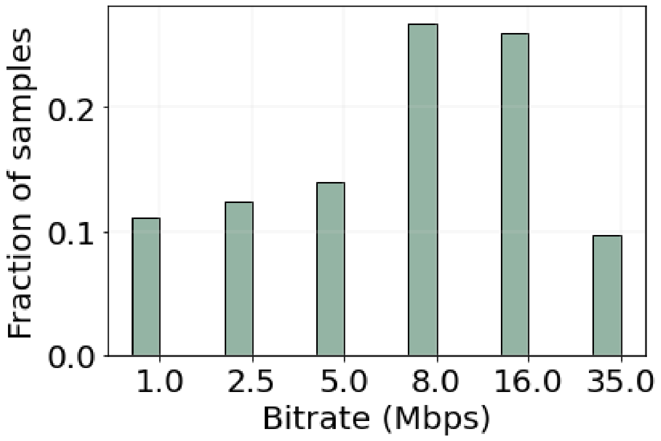

Figure 16 Histogram of request bitrate classes.

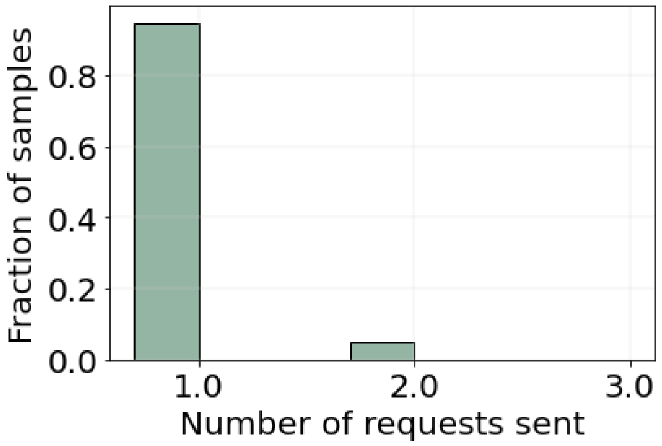

Figure 17 Number of requests in a 2 second prediction window.

The problem of predicting the next request bitrate is modelled as a supervised learning multi-class classification problem. Figure 16 and Figure 17 show the bitrate class distribution and the distribution of number of requests observed in 2 second windows in our dataset. The class imbalance is not high and more than 95% of the non-empty windows have only one request. This indicates that only in 5% of the cases it will be unable to cater to the prefetching requirements in window with our formulation. The set of available bitrates for each segment and hence the classes for classification are 1\2.5\5\8\16\35

Mbps (higher rates correspond to HD, 2K, and 4K video qualities). We implement four models for the proposed prediction task, two decision tree-based models, Random Forests (RF) and Gradient Boosted trees (XGB), and 2 neural network models, multi-layer perceptron (MLP), and Long-short term memory (LSTM).

RF is a method used in this domain in previous literature [RACA] [YUE], with good results. We explore another decision tree based approach, gradient boosted trees [CHEN2016] due to their consistent high performance in the recent past over structured tabular data [CAR2006].

Most recent results have shown that gradient boosted trees typically either outperform or are on-par with neural network approaches when it comes to structured tabular data. We observed that most recent results in our prior-art study were tree based approaches. Decision tree based approaches are faster to train and have fewer hyper-parameters to tune. They also have the advantage of being explainable through feature importance. Deep neural network models are very data hungry, and can outperform other approaches when there is sufficient data, decision tree-based ensemble methods typically win for smaller (< 100K samples) datasets. However, while expensive without much benefit for small datasets, deep neural network models can be pre-trained using, e.g., simulation data and then through transfer learning tuned for the varied scenarios where they are deployed. These advantages of the neural network class of models motivated us to explore their efficacy and compare them against decision tree based ensemble methods (RF and XGB) for our prediction task. The data we have is time series and LSTM is an approach that is designed to natively handle this through its recurrent network architecture. LSTM models, however, have a large number of parameters, and require even larger datasets to learn from.

The RF and XGB models use 350 estimators, max depth of 25 and min samples per leaf of two. The MLP uses a two hidden-layer architecture (30, 15) and LSTM uses a two layer (30, 30) architecture. We use the ReLU activation function and categorical cross-entropy loss function for training.

The baseline we use to evaluate our models is a persistent prediction baseline, where the next bitrate is predicted to be the same as what the previous was. Such a prediction for time series data could be useful for slow changing networks. However, in real networks under dynamic conditions the bitrate is frequently changing, which makes using persistent prediction inapplicable.

4.1.3.3.4. Experiment setup and Data collection

An ns-3 simulated network is used to generate data to train and evaluate the prediction algorithms and to evaluate the prefetching algorithms. The evaluation of prefetching is done offline by using as input to the algorithm the output from the predictor, and the ground-truth from the data. Two separate datasets have been generated from a simulated urban mobile network deployment scenario. We use the ns-3 DASH module implemented by Vergados et al. [DJGIT] to simulate the DASH client-server interaction for video segment requests and response. This implementation contains several adaptive bitrate algorithms at the DASH-client. We chose the FDASH [VERGAD16] algorithm as it has been shown by these authors to provide higher video rates, reduce buffer underflows, and prevent unnecessary video resolution changes compared to the state-of-the-art. It is important to emphasize here that our approach can handle any ABR at the client as long as it has trained on that data.

The simulation setup consists of 12 base stations (four cell sites with three gNBs each), and uses carrier aggregation with three component carriers of 20 MHz to provide a maximum downlink bandwidth of 60 MHz, which can reach an aggregated downlink bitrate of up to 225 Mbps. A remote video server hosts the DASH server. A variable number of users (between 27 and 68) move between these gNBs with velocities ranging between 1.4–5.0 m/s (walking/cycling speeds). Each user in the network is watching one of the 10 videos that are currently being streamed. The user's video stream play times are within 10 seconds of each other, representing realistic behavior of live streams that are viewed within small delays of the actual event. The videos are streamed at 50 frames per second and each segment contains 8~seconds worth of video. The choice of segment duration and frame rate are motivated by typical settings for the chosen use-case.

Several hundreds of thousands of seconds of video playback data was generated from clients streaming videos. After processing the data into aggregation and prediction windows we had around 80,000 samples for the ML models to train on, and 20,000 samples to test on. After training and testing the ML models, the entire predictive prefetching process was evaluated using the around 17,000 samples of evaluation data.

4.1.3.3.5. *Exploratory data analysis*

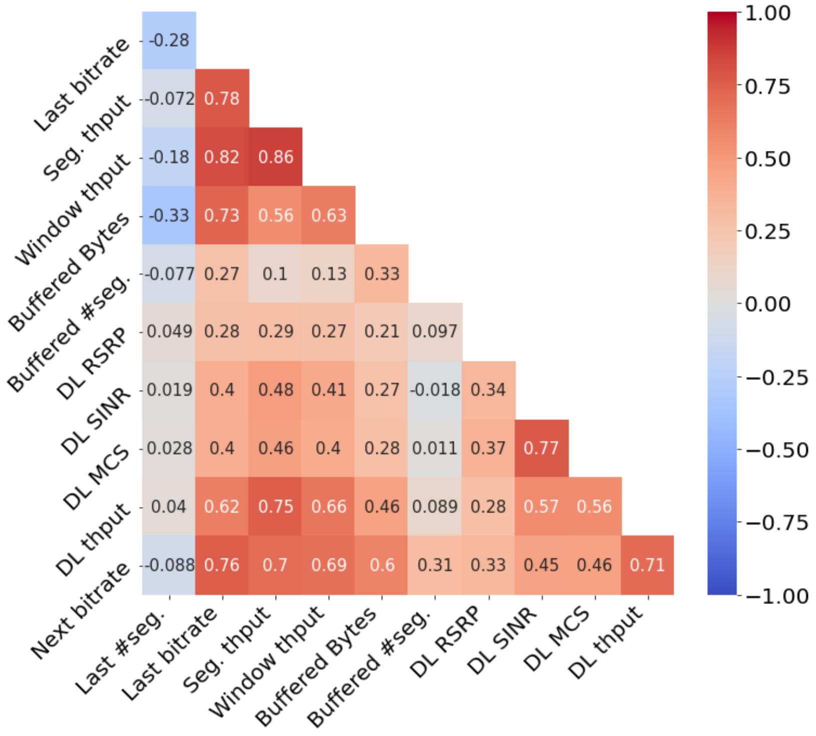

Figure 18 Correlation coefficient heatmap between input and output features.

Before we begin training prediction models with the data, it is essential to characterize the data and understand the properties of the features in them. To understand the relationship between input metrics, as well as between the input and output metrics, we plot a heatmap of the Kendall correlation coefficients between them. The Kendall score captures the strength of the monotonic relationship between variables. Since several observed metrics (e.g. bitrate and MCS) are ordinal values we use a rank based metric, Kendall score. Kendall is also known to be better than Spearman at capturing correlations in data that have rank clashes. In Figure 18, the bottom row (next bitrate) represents the output metric, and the rest the input features. We see several strong correlations between the input and output features, indicating the predictive power of the input metrics. We also see strong correlations between input features themselves, indicating some redundancy in the features.

4.1.3.3.6. *Model Evaluation*

The features available in the dataset consist of DASH metrics and RAN metrics observed at the mobile edge. The RAN metrics can be made available to network functions at the edge through the RNIS function. The DASH metrics can be made available through a proxy at the edge between the DASH client and the DASH server. We study the importance of each set of features by training the models on three different sets containing i) DASH and RAN metrics, ii) only DASH metrics, iii) when we have both DASH and RAN metrics but do not have historical information of last bitrate. Feature set i) includes all features observable, feature set ii) removes RAN features to see how much the additional gNB level observability adds to the prediction, and the feature set iii) is used to assess how much of the predictability power of the models come from just the last sample of bitrate.

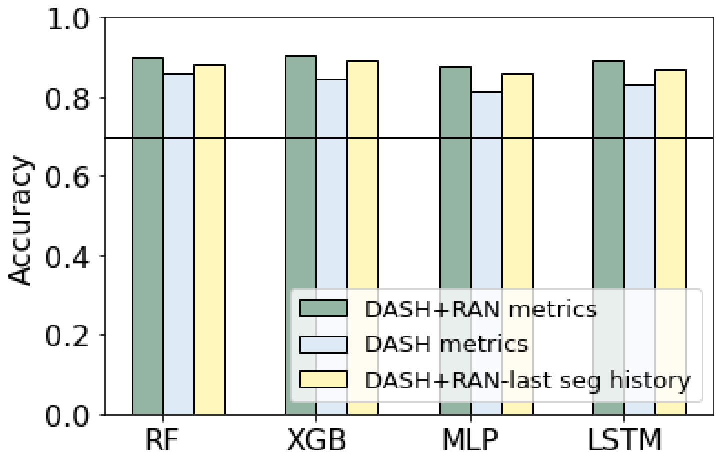

Figure 19 Comparisons of model performance for bitrate prediction tasks.

Figure 19 shows the results of evaluation over RF, XGB, MLP, and LSTM models for the three different feature sets. We use prediction accuracy as the metric for comparison between models and the baseline. Prediction accuracy was chosen since the cost of misclassification between classes is the same in the considered scenario. We see that all models have similar performance with a difference of at most 3% in prediction accuracy. Bitrate prediction using these models increases the prediction accuracy by around 20% from the baseline shown as a black.

Since the difference between accuracy with DASH+RAN features and only DASH features are small, it indicates that the contribution of the RAN features over the DASH features is small. Comparison between DASH+RAN and DASH+RAN-last segment history, also indicates that there is redundancy in the features that can contribute to prediction, even when the last segment value for bitrate is not available. This is important because it indicates that the model can learn to predict the bitrate in a non-trivial way using the network and client state, even when the new bitrate is not correlated with the previous bitrate. This improvement over the baseline is expected to be even higher in more dynamic networks, where the rate of change of network state is even higher resulting in a lower correlation between requested bitrates.

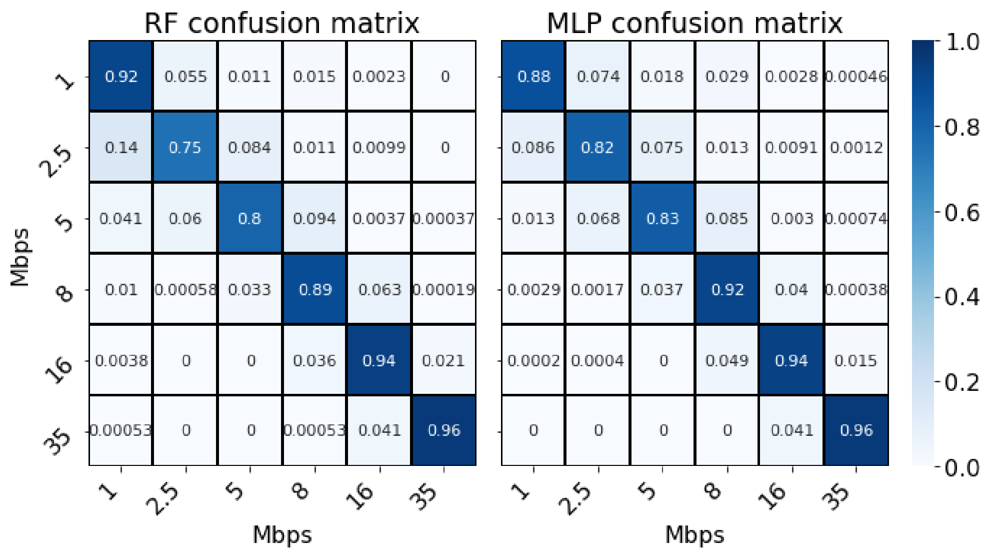

Figure 20 Ground truth normalized confusion matrix for RF and MLP predictions.

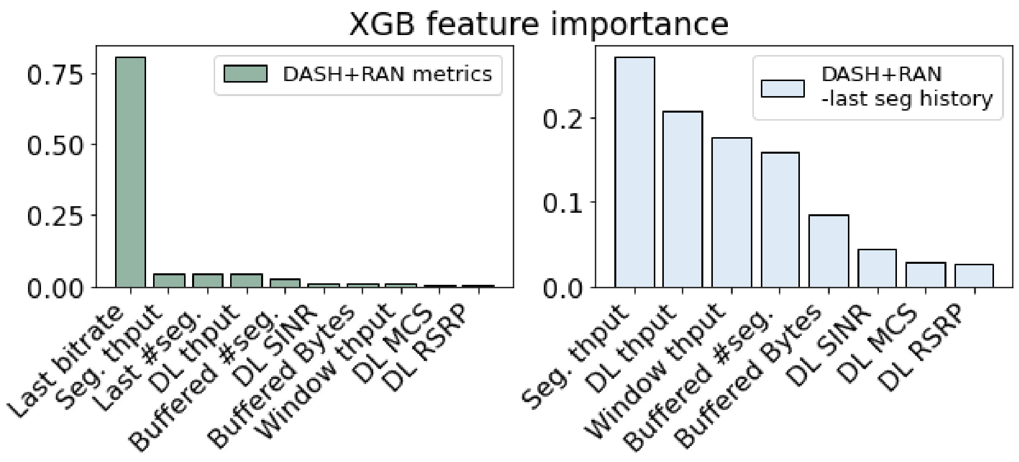

Figure 21 Feature importance for different sets of metrics.

Figure 20 plots the normalized confusion matrix for RF and MLP. The maximum difference between the true positives between bitrate classes is at most 21% for MLP and 13\% for RF. The confusion matrix shows that the prediction accuracy of the models is not swayed by very high and imbalanced true positives of select classes.

One of the advantages of RF or XGB is their inherent model interpretability through feature importance. Figure 21 shows the feature importance graphs for XGB for DASH+RAN, and the DASH+RAN-last segment history feature sets. We omit the graph for RF since the order of features remains the same and XGB feature importance shows the additional contribution of each feature over the previous, as opposed to RF, where multiple redundant features are shown to have the same importance.

When *Last bitrate* is available, it seems to be the most important feature. This is indicative of how the baseline of next is the same of previous still gives us a close to 70% accuracy. However, when that history is removed, we still get similar accuracy but the contribution of feature importance is distributed over all other existing features. As mentioned before, this reinforces that even when the rate of change of bitrate is high, the model can predict without the assistance of *Last bitrate*.

4.1.3.3.7. Summary

We have seen that all four models we evaluated have similar performance, with all of them showing a prediction accuracy increase of around 20% over the baseline. Through evaluation over different feature sets we learnt that the models are able to learn changes in the state, and do not need the history of number of segments and bitrate from the last window. The decision tree-based models, RF and XGB, have the advantage of interpretability, while the neural network models have the advantage of transfer learning over varied deployments. We see that using LSTM did not offer any advantages in this scenario over a simpler MLP neural network. This could be because of the much larger number of trainable parameters in an LSTM model, which requires a larger dataset to train on. While there is no clear winning model to use from just the metric of prediction accuracy, the advantages of using neural network models are far reaching to accumulate the capacity to represent and learn and then generalize these learnt structures through transfer learning.

4.1.3.4. Integration in the architecture

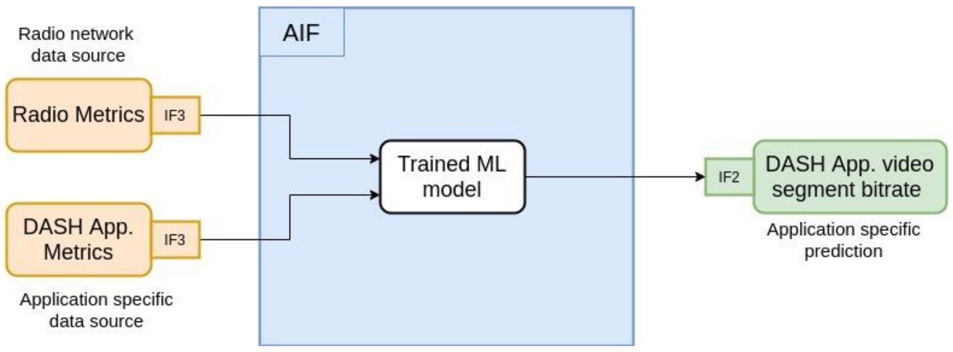
Radio network data source
AIF
Radio Metrics
IF3
Trained ML model
IF2
DASH App. video segment bitrate
Application specific prediction
DASH App. Metrics
IF3
Application specific data source

Figure 22 Outline of the AIF for predicting general and application specific user performance.

To integrate the method into the AI@EDGE architecture, we map the inference model into an AIF. Figure illustrates the design of the AIF for predicting general and application specific user performance. The AIF mainly consists of three components, which are the inference model and the data exchange interface, and the inference interface. The inference model is the core of the AFI. The model could be trained based on four potential architectures: RF, XGB, MLP neural network, and LSTM, which are all better than baseline. As for the data exchange interface, the IF3 processes two categories of data: low-level RAN data and high-level DASH application data, the details are shown in Table 1. Last but not the least, the IF4 outputs the predicted result.

4.1.4 Optimising resource scheduling with machine learning

The vast deployment of connected devices and IoT networks, both in industrial and consumer settings, requires fast, adaptable and efficient management policies to handle the resources among these heterogeneous and complex networks. Managing those resources implies solving optimization problems that are combinatorial in nature. Traditional methods for finding feasible solutions to such problems rely on human-crafted heuristics, which are often one-sided and sub-optimal, leading to waste of unnecessary resources. Moreover, artificial intelligence promises to develop more holistic management policies on the basis of learning agents and data-driven methods, that on the basis of stochastic processes are able to adapt to larger problem instances. Traditional machine learning methods, and the more recently developed field of deep learning, were geared to process fixed input/output structures (such as images, tabular data, and text) and remain unfit to process graph-structured data. Novel machine learning methods specifically developed towards processing graph-structure data can handle varying size of input/output sequences and enable devising more holistic policies to manage such resources [Vesselinova'20]. However, the applicability to such methods to managing heterogeneous network resources remains vastly unstudied.

4.1.4.1. Objectives

Networked systems are per definition graph structured. Previous machine learning, and more specifically, deep learning-based methods to manage resources among devices in the network are only applicable under certain limitations, such as fixing the network size (in terms of number of devices), or completely ignoring the underlying topological structure of the network in the learning process. The main goal of the present subsection is to evaluate the applicability of devising such graph neural network (GNN)-based models towards managing complex network resources. In particular, we present a suitable GNN model to serve as basis for devising such network management policies under the supervised learning setting.

Key evaluation aspects are the generalization to larger problem instances (in terms of the network size) and the inference time from the GNN based systems according to the respective network service requirements. More specifically, we aim at evaluating if a GNN model trained on small scale problem instances, for which the optimal solution can be computed using a constraint optimizer, can scale to much larger and complex problem instances on the basis of supervised learning. We do this by tackling a challenging problem proven to be NP-hard: the interrogation of battery-free sensor tags that employ backscatter communication in Internet of Things (IoT) networks [Perez-Penichet'20]. This problem was selected due to: i) the dynamic nature of the problem, in which any change in the number of IoT devices or sensor tags present in the network yields a completely different problem instance (for which a new solution must be computed); ii) the fact that optimal interrogation schedules require exploiting the graph structure of the IoT network to compute the solution; iii) the potential of recent backscatter communication devices to become one of the dominant sensing techniques in upcoming decades; and iv) its relation to use-case 2 of the project ("Secure and resilient orchestration of large (I)IoT networks").

Finally, the results of this subsection in terms of the learning models, will serve as basis to apply learning-based methods to other combinatorial optimization problems in the networking domain and under the scope of the AI@EDGE project.

4.1.4.2. State-of-the-art

4.1.4.2.1. Machine Learning for Combinatorial Optimization

Machine learning (ML) applied to solving traditional COPs has seen a surge in the past years [Vinyals'15, Dai'17, Li'18]. In contrast to traditional heuristics, an ML system can find hidden patterns in the data through supervision or self-interaction considering multiple objectives, which allows it to better direct the search when solving the COP [Vesselinova'20, Bengio'21]. Moreover, recent advances in Graph Representation Learning (GRL) and graph neural networks (GNNs) [Scarselli'2009, Gilmer'17, Kipf'17, Velickovik'18] allow to leverage the advantages of deep learning for processing graph-structured data. In contrast to other ML approaches, GRL can exploit the structural properties of the topology and can handle varying input and output sequence lengths. ML methods have been applied COPs over graphs in the past years [Veselinova'20], for both reinforcement [Dai'17, Manchanda'20], and supervised learning [Vinyals'15, Li'18]. Vinyals et al. [Vinyals'15] implement an attention-based sequence-to-sequence model that learns from optimal solutions and solves the traveling salesman problem. Likewise, Li et al. [Li'18] employ GNNs [Kipf'17, Defferrard'16] to solve three traditional COPs with a supervised approach.

4.1.4.2.2. *IoT Backscatter Sensor Networks*

Internet of Things (IoT) has positioned itself as one of the most disruptive technologies by enabling industrial and consumer end-devices to communicate over internetworking. Sensor networks have gained increased attention in recent years partly due to the introduction of low-energy bidirectional communication capabilities such as low-power radios and backscatter communication [Liu'13, Kellog'16, Perez-Penichet'16]. Particularly, battery-free sensor tags are components that leverage backscatter communication to transmit sensor values to their hosting IoT device when assisted by an external unmodulated carrier (a wireless radio signal) [BEN94], [Hoch97]. Combining the low-power sensing of the sensor tags with the IoT devices' internetworking has the potential to unveil novel application areas with high societal, environmental, and industrial relevance — e.g., more accurate air quality, temperature and humidity monitoring, smarter logistics and warehouse management systems, and agricultural smart irrigation management systems.

Backscatter communications enable new wireless devices that harvest energy from their environment to operate without batteries [Liu'13, Wang'17, Hessar'19]. Recent advances have demonstrated how battery-free backscatter devices – tags for short – can seamlessly interoperate with unmodified commodity wireless devices [Kellog'16, Perez-Penichet'16]. For battery-free tags to send and receive information from an IoT node, they require an external RF signal---an unmodulated carrier---provided by a second IoT node. Every tag is associated with (or hosted by) one IoT device responsible for collecting its sensor readings. Every IoT node in the network may host zero or more tags. The IoT devices in the network are equipped with radio transceivers that support standard physical layer protocols such as IEEE~802.15.4 or Bluetooth. They are able to provide an unmodulated carrier (by using their radio test mode [Perez-Penichet'16]) and employ a time-slotted medium access mechanism. The duration of a timeslot is sufficient for an IoT device to interrogate one tag by transmitting a request to the desired tag and receiving the response a short interval after while a second IoT node provides an unmodulated carrier, as demonstrated by Pérez-Penichet et al. [Perez-Penichet'20]. Finally, at least one of the IoT devices is connected to a cloud or edge server where the interrogation schedule can be computed.

4.1.4.2.3. *Scheduling Battery-free Tags in Sensor Networks*

Generating a feasible schedule for a given sensor network topology is an instance of a combinatorial optimization problem (COP). Computing an optimal solution using a constraint optimization solver is possible only for small problem instances (low number of IoT nodes and/or sensor tags) but takes a prohibitively long time for practical network deployments. Moreover, current heuristics are able to compute interrogation schedules in the order of seconds, even for large problem instances, but suffer from sub-optimal behavior.´Perez-Penichet et al. demonstrate TagAlong, a complete system with a polynomial-time heuristic to compute interrogation schedules for backscatter devices [Perez-Penichet'20]. TagAlong exploits knowledge of the structural properties of the wireless network for fast scheduling. However, TagAlong's carefully designed algorithm produces wasteful suboptimal schedules, especially as the network topology size increases (in terms of number of IoT devices and number of tags). Van Huynh et al. [Huyn'18] employ numerical analysis to optimize RF energy harvesting tags. Carrier scheduling resembles the Reader Collision Problem in RFID systems [Yang'11, Yue'12] in that both need to avoid carrier collisions. These works focus on the monostatic backscatter configuration (where the carrier generator and receiver are co-located). The bi-static configuration leads to a different problem, and our focus is on resource optimization rather than mere collision avoidance.

4.1.4.3. *Method Overview*

The network management problem selected is the interrogation of battery-free sensor tags in large-scale IoT networks. The battery-free sensor tags require a schedule to communicate with their hosting IoT node through backscatter communication.

4.1.4.3.1. *Interrogation Schedule Constraints*

The properties of backscatter communication impose physical constraints on the way tags are interrogated. First, within a timeslot, a tag can be interrogated by its host if and only if exactly one of its neighboring IoT devices provides an unmodulated carrier. Multiple incoming unmodulated carriers may cause disrupt the tag-to-host communication [Perez-Penichet'20]. Similarly, the limited communication range of the battery-free tags forces them to be interrogated exclusively by their hosts, which must be located nearby. Note that timeslots are permutation invariant, which implies that one can arbitrarily shuffle the order in which the timeslots appear in the schedule. Because of this and other symmetries, the scheduling problem typically has multiple optimal solutions. For more details, please refer to [Perez-Penichet'20].

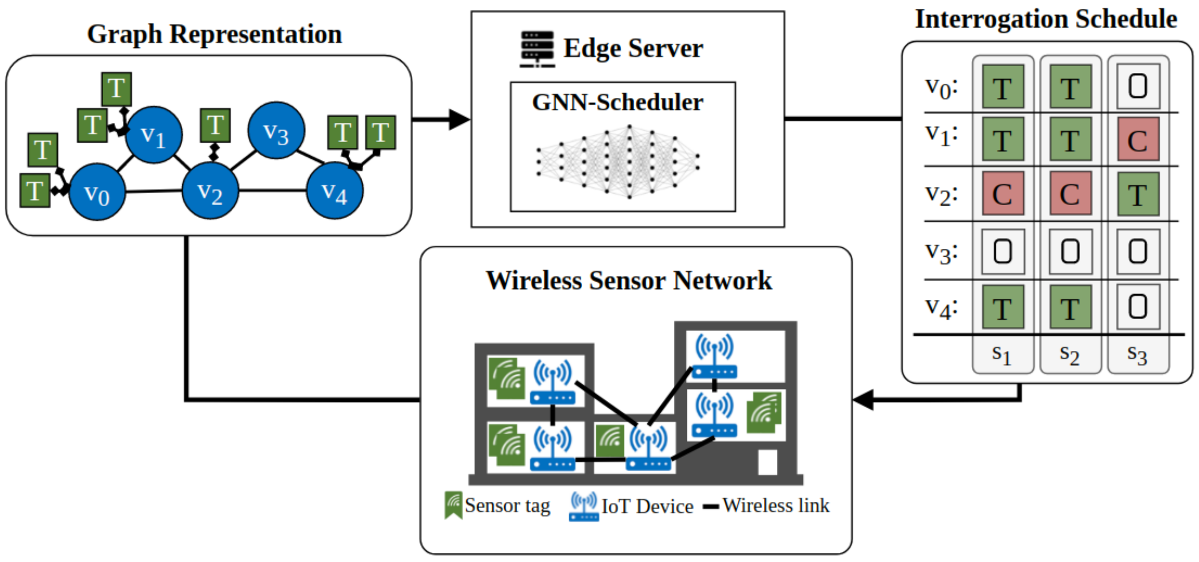

Figure 23 The scheduler takes a graph representing the network topology as input and produces a schedule. A schedule directs IoT nodes (v) to interrogate every battery-free tag (T) in the network with minimal resources by reducing the number of timeslots (s) and carrier slots (C).

4.1.4.3.2. *ML System Design*

The network of IoT nodes has the ability to collect link state information to determine the connectivity graph among themselves. This information is relayed to the cloud server, where it is, together with the tag-to-host mapping, used to assemble the graph representation of the network. Our scheduler uses this graph representation to produce a schedule, as depicted in Figure 23. An interrogation schedule consists of one or more scheduling timeslots, each assigning one of three roles to every IoT device in the network: Provide an unmodulated carrier (C), interrogate a sensor tag (T), or remain idle (O). The goal of the scheduler is to generate an interrogation schedule that queries all tags while requiring as few carrier generation slots and timeslots as possible. The number of carriers is the most important aspect affecting the overall spectral efficiency and energy consumption of the system. The ML-scheduler takes the network's topology graph as input and generates a corresponding interrogation schedule (see Figure 23). We adopt a supervised learning deep neural network approach instead of other paradigms such as reinforcement learning. This choice is mainly motivated by two facts. First, we can leverage the optimal scheduler to produce a training set necessary for a supervised approach. Second, it is straightforward to cast the scheduling problem as a classification problem, which is generally tackled with a supervised approach.

The ML-scheduler performs node classification on every IoT node to predict the role each of them will play within every schedule timeslot: remain off (O) or interrogate a tag (T), which corresponds to class

0, and generate a carrier (C), which corresponds to class 1. We seek to obtain the shortest possible schedule with the fewest number of carrier generation slots (C). The properties of the problem allow it to be modeled as an iterative per-node multi-class classification problem. The iterative nature of the approach is depicted in Figure 24: at each timeslot, the ML-scheduler performs a one-shot prediction for the IoT nodes in the topology assigning them to either class 0 or class 1. After this prediction, the tag information in the wireless network is updated: those tags that were predicted to be queried are removed from the topology. The new network state is fed to the same ML-scheduler to perform the prediction for the next timeslot, and the process is repeated until no more tags are present in the network.

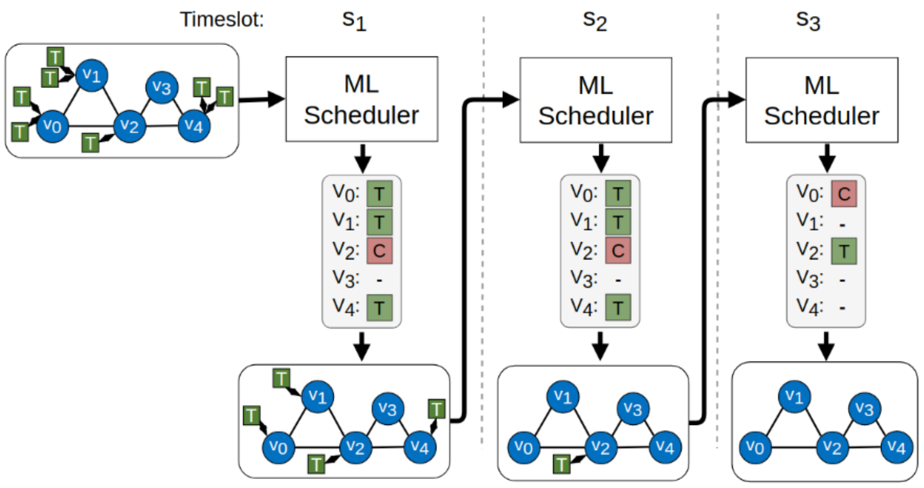

Figure 24 The ML-Scheduler iteratively performs node classification on every node, one timeslot at a time, removing scheduled tags from the topology and repeating the process until no more tags remain.

The ML scheduler consists of a stack of GNN blocks followed by a fully-connected neural network layer to perform the one-shot node classification. The training of the ML scheduler is done in an offline manner and can be performed where resources are not so critical, such as the cloud or the far Edge. Once the model is trained, it can be deployed at the near Edge to perform inference according to the procedure depicted in Figure 24.

4.1.4.4. Evaluation

To evaluate the feasibility of GNN to learning topological structural properties of wireless sensor networks, we first train the ML scheduler using small-size problem instances (networks with less than 11 IoT devices and less than 14 sensor tags) for which we can compute the optimal schedule using a constraint optimizer. One the training phase is over, we then implement the trained ML scheduler and evaluate its scalability to much larger, unseen, problem instances.

4.1.4.4.1. Training the ML Scheduler

We generate problem instances with topologies having N nodes and T tags of combinations from all possible combinations of the sets N = {2, 3, 4, …, 10}, and T = {1, 2, 3, …, 14}. The topologies are generated using the random graph generator from NetworkX [Hagberg'08] and the tags are placed randomly uniform among the nodes. Multiple tags per nodes are allowed. For computing the optimal solution, we use MiniZinc constraint optimizer [Nethercote'07]. To break the symmetry in cases where multiple solutions are available, we add additional optimization objectives to prioritize assigning carrier to nodes with lower node-IDs and to schedule as many tags as early as possible in the interrogation schedule. Note that random sequential node-IDs and tag-IDs were thus included. Since the slotframes of the interrogation schedules are permutation invariant and we include the complete tag information in the topology before performing the node classification, we can treat each separate slotframe of all the problem instances computed by the constraint optimizer as a separate data sample in the supervised classification setting. In total, around one million input-target data samples were generated (topology-schedule pairs). The data samples were randomly shuffled before they were split into a train-set and a test-set, were both sets contain instances of topologies from all considered sizes.

For training we use a 20-80 test-train set split of the problem instances. We train the Inference Module with the Adam optimizer [Kingma'15] on the basis of mini-batch gradient descent with standard optimizer parameters and an initial learning rate of 10−3. As loss function we use the weighted multi-class cross entropy loss, whereby the weights are estimated according to the class distribution present in the train set. We implement learning rate decay by 10% every five epochs, and early stopping after three periods of 25 subsequent epochs without minimization of the test-loss, whereby we re-set the learning rate to the initial value each time an early stop condition is reached to anneal the search space. We save the best performing model on the basis of the F1-score on the test set. The ML scheduler is instantiated using PyTorch and PyTorch Geometric, where more than 10 GNN layers were utilized for the model. We train the model using an NVIDIA Titan RTX (requiring only ~3000 MiB of GPU memory) for 131 epochs (or 14.6 hours) before reaching the early-stop condition.

Figure 25 shows the behavior of different training metrics. One can see that the learning rate restarts are responsible for the spikes in the graphs at approximately epoch 75 and epoch 115. These graphs demonstrate that the ML schedule is able to correctly predict the individual timeslots of the interrogation schedules and learn the topological dependencies that the constraint solver uses to produce the optimal schedules. The model achieves 99.82% accuracy, a carrier-class F1-score of 98.4% and is able to correctly compute schedules for 99.43% of the problem instances analyzed. The ML scheduler performs within 3% of the optimal scheduler in terms of the average number of carriers utilized in the interrogation schedules.

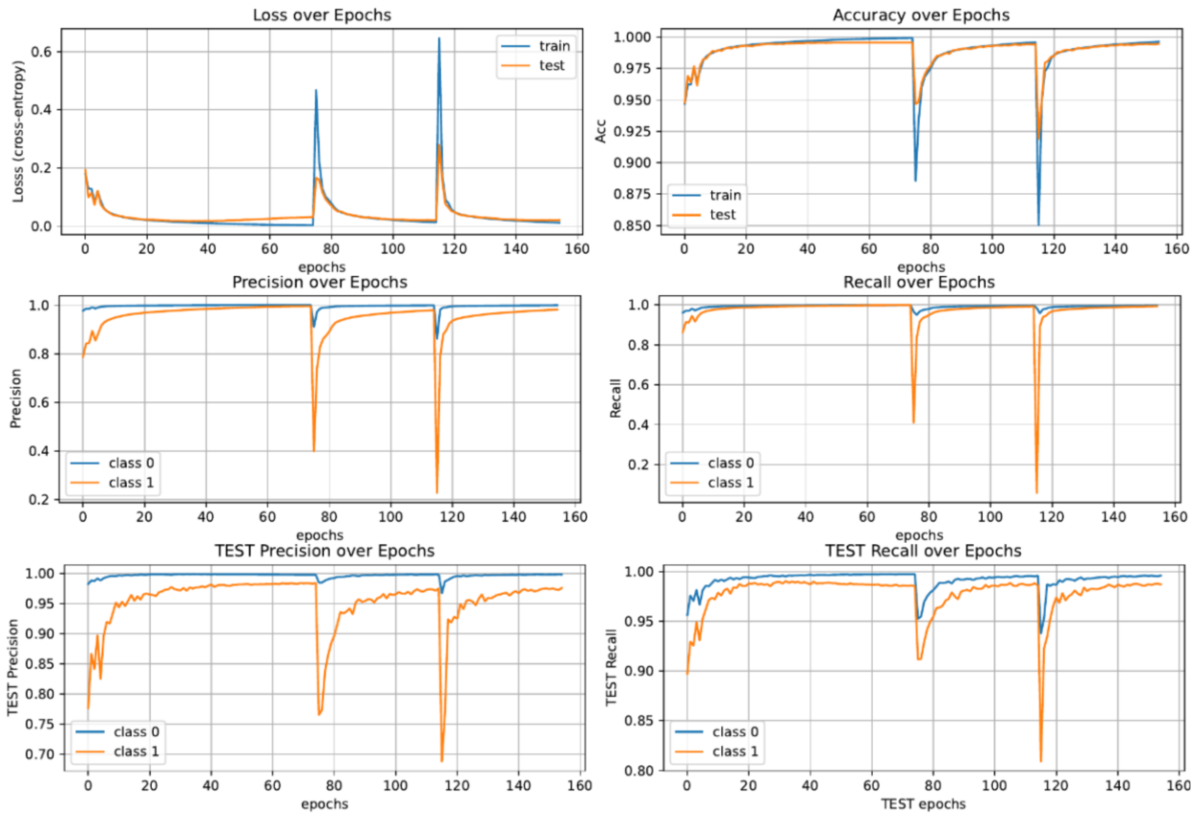

Figure 25 Performance results after training the proposed GNN under the supervised learning setting using the optimal solutions from small problem instances.

4.1.4.4.2. *Deploying the Scheduler for Larger Problem Instances*

After successfully training the ML scheduler, we compare its performance relative to the state-of-the-art heuristic, TagAlong [Perez-Penichet'20]. We compute schedules for 1000 problem instances for each combination of number of nodes N={10, 20, 30, 40} and tags T={60, 80}. Compared to TagAlong, the ML scheduler is able to reduce the percentage of necessary carriers ~ 10 % on average for large numbers of tags, as depicted in Figure 26. The ML scheduler is able to reduce the number of carriers by up to 40% for all number of nodes configurations. It worth noting that the optimal scheduler would require prohibitively long time to produce a solution to the candidate cases evaluated in this subsection, which yields it practically unapplicable.

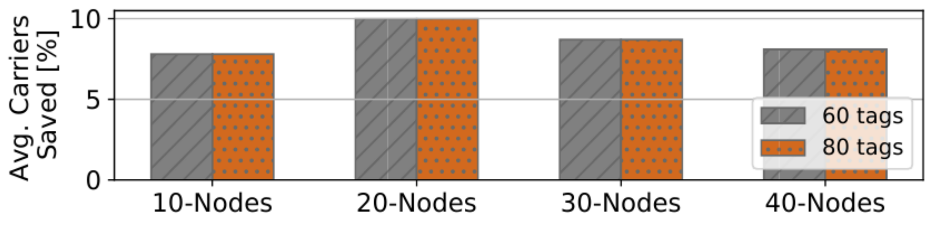

Figure 26 The ML scheuduler maintains an average saving of almost 10% in scheduled carriers while scaling up to four times the maximum training network size. Average percentage of carriers saved by DeepGANTT compared to TagAlong for various topology sizes and numbers of tags.

The final factor that determines the real-world applicability of a scheduler is the computation time. The ML scheduler has an inference time that is always below 1.9 s and on average is 500 ms for the largest problem size considered, a radical improvement over the optimal scheduler and in the range of the benchmarked heuristic. While TagAlong runs faster, the absolute values are so small that the difference is negligible in practice.

4.1.4.5. *Integration in the architecture*

The integration of the proposed ML scheduler for generating interrogation schedules in large-scale wireless sensor networks is described in Figure 27. The ML scheduler consists of the GNN model that is either i) trained locally by the "Model Handler" using data samples from the "Training Data Handler", or ii) updated with model parameters coming from another AIF instance upon receiving a re-configuration flag through IF1. In principle, multiple instances of the AIF can be deployed in different geographical regions, and their global parameter update can be controlled through their "Model Handler" using AIF interfaces IF1 and IF4 to start and terminate the process and IF2 as secure channel to exchange model parameters. At inference, the only active module within the ML scheduler would be the "GNN model", which received the network configuration information through IF3.1, and deliver the corresponding schedule to the wireless network through IF3.2.

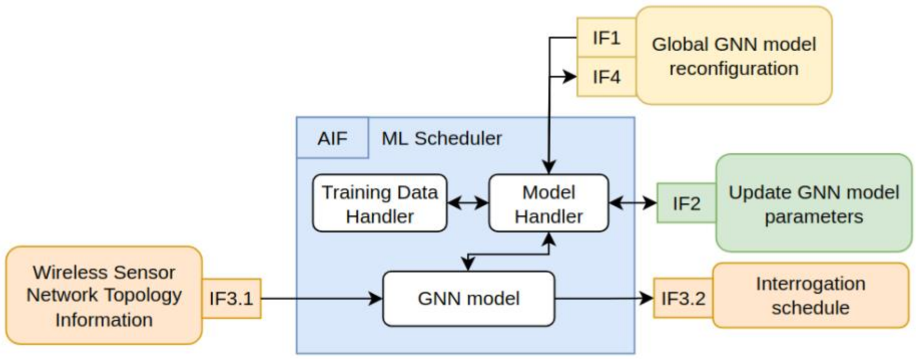
IF1
IF4
Global GNN model reconfiguration
AIF
ML Scheduler
Training Data Handler
Model Handler
IF2
Update GNN model parameters
Wireless Sensor Network Topology Information
IF3.1
GNN model
IF3.2
Interrogation schedule

Figure 27 Integration of the ML Scheduler in the overall AI@Edge architecture with its respective interfaces.

4.1.4.6. Summary

This section introduced a ML system that employs supervised deep learning to schedule a network of IoT devices interoperating with battery-free backscatter tags. The ML scheduler leverages recent advances in GNNs to overcome the challenges posed by the graph-structured nature of the problem and by variable-size inputs and outputs. It also exhibits strong generalization capabilities to problem instances up to four times larger than those used in training and can compute schedules, requiring on average 10% and up to 40% fewer carriers than an existing, carefully crafted heuristic, even for the largest problem instances considered. More importantly, the ML scheduler performs within 3% of the optimal on the average number of carrier slots while lowering computation times from hours to fractions of a second. Furthermore, the characteristic of the approach to deal with multiple solutions in the scheduling problem could have far-reaching implications in solving the broad class of graph-related NP-hard combinatorial problems using supervised ML techniques, and will lay the ground for future work within the project.

4.2 Advanced support for AI-enabled applications

4.2.1 Anomalous event detection with Federated Learning

As a core step in a closed-loop automation system, we need to determine upon which event to trigger reconfiguration of the connect-compute platform. We refer to such events as anomalies. In particular, we identify as an anomaly whichever deviation from the nominal working conditions of the system. Anomalies can therefore be, for instance, attacks or infrastructure impairment states such as due to congestion or service degradation.

Anomaly detection in SDN/NFV systems driven environments is a challenging task compared to legacy hardware-based networks due to a number of reasons; e.g., the multi-layered nature of softwarized networks, the introduction of new faults and vulnerabilities stemming from the decoupling of software from hardware, and their network provisioning and reconfiguration flexibility. Such environments do call for a data-driven framework rather than a model-based one, to scale with the multiple dimensions and large scales of virtualized infrastructure components' monitoring data.

In the frame of AI@EDGE, we are extending a preliminary framework [Ae] based on Long-Short-Term-Memory (LSTM) AutoEncoders (AE), called SYRROCA (SYstem Radiography and ROot Cause Analysis), to include a larger variety of network nodes, both physical and virtualized environment, and also covering virtualized RAN nodes such as eNodeB, gNodeB, radio-front-end information and disaggregated RAN components.

4.2.1.1 Objectives

As there are large varieties of network components in both physical and virtualized forms, it is important to provide a framework to inspect and detect anomalies in a collective and non-centralized manner. In AI@EDGE, our goal is to implement and compare two methodologies for anomaly detection in computing environments: centralized learning and Federated Learning (FL). While the former is an adaption and application of [DIA20] to the AI@EDGE environment, in the latter training of ML models are carried out on nodes where the data exist. These nodes could be any network component, both physical or virtualized, that has data and compute powers to run the training process.

In both cases, the training operations are coordinated and supervised by a single central node (the aggregator in the FL case), which is also responsible for aggregating the results obtained from the participating nodes. At the core of our framework, a LSTM ML model autoencoder will be used. The notion behind using LSTM and autoencoders is that LSTM is suitable for time-series as it has the capability of learning long-term dependencies between timesteps [Hoch97]. In the meantime, autoencoders present good mechanisms for detecting anomalies and outliers.

4.2.1.2 State-of-the-art

Before a system can detect anomalies and make the necessary adjustments to restore itself to the normal working conditions, the nominal state has to be identified. For distributed and multi-layered systems it is challenging to model nominal conditions for each component [Shaw02].

A well-known approach consists of the application of a set of association rules and frequent episode patterns to classify events as an anomaly or not. While rules tend to be intuitive, they are inadequate to represent many types of anomalies, as is the case for softwarized networks due to the immense variety of faults and threaten. Inductive rule generation algorithms have been proposed to overcome this limitation, as for example genetic algorithms [Crosbie95].

Statistical theory outcomes are also widely used to detect anomalies. For example, in [YE01] the chi-square test statistic value is used as a distance measure to detect anomalies: when an observation chi-square value is greater than a fixed threshold, the observation is tagged as an anomaly. In [Stan02] authors present the Statistical Packet Anomaly Detection Engine (SPADE) as a statistical anomaly detection system. A simple frequency-based approach is used to calculate the 'anomaly score' of a packet: the fewer times a given packet was seen, the higher was its anomaly score. Once the anomaly score crossed a threshold, the packets were forwarded to a correlation engine that was designed to detect port scans. [Pasch10] uses Markov models to characterize the "normal" behavior of the sensor network. In particular, a series of Markov models are developed and for each model, an anomaly-free probability law is estimated from past traces. Then, recent observation is studied with the Large deviations theory to understand if the empirical measure takes very unlikely values.

With the growth in availability and in amount of monitoring data which characterize latest and future network, machine learning (ML) started to be applied to anomaly detection as well.

When dealing with high dimensional data, one of the most widely used approaches is to project the data into a lower dimensionality sub-space, spot in this sub-space spontaneous clusters of data and then tag as anomaly those data which fall apart from the clusters. Principal Component Analysis (PCA) and K-Means are the most used algorithms respectively for dimensionality reduction and clustering [Zang09]. For example, authors in [Munz07] trained a k-mean algorithm on unlabeled flow records to cluster normal traffic. Then, the distance from clusters centroids is used to compute samples anomaly score. Similarly, authors in [He17] develop a two-stage anomaly detection algorithm based on feature selection and Density Peak-Based Clustering to handle large-scale, high-dimensional, and unlabeled network data. In [George12] instead, first PCA is applied to the KDD99 dataset for network IDS, and then a Support Vector Machine (SVM) is used to classify anomalous and nominal samples. However, clustering-based algorithms showed to be sub-optimal, mainly for a high false-positive rate [Say12]. Furthermore, PCA is recognized to fail in capturing temporal correlation [Bra09] and in analyzing non-linear correlated metrics. Even though several non-linear approaches were proposed in the literature, it is broadly recognized that Deep Neural Networks (DNN) are very flexible and they can introduce a theoretically infinite level of non-linearities by using non-linear activation functions. In anomaly detection, one of the most widely used architecture is the Deep AutoEncoder (DAE) one [An2015]. In [Sak14] authors demonstrate that AEs clearly outperforms PCA in terms of accuracy and computation time. They also show that autoencoders learn the normal state properly in the hidden layers and that they activate differently with anomalous input.

In [Alawe18] authors propose a mechanism to scale 5G core resources by anticipating traffic load changes through LSTM and deep neural networks forecasting. They show that LSTM-based anomaly detection can be more accurate, thanks to its ability to store data pattern without degradation over time. [Mal15] propose a stacked LSTM architecture to detect anomalies within time series data by evaluating the deviation of predicted outputs based on a variance analysis. In [Ergen19], a compound architecture is presented. Here, the LSTM network predicts regular system dynamics, and a support vector machine is applied as classifier for anomalies to realize an adaptable and self-learning detection mechanism.

Federated learning has become a viable alternative to the centralized approach in ML in many areas, especially if data privacy is of a concern. In fact, Federated learning distributes the global model training process such that each learning participant's data is used to train a local model, rather than aggregating large amounts and types of data into a central location. Recently, FL has been adopted in network anomaly detection and proposed in a plenty of state-of-the-art works. Nguyen et al. [NMMF] presented an autonomous self-learning federated learning-based anomaly detection approach for IoT devices named "DIoT". It operates on device-type-specific communication profiles without human intervention. DIoT architecture consists of a security gateway to connect IoT devices to the Internet, and IoT security services. The anomaly detection component is integrated with a security gateway, which monitors the network for abnormal activity. A repository of device-specific anomaly detection models is maintained by the IoT security service where model weights updates from IoT devices are aggregated. Yurochkin et al. [YAGGHK] developed a probabilistic federated learning framework with neural networks. In a such approach a global model is constructed by matching estimated local model parameters across data sources using the Bayesian nonparametric (BNP) model. BNP allows the local parameters to either match existing global ones or to create new global parameters if the existing ones are poor matches. Li et al. [LCLWC] proposed an autoencoder based anomaly detection technique at the server side for anomalous detection of local weight updates from the clients in a federated learning system. The idea is to generate low-dimensional surrogates of model weight vectors and use them to perform anomaly detection on client data. In [ZCW], Zhao proposed a multi-task deep neural network federated learning (MT-DNN-FL) for network anomaly detection. It also performs traffic recognition and traffic classification.

4.2.1.3 *Overview of the method*

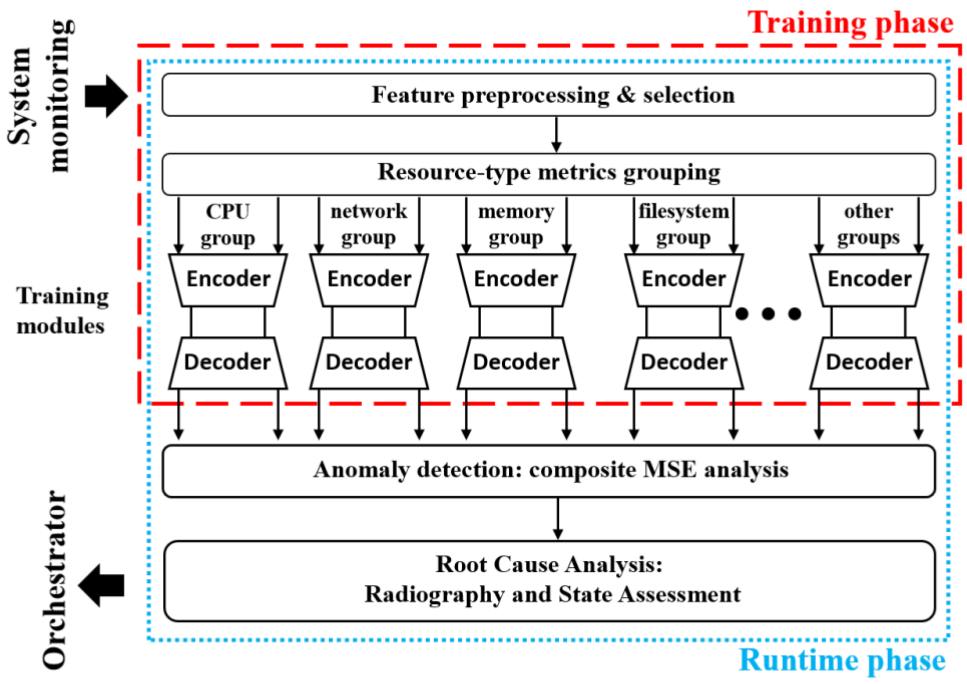
Training phase
System monitoring
Feature preprocessing & selection
Resource-type metrics grouping
CPU group
network group
memory group
filesystem group
other groups
Encoder
Decoder
Training modules
Anomaly detection: composite MSE analysis
Root Cause Analysis: Radiography and State Assessment
Orchestrator
Runtime phase

Figure 28 SYRROCA Framework

SYRROCA is used to detect and characterize anomalies in softwarized network environments. Starting from metrics collected at both physical and virtual (container) levels, a methodology to detect anomalies and present them is introduced to infer the running state of a virtualized network service. Through this methodology, it is not only possible to inspect the whole service, but also of their inner components (e.g. containers or VMs). Hence, supporting root cause analysis to explain network state deviations happening when particular network events and anomalies manifest.

As represented in Figure 28, the SYRROCA framework processes metrics of component-specific groups (e.g., CPU-specific group) from different layers (physical and virtual) and uses one autoencoder cell for each group. An autoencoder is a multi-layer Neural Network (NN) composed of two blocks: an encoder and a decoder. The encoder reduces the n dimensions of the input to s dimensions (latent-space), while the decoder takes those s dimensions to reconstruct back the input. The AE is trained to learn how to reproduce the input vector X of n features by learning how to optimize weights and biases on the encoder and the decoder NNs to minimize the overall reconstruction error. It has been demonstrated that composing several encoder and decoder layers to build a DAE allows to effectively represent complex distributions [ZHO17].

AEs can be used to detect anomalies as the decoder block compresses the data input dimensionality. Assuming that the input data has a certain correlation level, it can be embedded into a lower-dimensional subspace, where anomalous samples look significantly different from nominal samples, which makes anomalous samples reconstruction error increase significantly. AEs are considered an auto-supervised NN, as the learning target value is the input itself, so no labels are required in the training phase.

According to the type of dataset the AE has to analyze, several types of NN can be chosen. When it comes to dealing with time-series problems accounting for the temporal dimension, Recurrent Neural networks (RNNs) are generally used. However, they suffer from the vanishing gradient problem [BEN94], preventing long-term relations to be learned. As a solution, Long-Short-Term (LSTM) RNN has been proposed [Hoch97]. LSTM enforces constant error flow through the internal states of special units called memory cells by employing multiplicative gates which allows learning of long-term sequence correlations and modeling complex multivariate sequences [MAL15].

In SYRROCA, after an anomaly is detected, the framework performs an in-depth analysis of the AEs reconstruction error to identify the set of most deviated metrics. This analysis is then used to produce the so-called radiography visualization along with a state graph multi-layer representation. While the former compactly visualize the propagation of anomalies across layers, the state graph aims at establishing and characterizing the deviated state of the system as the basis for a re-orchestration algorithm. Our anomaly detection framework is extending SYRROCA to level up to AI@EDGE requirement. The framework will reflect the decentralized nature of the Edge components by applying FL to train SYRROCA instead of its centralized approach to ML.

As already mentioned, and represented in Figure 28, to have a greater control on training, data features collected for training are grouped by resource type. In the experiments of [DIA20], CPU, network, memory and file system related metrics are used. Likewise, additional sources of metrics can be safely added to the framework with no restriction. Data also has to be re-scaled to eliminate the huge variation in magnitudes of feature values by using the normalization technique (in contrast to standardization rescaling technique), which transforms the original metrics so that all values fall within the [0; 1] range.

During the training phase, the framework builds a model of the system during the delivery of a virtualized service. Training dataset is split into several sub-datasets each of them fed to a dedicated deep AE. AEs are trained with a dataset built in nominal conditions, and covering a sufficient period, so that they can learn an abstract representation of what is considered to be a nominal state.

On-going work is on the extension of the preliminary centralized framework described in [DIA20] toward its distribution across the network using a federated learning (FL) approach. The motivation is to, on the one hand, better scale with the large amount of data expected hence to envision on-line usage of the framework and, on the other hand, ensure low-latency processing and detection. The resulting FL-based anomaly detection AIF architecture, should therefore, leverages on multiple different AIFs:

edge AIFs for data pre-processing; centralized and intermediated AIFs for LSTM neural network (NN) parameter training and update.

Figure 29 represents the envisioned FL-based anomaly detection AIF systems.

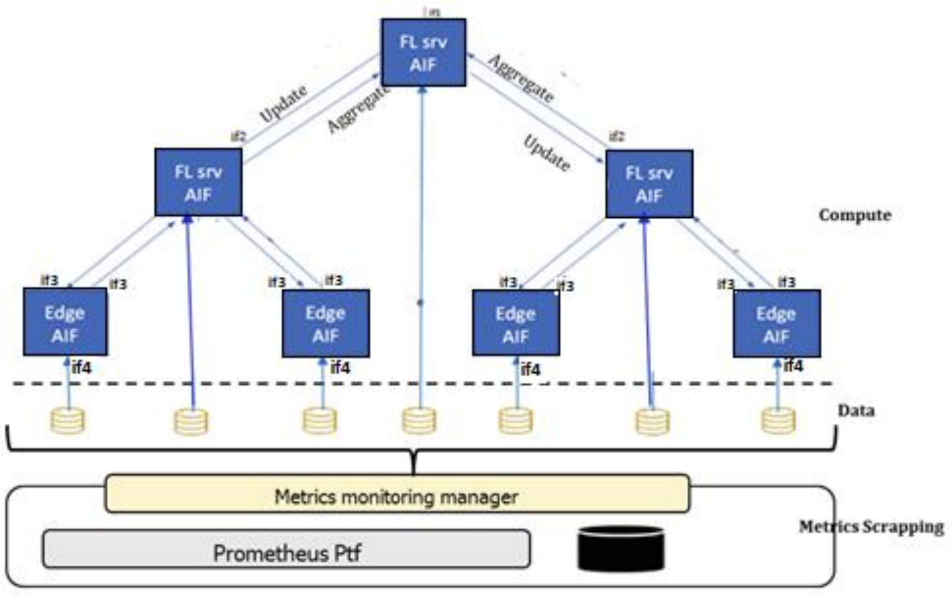

Figure 29 Graph representation of decentralized Distributed ML

The anomaly detection AIF system is therefore functionally composed of the following subsystems:

- a central FL server, responsible for the aggregation and update of a global ML model. Also, it is responsible for the initiation and orchestration of the training process. The central FL server can also be distributed hierarchically with multiple central FL sub-servers, depending on the deployment strategy.
- the edge AIFs. The AIF instances are distributed and are collectively responsible for training the model where each AIF trains part of the data. The training process starts initializing a global model, which is then distributed to all participating AIFs in the platform. Each AIF trains the model following a federated learning (FL) logic, on the part of the data that is specifically available to it, for a predefined number of epochs. Inference can also happen at the AIF level.
- A data-lake system, made accessible to all AIFs with even rights on the data, i.e., all AIFs access the same type of data and are configured to process different portions at different AIF instances based on load-balancing policies.

4.2.1.4 Integration in the architecture

We aim at integrating the FL-based redesigned implementation of SYROCCA in the AI@DGE functional and experimental architecture. Most of the AI@EDGE use cases are composed of distributed entities that communicate to each other at different levels (i.e. near-edge, far-edge and the cloud). At the ground level, for example IIoT at a factory, each end-devices group share the same characteristics and can produce the same type of data. ML training is applied using LSTM DAE which groups each monitored resource metrics together into a single autoencoder within the framework.

We can conceive two different data-partitioning approaches to deploy the anomaly detection AIF system, depicted in Figure 30.

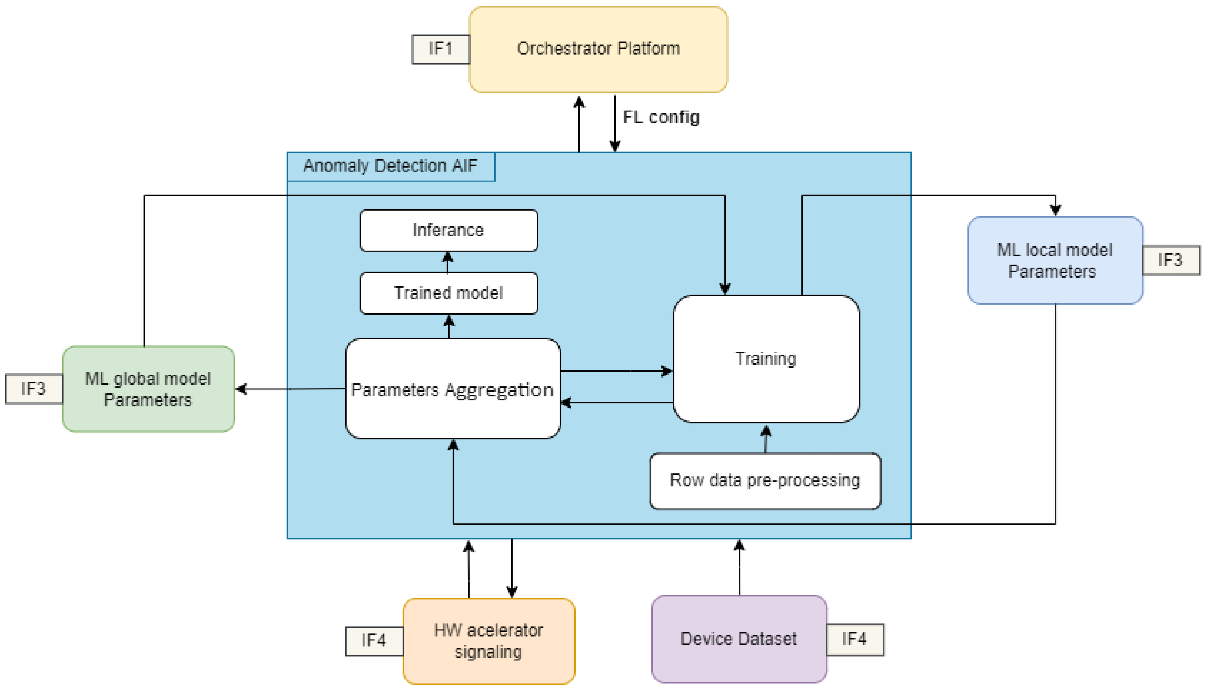

Figure 30 Anomaly detection FL-based AIF

4.2.1.4.1 Flat load-balancing approach

In this approach, the complete data-lake of the connect-compute is made available to edge AIFs. Edge AIFs act as an over-the-top computing system that can automatically scale as a function of the monitoring data rate and number of running nodes.

There is a dedicated layer for managing data collection and monitoring. The layer would have a repository for storing collected metrics and offer mechanisms for data analysis and integration, hence forming the platform data-lake. Indeed, at the heart of the layer, there would be a parameter scrapping system (Prometheus platform for example). Such a layer provides the data-lake needed for distributed anomaly detection AIFs. The metric monitoring layer and the ML algorithm either could reside at the edge or in the cloud where they will benefit from the power it offers; it is a matter of trade-offs between resources availability and communication constraints.

In this approach AIFs are used to absorb the anomaly detection load, with equivalent AIF instances, and no locality constraints on the data made accessible to each AIF instance.

During the execution of distributed ML algorithms, they may have to stop in order to exchange parameter updates (i.e., synchronize). In order to reduce the time spent waiting, it is desirable to load-balance the work on each AIF. This is especially important in our case as the datasets exhibit non-iid data and as data is collected from different physical and virtual components that are deployed to perform different tasks in different environments. For example, gNodeB metrics, and function components of the 5G core network.

The load balance should guarantee the loading of data to AIFs to maximize their participation in the training process and reduce the stragglers effect (delaying the update of the global model) which brings down the performance of the whole system.

4.2.1.4.2 Device-specific hierarchical approach

As explained in D2.1 [D2.1], federated Learning is a distributed ML paradigm where several nodes are collectively participating in training global ML models locally under the orchestration of a central server.

Its aim is to produce a generalized model that captures all the different distributions associated with each node's data, without the need for the data to leave the node. Indeed, FL produces a single output for all the participating nodes, and therefore, it does not adapt the model to each node completely. Especially in heterogeneous settings, where the underlying data distribution of nodes are non-iid, the performance of the global model obtained by minimizing the average loss could be poor once applied to the local dataset of each node.

In such an approach, following the scheme in Figure 30 , an AIF learns on data communicated through it by IF4 interface, whose data should exist on the same node where the function runs. Moreover, the ML global model (i.e. weight and bias parameters) is also sent to the AIF through the IF3 interface and it is used to carry out the training. The result of the training is a ML local model that is sent over the IF3 interface for the aggregation with the other FL AIFs that are part of the FL training framework. As a result, the Global model and local models are exchanged multiple times during the training process.

Indeed, there should be an initialization phase before the AIF start to learning. The global model has to be initiated with given parameters through IF1, together with the hyperparameters that control the duration of the operation, the learning rate and other house-cleaning configurations. Modern ML development frameworks provide for the utilization of accelerations. As such, our AIF can benefit from that by running on accelerators and exchange signals through IF4 interface. The result of such operations by the AIF is a ML model that is ready for deployment as an inference AIF which is used for anomaly detection.

Differently from the flat load-balancing approach where different edge AIFs work on the same type of data, a device-specific approach can therefore be designed with differentiated AIF data profiles. In this way, each group of devices belongs to a specific FL instance to which all devices of the same types are attached. For instance, we would have one group for each class of IoT device, one for gNodeBs, and one for 5GCN function instances.

As the expected number of nodes can be high, the AIF system can be made more scalable passing from a baseline two layers aggregation to three layers aggregation, where each group of nodes of the same type (e.g., gNBs or UPFs) is clustered. Each cluster will have a cluster central server residing in the middle of the nodes of that cluster and a central server above it in the hierarchy. In such a hierarchy setting, the cost of communication will be reduced. The AIFs will receive data through IF4 interface. The exchange of the ML model will happen via IF3, while the ML control plane interface used to exchange model parameters. The IF1 is the northbound interface used for (re)configuring the AIFs, i.e. setting hyperparameters, accuracy value, etc. IF3 could be also used to exchange mutual parameters among middle FL server, in case a collaboration between those servers is envisioned.

4.2.2 Predictive approaches for radio context and connection performance for distributed and federated learning

One of the benefits provided by the AI@EDGE platform is the convergence of communications resources provided by mobile networks and computing resources enabled by the edge systems. This compute-connect paradigm facilitates the applications of artificial intelligence/machine learning techniques. The role of AI/ML will be two-fold: on the one hand, this paradigm will facilitate the proliferation of online AI services (network for AI); one the other hand, it will embed AI functionality to optimize the network operation (AI for network). As a focus of the latter, predictive radio context and performance will use machine learning techniques to provide radio prediction, and then apply this predicted information to promote the operation of distributed and federated learning operated in this system.

4.2.2.1 *Objectives*

Federated learning and distributed learning need to exchange information frequently, such as the distribution of a universal model to the participants, the transmission of a partially trained model to other nodes, and essential control signaling and management messages. The origin of Federated learning and distributed learning comes from the Internet, where the default connectivity among the learning nodes is implemented with wired links such as optic fiber and LAN. The properties of a wired link, including transmission rate, delay, packet error rate, and availability, are regarded as deterministic. Moreover, these parameters keep constant for a long period on the magnitude of minutes, hours, and even days until the network status varies, for example, because of the maintenance of the equipment or the transmission link. When these learning methods migrate to wireless networks, the learning nodes suffer from a dynamic environment. To be specific, a learning node could be located at the cell edge, where the received signal is weak, while the co-channel interference from its neighboring cells is strong, leading to low transmission rate and high packet error rate. In contrast, a learning node close to the serving base station has much higher data rate with high reliability. If a learning node stays at the cell edge and does not move for a long time, it becomes a bottleneck and degrades the performance of the whole system carrying out federated learning and distributed learning. Does the system have the ability to predict such nodes and stop the learning tasks on these nodes? In addition to the radio context in large-scale granularity, wireless channels vary randomly quickly in small-scale granularity on the magnitude order of milli-second due to the constructive and destructive interference of multi-path components. Such dynamics has a strong impact on the performance of a transmission link, which in turn determines the efficiency of the learning system. If the radio context can be known beforehand with some predictive approaches, the learning system is able to do some adaptive methods to improve the efficiency, robustness, and performance of federated learning and distributed learning at the network edge.

At the initial stage, two steps can be expected: (1) finding suitable ways to get the data for training and testing machine learning (through statistical modeling or measurement, if possible); (2) an investigation of the feasibility of machine-learning algorithms.

4.2.2.2 *State-of-the-art*

Through modeling a wireless channel into a set of radio propagation parameters, two statistical prediction approaches – Auto-Regressive (AR) [Due07] and Parametric Model (PM) [ADEOGUN] – have been proposed. However, the modelling is fossilized, leading to a gap between these models and real channels, and – in addition – the parameter estimation process relying on complex algorithms such as Multiple Signal Classification (MUSIC) and Estimation of Signal Parameters via Rotational Invariance Technique (ESPRIT) [GARDNER] is tedious, harming its applicability in practical systems. As a classical AI technique, neural networks (NNs) [JIANG1] can avoid the parameter estimation thanks to its data-driven nature, and therefore attracts the interest from researchers in the field of channel prediction. Making use of its capability on time-series prediction, the predictors based on deep learning with advanced recurrent structures such as LSTM or gated recurrent unit (GRU) [JIANG] exhibit good predictive performance over channel quality, which is promising to be applied for this scenario. However, the current deep learning-based prediction approaches focus on the single-user mobile environment, which does not fit with the multiple learning terminals working simultaneously. In this project, therefore, the focus will be on the multi-user prediction for the setup of federated learning and distributed learning

4.2.2.3 *Overview of the method*

The link performance of wireless transmission heavily depends on the quality of wireless channels. Since a wireless channel is highly dynamic and time-varying, a transmitter always prefers to adjust their transmission parameters for better performance. With the assistance of channel state information (CSI), the transmitter is able to adaptively choose its parameters such as the transmit power, constellation size, coding rate, transmit antenna, and precoding codeword to achieve great performance. If we can track and predict the CSI of all the links in a federated/distributed learning system, we can achieve better communication performance to guarantee the efficient operation of the intelligent system.

4.2.2.4 Integration in the architecture

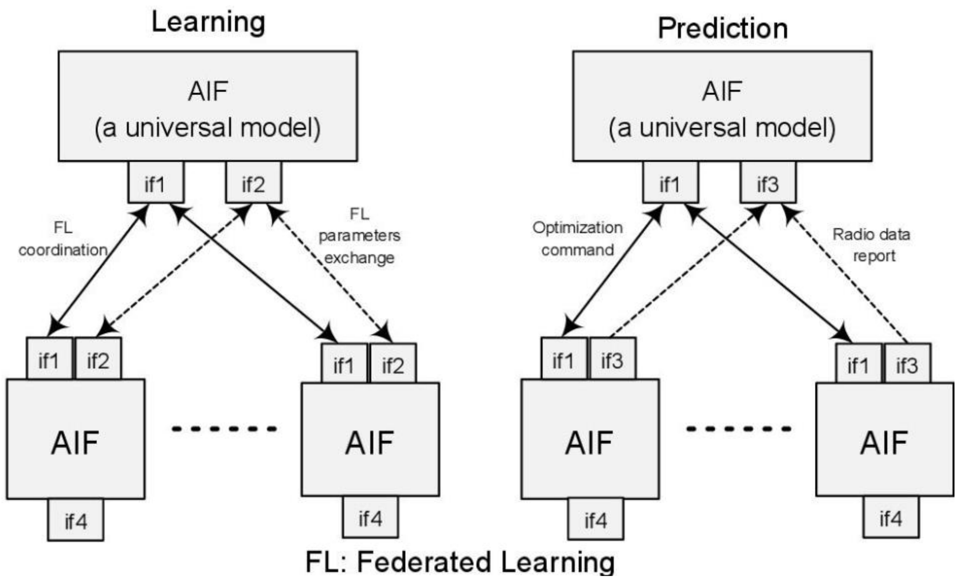

Figure 31 Schematic diagram of predictive radio context and performance optimization for feaderated and transfer learning, which is divided into two phases: learning and prediction.

In the AI@Edge platform, the consumers of the computing and AI capabilities offered by the edge are usually mobile and portable equipment connected through wireless connectivity. To realize predictive radio content and performance approaches, these mobile equipment or personal devices need to run at least one AIF to assist the main predictive functions to train a universal model (for federated learning or transfer learning) and preprocess the required data. The universal model is contained by one or more AIFs running generally at radio access site (the far edge of the AI@Edge platform) to control the radio resource and optimize the performance of all mobile nodes in the coverage of one site. It is possible that the universal model is performed in a higher level, e.g., the local access site (the near edge of the AI@Edge platform), to obtain a wider view of the communication network and computing resources, in order to optimize, e.g., the inter-cell interference, to improve the performance at the cell edge. It is possible that a set of AIFs can be deployed in a distributed manner in the near edge and the far edge to collaboratively make decisions.

In such a scenario as illustrated in Figure 31, the first step is the AIF operating in the far edge or the near edge that uses IF4 to align the AIFs on mobile equipment to trigger a federated learning process, while the distributed AIFs receive the configuration information via their IF1 in a master-slave fashion. The AIFs need to use the IF2 to get an initial universal model for training, and then continuously send the updated model parameters to the universal model(s) to iteratively update its parameters via IF3. Once the training of the universal model is converged, the master AIF at far/near edge switched from the training mode to the prediction mode, and align this process via IF1 and IF4. During the prediction, the AIFs dedicated to predicting the radio context and performance needs to continuously report the radio context information to the AIF used for resource management and system operation since the radio environment is time-varying, especially in high mobility. Then, the data pipeline is required for the AIFs to transfer the information through IF3; otherwise, the information will be aged and degrade the system performance.

The main functionality of the aforementioned AIFs is to predict the radio context and its associated performance. It is possible that more AI/ML-based radio resources control and optimization functions can be jointly applied. These AIFs can be allocated distributedly, one or serveal AIFs per mobile devices, at the far edge, or at the near edge, based on what the function it is, e.g., inter-cell interference, scheduling, and adaptive modulation. Therefore, the AIFs for predictive radio context needs to send the up-to-date prediction to the AIFs for dedicated optimization. These kinds of information are better to transfer through IF1/IF4 for high reliability.

4.2.3 Autonomous system operations and continuous learning processes

The AI@EDGE project aims at the management of networks with minimal human interference. To achieve this goal, the usage of artificial intelligence is going to be very important. Artificial Intelligence methods consist of the usage of a dataset to train a specific model. This model will learn how to minimize a specific cost function. Traditionally, the training step is done offline before the deployment. This approach works well for a static environment. However, network conditions and statistics tend to change over time. For this scenario, the statistic dataset used to train the model will not represent the real environment resulting in a loss of performance. To improve the model performance, the usage of continuous learning appears as a learning paradigm used for the stream of data. This appears to be more suitable for the AI@EDGE environment.

4.2.3.1 Objective

The objective of this subsection is to describe the usage of federated learning in a continuous learning environment and to use this method to enable autonomous system operation. Federated Learning is a technique deployed in a distributed scenario where there are two components: the clients and the server. The former is composed of the clients (e.g., mobile phones, vehicles, IoT sensors, personal computers) that receive a global model from the cloud, train it using its data, and send to the cloud the model updates. The latter component of federated learning is composed of a central model that is responsible for generating a new global model based on the client update models. The usage of federated learning improves privacy performance as no personal data is sent to the cloud server. To implement continuous learning in a Federated Learning scenario, some updates on the traditional methods need to be implemented (e.g., solve the catastrophic forgetting problem and solve the asynchronous update problem) as described in the next section. The usage of continuous federated learning can enable autonomous operations by continuous monitoring of a status of a specific component of the network and use the models to help about what decision should be taken given the input of the models. As in federate learning there is one or more clients, each client can be deployed in a specific part of the desired system to be automated and can collect different types of data and do the learning step to improve the general model parameters. This general model will be used to help about what is the best decision to be made to improve the necessary system capabilities. Also, this federated learning algorithm will be combined with the proposed data pipeline system to enable the necessary granularity of data in the edge.

4.2.3.2 State-of-the-art

The usage of Federated Learning in continuous learning approach is a recent area of research. The combination of the distributed learning fashion of Federated Learning with the continuous learning can contribute to an efficient implementation of applications composed of AIFs in the AI@Edge network. In offline learning, the model is trained using a predefined dataset and, after the training step, the model is deployed and runs using a different dataset from the one used during training. In contrast with offline training, in continuous training approach the model parameter is updated in a online way which means that the model can train using a stream of data instead of an offline dataset. In [Le21], a combination of federated learning with continuous learning is developed resulting in a method called **F**ederated **C**ontinuous **L**earning based on **B**road **L**earning (FCL-BL). In this method, the problem of catastrophic forget is solved by the usage of Broad Learning algorithm [broad learning paper].

In [Yoon21] is proposed a method called Federated Weighted Inter-client Transfer (FedWeIT). In this method, there is a decomposition in the global network weights and in the called "sparse task-specific parameters" to let each client to receive a specific knowledge from other clients [Yoon21]. One of the goal of this method is to minimize the interference between tasks that are incompatible. The code of the method can be found in [Yoon21Code].

Even if Federated Learning is known as a privacy concerning algorithm by avoiding the sending of data through the network, there is still work aiming to improve the security of Federated Learning. As an example, in [Zuh2019] the security is improved by the usage of compression on the gradients. The authors proved that the compression of the gradients can improve the security against leakage without decrease the accuracy of the system. In [Huang2020], the security is improved also by the usage of compression. However, different from the previous work, in this method, the compression is done by net pruning.

4.2.3.3 Overview of the method

The proposed method is depicted in Figure 32. The first component that will be listed here is the Server Layer that is responsible for updating the global model based on the client model updates. The next component of the method is the Client Layer which is responsible for the training step based on its data. Instead of the static dataset, in this scenario, the data changes as new data appears to the client. This scenario can lead to the problem called catastrophic forgetting [YI20]. To describe this problem, Figure 33 describes two datasets called "A" and "B". In timestamp 0, the model is trained using the dataset "A". In timestamp 1, the new data generated the dataset "B". If the model is trained for dataset B, its new configuration and parameters will not be suitable for the old dataset "A" anymore. The model "forgets" all that was learned using the dataset "A". One approach to overcome this issue is to train the new model using a merge of the datasets "A" and "B" creating the dataset "C". However, this approach is not efficient in terms of computational cost because of the dataset growing process.

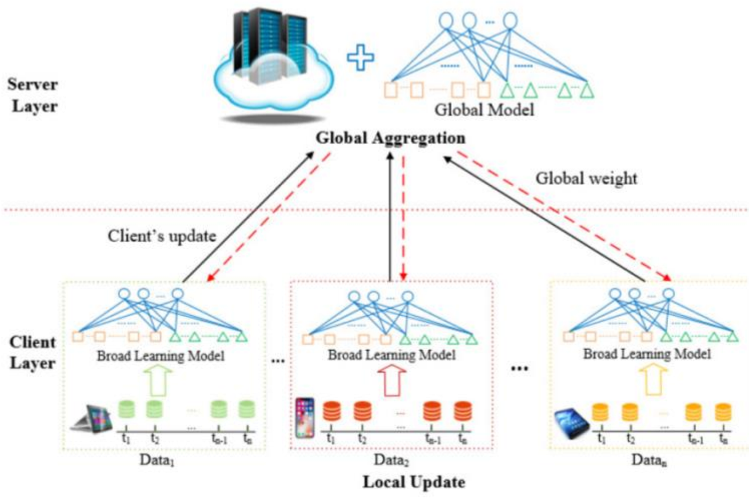

Figure 32 big picture of the proposed method [YI20].

The proposed method overcomes this issue by using the technique called "Broad Learning" [Chen18]. This method resolves the catastrophic forgetting problem and is deployed in each client as described in Figure . In the first iteration of the method, the server layer deploys the initial global model for each client layer. In the client layer, the local training is done by using the broad learning technique. After the local train step, the client model update is sent to the server. The Server will update the global model only after a predetermined number of updated clients model is received. This approach is called "Asynchronous" because there is no necessity for all the selected client models to send to the server its updates. Instead, only a few clients need to send it. This approach improves the speed of the global training as there is no necessity to wait for all the selected clients to send to the server the updates.

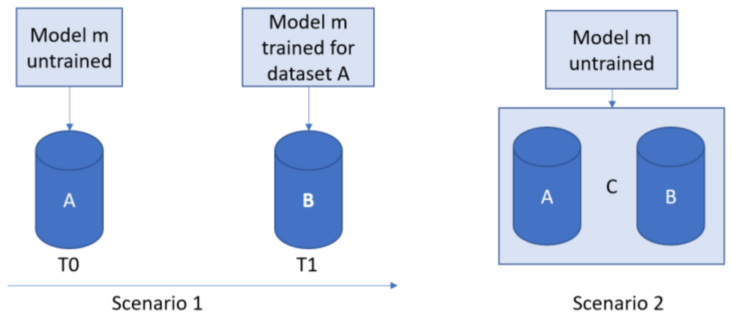

Figure 33 catastrophic forgetting problem illustration.

4.2.3.4 Integration in the architecture

In the framework of AI@EDGE, federated learning in a continuous scenario can be implemented under the context of the AIFs. Each client should have an AIF responsible for local training and the cloud server has an AIF responsible for the global update model. Figure 34 illustrates the implementation of the proposed method in the framework of AI@EDGE using AIFs. In summary, an AIF is deployed in the server layer who is responsible for the global aggregation of the client updates and to generate a new version of the global model at each iteration of the federated learning algorithm. The second type of AIF is deployed in each client. In each client, the AIF is responsible for the continuous learning local algorithm, update the global model received by the server layer, and to send to the server the client updates. Regarding the usage of the interfaces, the interface IF3 is responsible for sending to the server the updates of the client where IF2 is responsible for sending to the client the global model parameters.

This method can receive as input a specific data format depending on the application. As an example, at user application level, this method can receive the UE-level (described in Section 5.4) data to improve the capabilities of the application (e.g, by improving the model parameters given the change of the statistics of the user data) or can improve the eNodeB (described in Section 5.3) or gNodeB (described in Section 5.5) automation managements given these data input.

To enable the deployment of this method with the necessary granularity of data, a data pipeline system should be designed to enable the delivery of the necessary data with the necessary granularity to each component of this method (e.g, the server AIF and the client AIFs). This data pipeline architecture design is proposed in Section 3.2. In the data pipeline system proposed in this project, the data collectors is used to collect the data from the data sources and deliver to the necessary components through the data ingestion pipeline. To avoid the unnecessary collection of the same data, the method can first check if the desired data is already in the data repository or in the data analytics component. This will avoid the unnecessary stream of data through the data pipeline system and will increase the granularity of the data needed.

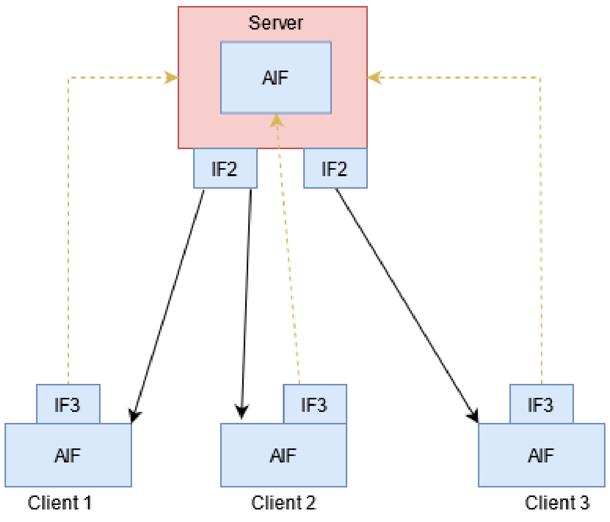

Figure 34 continuous learning using federated learning in the AI@Edge framework

4.2.4 Distributed and collaborative models for service placement on distributed and limited edge resources

Networks will evolve into a broad intelligent and distributed system created by intelligent units. This collaborative intelligence will enable the various elements of the network and services to be self-configured and dynamically self-managed [SUB21]. All of them will respond to a constantly changing environment, sharing knowledge of how to cope with their surroundings and generating behavior to obtain optimal responses with minimal human interference. In this way, solutions should be scalable to provide an elastic system, adapting to increasing or decreasing demands.

In this scenario, due to the rapid growth of data services and applications both in volume and in variety, there is a need to scale up AI/ML/DL/RL techniques [YE20] [YU21]. Federated learning (FL) is proposed as a distributed ML solution for learning on edge devices [WAN19]. In FL, data does not leave users' devices and all users collaboratively train a model without sharing their data [LAM20].

4.2.4.1 Objectives

The main objective of this subsection is to develop and evaluate a distributed and collaborative system applying AI and NFV, placing, creating and executing functions at the edge according to the services and application control and management requirements.

Thus, this subsection is mainly focused on algorithms and methods for enabling distributed AI/ML and federated learning over MEC and cloud infrastructures, supporting adaptive, re-scalable and resilient distributed computing infrastructures, encompassing local and global learning models for resilient infrastructure management and performance prediction.

This work involves the following main sub-objectives:

- According to the AI@EDGE defined architecture, model intelligent units or intelligent agents/multi-agents, semantic and syntactic representation models, action formats, as well as the language that describes them and their interfaces. Define intelligence performance-aware VNF placement algorithms to optimally locate agents according to certain parameters and constraints (about services, users, etc.)
- Model the exchange of actions between agents, centralized or distributed, autonomous or cooperative, and control of distributed agents.

- Simulate distributed AI on the AI@EDGE architecture optimizing some objective (ex. % of max. successful services flows) based on:
- Decentralized Distributed Deep Reinforcement Learning (RL) in a server-less architecture. In this solution, there is a need for optimal placement of VNFs acting as micro-clouds, deployed close to edge devices. These VNFs will be used to efficiently build DL/RL models at the edge, for faster DL/RL inference, and will support online and incremental learning, using continuously generated data from a large number of devices. The VNFs could be equipped with heterogeneous computing resources, and the VNFs connection and communication will be dynamic adapted to the services and applications requirements;
- Federated learning to train machine learning models in a distributed system. This is a server-less learning approach based on the cooperation of devices. The federated learning approach decentralizes training across devices dispersed across geography, and they collaboratively develop machine learning while keeping their personal data on their devices.

We consider server-less architectures, without needing to manage dedicated servers or instances for the services and applications. Instead, we need to optimally locate the required stateless management and control functions close to the edge for providing faster services with minimum latency. Hence, applying AI for the control and management will not need to migrate large amounts of data into centralized clouds, improving the cost (e.g., bandwidth), performance, and privacy.

4.2.4.2 State-of-the-art

VNF is an important technology in the development and implementation of 5G networks, it provides greater elasticity to scale to larger networks providing greater efficiency and lower resource consumption, reducing operating and consumption expenses and also improving time response in the different services, thanks to the virtualization of the different functions [MI16].

The VNF placement algorithms are a fundamental link to take advantage of the network resources optimally. In order for the location of the nodes to be carried out in a more efficient way, the best set of parameters must be selected that allows optimizing the process of location of the nodes. In [SAN18], an investigation is carried out on state-of-the-art and a set of parameters to be used in a Placement algorithm is proposed, which helps to perform an evaluation of the location process for 5G networks and Edge Computing efficiently.

In [SUB21], two deep learning models are proposed, one centralized and the other federated, to solve the problem of scaling the VNF. They perform a comparison of various deep learning models reviewing the advantages and disadvantages of FL over centralized learning algorithms, showing that federated learning has a lower performance than centralized learning for VNF autoscaling.

Another study by Wu et al. [WU20] introduces BVCP (Border VNF Chain Placement), a novel VNF placement method that consists of dividing the network into multiple subnets and using the border hypervisors features, to allow the algorithm to be scalable for big networks. The results of the work show that the BVCP method outperforms the baseline algorithm Dynamic Network Function (DNF) in VNF chain placement.

In the case of [CAO17], to solve the VNF location optimization problem, four genetic algorithms are proposed using the frameworks of two existing genetic algorithms, MOGA and NSGA II. The results show that the best algorithm is the Greedy-NSGA-II. Our work will use an approach based on [SUB21] that presents better results, more up-to-date algorithms, and satisfies the need to optimally locate agents according to certain defined parameters and restrictions.

Due to its operation consisting of training the models in the client devices, Federated learning can be the target of different attacks that can compromise the security and information of the end-user. To solve this problem, in [ISA20], end-user security is addressed, proposing a security mechanism that can be integrated into the 3GPP 5G Network Data Analytics (NWDA) architecture, and a multiparty computing protocol (MPC) to protect confidentiality is added. Although a network is never completely secure, this mechanism helps protect users' confidential information. In [LIU20], a secure FL framework based on blockchain is proposed to prevent the intrusion of malicious clients that can attack and affect the entire FL process. The general idea of this paper is that the Aggregation Server can recognize unreliable FL participants and discard or exclude them from participating in the training. They also use local differential privacy techniques to prevent membership inference attacks. The results of the work show that both poisoning attacks and membership inference attacks can be effectively mitigated.

4.2.4.3 Overview of the method

The proposed method is composed of a Central FL Server, responsible for the aggregation and updating of a Global ML model, which includes the collection of the updated local model's parameters from the Edge devices, the aggregation and updating of the Global model, and the sending of the Updated Global model to Edge devices to be used in the next iteration. The other components are Edge devices, the Edge AIFs are distributed by the Placement Algorithm and are responsible for the local update, which is the training process of the local models that uses the data of a certain number of clients devices to train the models. The main reason for using this model is the possibility that this Algorithm provides to distribute the learning task to the edges, and only transmit the updates of the locally trained model, without the need to send raw data to a Central Server.

4.2.4.4 Integration in the architecture

The proposed method has two types of AIFs, the server's AIF and the local clients' AIF. On one hand, the Server's AIF is composed of four blocks: Placement, Moderator, Policies, and Aggregator. The Placement role is defining the location of the VNFs according to certain criteria. The Moderator role is reviewing the requirements and specifications that customers must meet (in this block more functions can be added to satisfy the new algorithm functions). The Aggregator is in charge of sending the Global Model to the clients and then generating a new Global Model. The Policies block defines policies/directives to reach a certain level of model accuracy.

On the other hand, the client's AIFs are in charge of training the models in each client. Figure 35 shows three interfaces: IF1 is responsible for including the initial parameters of the previously trained Global Model and the policies/directives that the model must comply with to reach the desired level of accuracy to complete the training. Through the IF2 interface, the exchange of information between the Aggregation/Central Server and clients' devices is carried out, i.e., the parameters of the models are transmitted through this interface. IF3 is the data exchange interface, it carries out the exchange of clients' information such as connectivity, load level, computing resources, etc.

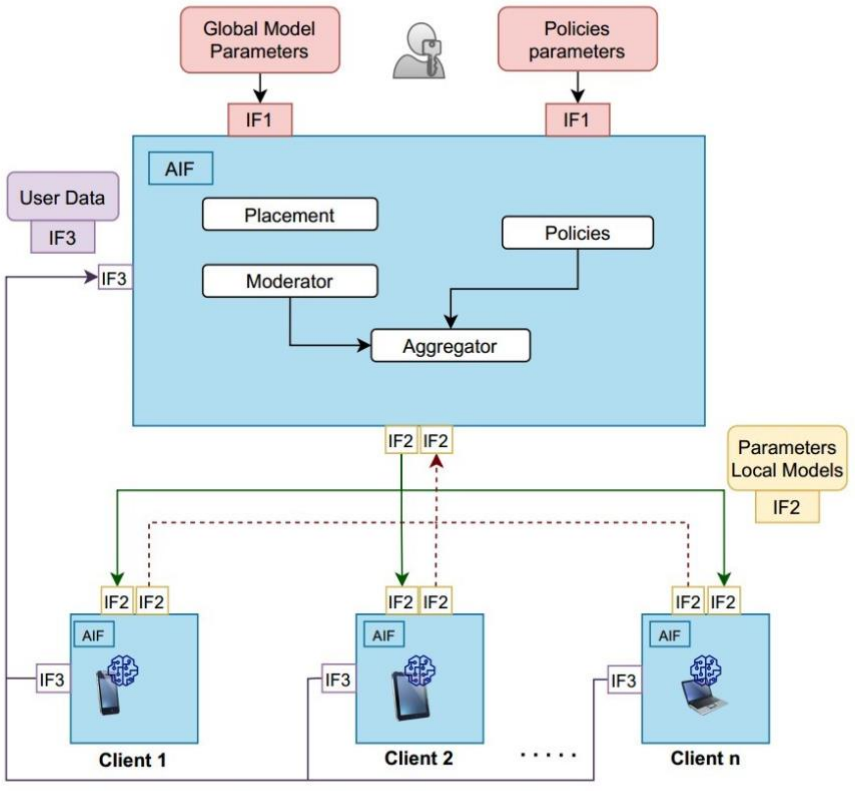

Figure 35 AIFs and interfaces of the proposed method

For each type of AIF, the interfaces perform different functions. In Server's AIF, the IF1 interface transmits the initial parameters of the Global Model that will be sent to the local devices to start the FL, defines and sends the policies/directives that the Global Model must comply with to reach the desired level of accuracy to complete the training. IF2 sends the Global Model's parameters to the local devices to start the training locally and receives updates from Local Models to update the Global Model. Interface IF3 receives and reviews the requirements that clients must meet to participate in the FL, such as connectivity, load level, computing resources, etc. In the AIF of the local clients' side, the interface IF2 receives the Global Model's parameters to start the training locally and sends Local Models updates to update the Global Model. IF3 sends the client information necessary to verify the requirements that clients must meet to participate in the FL.

4.2.5 *Data augmentation to increase the robustness of learning*

For decades, researchers and practitioners in networking have explored different ML techniques to assist network operations [Bou18]. A field that largely relies on ML is network security to detect attacks and anomalies [Dap15]. With the growing variety and complexity of attacks, using Machine Learning (ML) for network security is now unavoidable and commercial products in this sector also started to integrate Artificial Intelligence (AI) methods. However, like in other domains, among the obstacles against their large adoption in practice, we can mention the lack of these techniques to be able to generalize the behavior or attacks they have learned. Actually, the learning suffers from the lack of enough labeled data to represent the different and possibly infinite variation of attacks. To avoid this problem of over-fitting, different approaches exist. Among them, data augmentation consists into extending artificially the set of input data for learning in a realistic way. Although it is not specific to network security, the problem is exacerbated due to the presence of attackers that induce larger and unpredictable variations in data.

4.2.5.1 Objectives

When deploying an AIF or an application leveraging multiple AIF, an issue concerns how such a deployment fits to the current context and especially to the data to be analyzed. Two cases may occur:

- Case 1: the model was pre-trained on a dataset with different characteristics. So, the model is unable to infer correct information (supervised training). For network security, it can be simply a dataset that does not contain all attacks that an AIF module is supposed to detect when deployed.
- Case 2: the model is trained with representative data, but its performance is very low. In that case, the model and so the AIF was validated with another dataset and have proved good performance, but it is not efficient with data presenting new characteristics, e.g., underlying statistics about data distribution are different.

The problem is summarized in Figure 36. In the ideal case, the ML method has been trained with data that represents data which will be used for the inference in the second stage. However, in many cases, the data space is larger than what has been used for learning. It can be infinite but most of all even unknown. For example, we cannot always know the boundary of the data to be analyzed. In the worst case, the data given as input for the inference of the trained model are completely different. In that case, performance will be dramatically low by nature. The problem is more challenging as there are more dimensions over data.

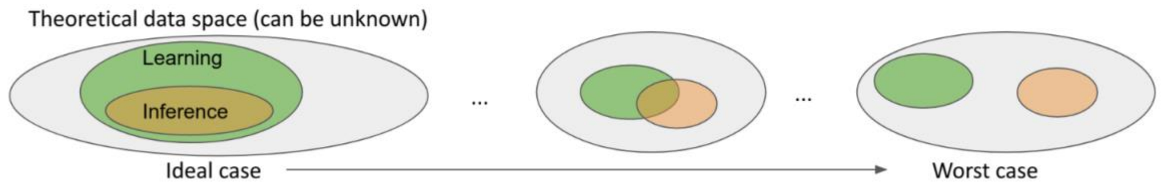

Figure 36 Obj-DataAugmentation: Summary of the problem of data representativeness in ML algorithms

In case 1, the objective is to construct datasets used for learning with a higher coverage of possible cases to support a higher generalization of an AIF. Obviously, our goal is not to create new types of attack in the context of security. This is a very challenging problem and it is out of the scope of our study. However, the goal is to generate data that could present a new configuration of a given attack in an original dataset. For example, our goal could be to artificially change the size of a botnet and automatically infer what the impact on the network traffic present in a dataset.

In case 2, the objective is also to construct or modify datasets but with different goals. In that case, our objective would be to automatically derive when an AIF and maybe a set of AIFs is valid to be deployed according to an expected accuracy. In a nutshell, the objective here is to identify what are the “borders” in the data space that split good and bad performance of an AIF. It thus supports the decision of the applicability of an AIF. Moreover, such a technique can help to choose among multiple AIFs fulfilling the same functionality, by evaluating each of them according to the condition of their deployment. The final objective here is to evaluate if data augmentation technique is relevant to evaluate the robustness of an attack detection mechanism based on a ML classifier and if this can be used to produce more robust classifiers. Concretely, it consists in extending datasets of network traffic containing attacks and evaluate the accuracy of the ML classifier with the newly generated data (with or without retraining).

4.2.5.2 *State-of-the-art*

In the last decade, the interest in data augmentation techniques has increased. It first focused on data augmentation techniques for image datasets to provide better accuracy to image classifiers. A new algorithm AutoAugment proposed in [Cub19] searches for best augmentation policies. Their method consists in two components: a search algorithm implemented as an RNN and a search space. A policy is composed in 5 sub-policies, each one has two operations to apply on a image. There are 16 operations, the magnitudes of these operations (translate, rotate...) are divided into 10 values and the probabilities to apply one in 11 values. It becomes a search problem with (16x10x11)^10 possibilities which is computationally expensive but offer better results than baseline models. The authors in [Ho19] improve AutoAugment technique and propose Population Based Augmentation (PBA). Their goal is to learn schedules of augmentation policy in contrast to AutoAugment which learn a fixed policy. PBA is based on Population Based Training (PBT) that optimizes the weights and hyperparameters of the neural network, copying the weights of top performers and muting the hyperparameters during training. Their technique offers comparable results than AutoAugment but requires 1.000x less GPU hours to produce it.

Another improvement is proposed by the research done in [Cub20] by reducing the search space. They found that in PBA the optimal magnitude of augmentation increases during the training, RandAugment does not search optimal magnitudes but have a fixed magnitudes schedule. Their algorithm contains two human-interpretable parameters: N (number of transformation) and M (magnitudes) and they use a naive grid search for hyperparameters optimization which offers similar results than PBA and AutoAugment this time with a reduced search space.

More recently, some studies have been focused on data generation. Particularly on unbalanced datasets to generate synthetic instances belonging to minority classes. With the emergence of deep learning, new techniques emerge. Generative Adversarial Networks (GAN) is proposed in [Goo14]. It consists in two neural networks defined as two multi-layer perceptrons, a discriminator D and a generator G playing a two-player minimax game. Where G try to generate new samples and D try to predict if the data generated are real or fake. The goal of G is to maximize the probability of D of doing mistakes. Image classification is not the only domain of research, recently with the emergence of new technologies, the industry 4.0, the spread of IoT, etc. Network security becomes more and more challenging. In particular, network anomaly detection gain in interest but network datasets are unbalanced where the anomalies are lost in normal samples. The authors in [Ola20] show that a naive adoption of GAN does not work well for anomaly detection and propose Divide Augment Combine (DAC). Samples are grouped on their unique characteristics, then augmented on a group basis using GAN and the extended data are fed into a classifier to produce a learning model. They propose two grouping methods: by cluster and by attack, the first one seems to be better and offers significant results.

As highlighted, GAN [Goo14] has gained a large interest to produce robust models by enabling automated generation of data but are prone to the mode collapse problem. It results in producing artificial data with low variations limiting the value of the newly generated data. Alternative approaches can rely on applying a well selected, or learned, sequence of data transformations [Cub20] also named as data augmentation policy. Our technique will be inspired from the latter and others techniques existing in this field [Cub19,Ho19]

4.2.5.3 *Overview of the method*

From a general perspective, a data augmentation technique relies on the following steps:

1 Definition of transformations to be applied on the dataset
2 Definition of the policy
2.1 Number of operations to be applied
2.2 Selection of operations to be applied (the order of the sequence is important as operations can be successively applied to the same data)

  2.3 Definition of the hyper-parameters of the operations
3 Validation of the data augmentation policy

Actually, step 2 can be statically and manually defined, generated dynamically or learned to optimize the generation of the parameters. Indeed, using a validation set, the augmentation policy can be evaluated and considered as effective or not. Different optimization techniques can be used for this task including a grid search over the different operations and their parameters assuming the number of operations is human defined.

Most existing techniques focus on image processing and rely on corresponding transformation: rotation, change color, change contrast or translate. To the best of our knowledge, no work exists in the area of network processing. Obviously, image operations are meaningless for the type of data we will rely on, mostly network capture contains anomalies.

Defining relevant operations is thus critical. Once done, we can rely on off-the-shelf techniques and tools cited above. Most features are or can be transposed numerically but a blind approach applying random numeric operations will not be cost-effective and would require a large amount of time to find a suitable policy. Similarly, for image processing, the operations have not been randomly defined.

To identify these relevant operations, we define the following method:

- List all features used in relevant and accessible datasets focusing first on UBN datasets (https://www.unb.ca/cic/datasets/)
- For each feature
    - Characterize its type (numeric, categorical, boolean)
    - Identify the boundaries (min, max, set of values)
    - Identify if dependencies exist among the features
    - For each type of feature
        - Define the type of applicable operations (addition, select of value in a set…)
        - Evaluate a priori the impact of the operations and their parameters and define limitations, for example based on the identified boundaries (do not apply an operation that lead to a value which will never be observed in reality, for example the MTU limits the size of network packet)
        - For dependent features, model the dependency and its impact on the operations to be jointly applied.

4.2.5.4 *Integration in the architecture*

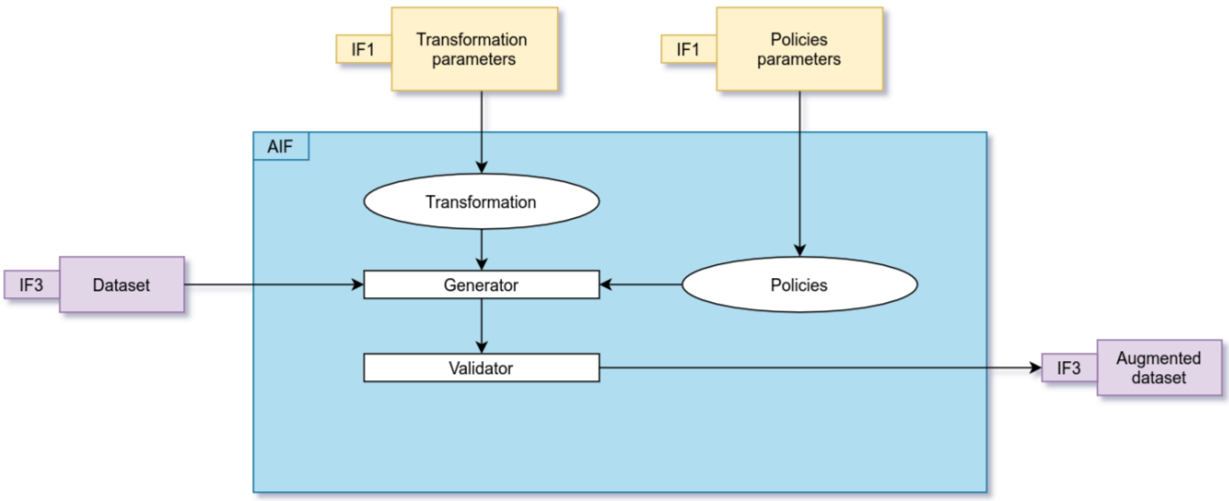

Figure 37 Overview of a dataset generation AIF

In Figure 37, our technique is structured as a single AIF with three internal components:

- **Transformation**: it takes as input a set of transformations parameters to be applied on generation function. For example, if we want to generate a new set of images, the different applicable transformations can be rotation of the image, change the brightness of the image, or change some colors. The **transformation** task will keep in memory the different transformations.
- **Policies**: initially, the **policies** task will be statically and manually defined. It will define how to apply the transformations (“Random”, “Each x lines”, “Only BW”), the different applied transformations (“Rotation”, “Brightness”, “Color”), the order of those transformations or eventually some hyper-parameters (e.g., how many rotations or degrees an image will be rotated in the case of the Rotation transformation)
- **Generator**: the generator takes the pre-configured transformations from the **transformation** task and apply them according to user-defined policies from the **policies** task**.**
- **Validator:** Once the new dataset is generated, the validator will take the new one as input and will verify the generation efficiency. This data will be sent through an interface that will be fully defined in the next stages of the project.

The AIF can be used as an action by the orchestrator in the closed-loop in order to generate a new dataset to assess a priori the capacity of another AI-based function to be generalized. The AIF aims to transform a dataset into a new dataset with generated data into it. It takes some different inputs on each interface as follows:

- IF1: Parameters for the transformation feature
- IF1: Parameters for the policies features
- IF3: The AIF takes a dataset for the data augmentation
- IF3: Augmented dataset in output

4.2.5.5 *Preliminary results*

The dataset used for this research is the UNSW-NB15 dataset. It was created by the university of new south wales. They used the IXIA PerfectStorm tool to create a mix of the modern normal and abnormal network traffic and simulate nine different types of attacks. They captured traffic in the form of packet with the tcpdump tool. They simulated the traffic during two days: 16 hours on the 22nd of January 2015 and 15 hours on the 17th of February 2015 and they captured 100 GBs of data. Then, they splat the pcap files into 1000 MB smaller ones. Finally, they used Argus and Bro-ids tools and twelve different algorithms to analyze the flow of the connection packets in order to create reliable features. To represent real life flows, the dataset is very unbalanced. Table 3 shows the number of samples in each category and the percentage of each class for the entire dataset.

Table 3 Distribution of classes into the dataset

Classes	Number of records	Percentage of the dataset
Normal	2 218 761	87.35
Generic	215 481	8.48
Exploits	44 525	1.75
Fuzzers	24 246	0.95
DoS	16 353	0.64
Reconnaissance	13 987	0.55
Analysis	2 677	0.11
Backdoors	2 329	0.09
Shellcode	1 511	0.07
Worms	174	0.01

To counter the problem of unbalanced dataset, a technique of generation has been explored.

We used the GAN framework and especially the DAC (Divide Augment Combine) technique explained in the State-of-the-Art section. DAC was used for the multiclass generation, meaning that for each attack, a GAN model was trained according to the distribution of each attack. For the binary, the normal and attack samples were generated together. As the GAN generated float values to express the label of both categories, a threshold of 0.5 was used, each sample with a value label above this threshold was set to 1 as for the anomalies and under to 0 for the normal ones.

The results of the classification before and after the augmentation are reported in Table 4 Results of the classification before and after the .

Table 4 Results of the classification before and after the augmentation

	(a) Binary classification				(b) Multiclass classification			
	Acc.	Pre.	Rec.	F1-sc.	Acc.	Pre.	Rec.	F1-sc.
Original	99.42	99.02	98.34	98.68	98.12	78.86	52.67	53.94
Augmented	99.42	99.01	98.34	98.98	98.12	78.86	52.78	54.02

The purpose is different whether the classification is binary or multiclass. In the first classification, we aim to know if there is an attack on the network or not independent of the type of the attack. In the second classification, we aim to know exactly which attack is used among the abnormal data. The accuracy for the multiclass classification on unbalanced data is not enough to represent the effectiveness of the model. The precision, the recall and the F1-score are used to have a better understanding as it is possible to express these metrics tacking into account the unbalanced data. The fact that the dataset is really unbalanced makes the training more difficult. The train will focus on the majority classes and almost ignore the minority ones. For this reason, the last two metrics are not as high as the accuracy. A solution to overcome this problem could be re-balance the dataset using the GAN. Unfortunately, the GAN used here does not improve the results, a reason could be that the data are too complicated to generalize. The GAN learn the distribution of the training set only (not with the test set). The GAN manages to create a distribution that look like real, but the generated sample seems to be not used for the test. The training scores increase, but the test does not. The main reason could be that the distribution of the fake sample is too different compared to the test one as the GAN is trained on the training part of the dataset. The generated samples act like a noise and are not used.

5. Data sources

In this section, we report on raw data available to AI/ML algorithms. When applicable, we also describe how derived data can be created starting from the raw data.

5.1 Container-level data

Due to their ephemeral nature, containers are rather complex and more challenging to monitor compared to traditional applications. In the state-of-the-art, several tools exist to collect and store containers runtime metrics, and CAdvisor (Container Advisor) [CAdvisor] is one of the most used. It collects resource isolation parameters, historical resource usage, histograms of complete historical resource usage and network statistics. It is natively embedded into Kubernetes, where it exposes metrics in the standard Prometheus format. Table 5 summarizes the metrics collected by Kubernetes embedded CAdvisor used in the SYRROCA framework (Section 4.2.1). It is worth noticing that the collection rate can be tuned.

Table 5 CAdvisor raw data information

name	collection rate	type	Cumulative/ Instant
timestamp	5s	string	I
container_cpu_load_average_10s	5s	string	I
container_cpu_load_average_10s	5s	string	I
container_cpu_system_seconds_total	5s	string	C
container_cpu_usage_seconds_tota	5s	string	C
container_cpu_user_seconds_total	5s	string	C
container_fs_inodes_free	5s	string	I
container_fs_inodes_total	5s	string	I
container_fs_io_current	5s	string	I
container_fs_io_time_seconds_total	5s	string	C
container_fs_io_time_weighted_seconds_total	5s	string	C
container_fs_limit_bytes	5s	string	I
container_fs_reads_bytes_total	5s	string	C
container_fs_read_seconds_total	5s	string	C
container_fs_reads_merged_total	5s	string	C
container_fs_reads_total	5s	string	C
container_fs_sector_reads_total	5s	string	C
container_fs_sector_writes_total	5s	string	C

container_fs_usage_bytes	5s	string	I
container_fs_writes_bytes_total	5s	string	C
container_fs_write_seconds_total	5s	string	C
container_fs_writes_merged_total	5s	string	C
container_fs_writes_total	5s	string	C
container_last_seen	5s	string	I
container_memory_cache	5s	string	I
container_memory_failcnt	5s	string	C
container_memory_failures_total	5s	string	C
container_memory_mapped_file	5s	string	I
container_memory_max_usage_bytes	5s	string	I
container_memory_rss	5s	string	I
container_memory_swap	5s	string	I
container_memory_usage_bytes	5s	string	I
container_memory_working_set_bytes	5s	string	I
container_network_receive_bytes_total	5s	string	C
container_network_receive_errors_total	5s	string	C
container_network_receive_packets_dropped_total	5s	string	C
container_network_receive_packets_total	5s	string	C
container_network_transmit_bytes_total	5s	string	C
container_network_transmit_errors_total	5s	string	C
container_network_transmit_packets_dropped_total	5s	string	C
container_network_transmit_packets_total	5s	string	C
container_spec_cpu_period	5s	string	I
container_spec_cpu_shares	5s	string	I
container_spec_memory_limit_bytes	5s	string	I
container_spec_memory_reservation_limit_bytes	5s	string	I
container_spec_memory_swap_limit_bytes	5s	string	I
container_start_time_seconds	5s	string	I

container_tasks_state	5s	string	I

5.2 Physical server-level data

Unix-like systems provide several means for collecting hardware and OS metrics. Among them, Prometheus NodeExporter [NodeExporter] provides a complete and extensible mean to retrieve physical server metrics. As shown in Table 6, default collected metrics span the CPU, the memory, the disk, and the network. As for the virtual level metrics, the collection rate is tunable.

Table 6 NodeExporter raw data information

name	collection rate	type	cumulative/instant
timestamp	5s	string	I
node_cpu_seconds_total_mode_idle	5s	string	C
node_cpu_seconds_total_mode_iowait	5s	string	C
node_cpu_seconds_total_mode_irq	5s	string	C
node_cpu_seconds_total_mode_nice	5s	string	C
node_cpu_seconds_total_mode_softirq	5s	string	C
node_cpu_seconds_total_mode_steal	5s	string	C
node_cpu_seconds_total_mode_system	5s	string	C
node_cpu_seconds_total_mode_user	5s	string	C
node_cpu_core_throttles_total	5s	string	C
node_cpu_package_throttles_total	5s	string	C
node_cpu_frequency_max_hertz	5s	string	I
node_cpu_frequency_min_hertz	5s	string	I
node_cpu_guest_seconds_total_mode_nice	5s	string	C
node_cpu_guest_seconds_total_mode_user	5s	string	C
node_cpu_scaling_frequency_hertz	5s	string	I
node_cpu_scaling_frequency_max_hrts	5s	string	I
node_cpu_scaling_frequency_min_hrts	5s	string	I
node_context_switches_total	5s	string	C
node_disk_io_now	5s	string	I
node_disk_io_time_seconds_total	5s	string	C
node_disk_io_time_weighted_seconds_tota	5s	string	C

node_disk_read_bytes_total	5s	string	C
node_disk_read_time_seconds_total	5s	string	C
node_disk_reads_completed_total	5s	string	C
node_disk_reads_merged_total	5s	string	C
node_disk_write_time_seconds_total	5s	string	C
node_disk_writes_completed_total	5s	string	C
node_disk_writes_merged_total	5s	string	C
node_disk_written_bytes_total	5s	string	C
node_boot_time_seconds	5s	string	I
node_arp_entries	5s	string	I
node_memory_Active_anon_bytes	5s	string	I
node_memory_AnonHugePages_bytes	5s	string	I
node_memory_AnonPages_bytes	5s	string	I
node_memory_Bounce_bytes	5s	string	I
node_memory_Buffers_bytes	5s	string	I
node_memory_Cached_bytes	5s	string	I
node_memory_CmaFree_bytes	5s	string	I
node_memory_CmaTotal_bytes	5s	string	I
node_memory_CommitLimit_bytes	5s	string	I
node_memory_Committed_AS_bytes	5s	string	I
node_memory_DirectMap1G_bytes	5s	string	I
node_memory_Dirty_bytes	5s	string	I
node_memory_HardwareCorrupted_bytes	5s	string	I
node_memory_HugePages_Free	5s	string	I
node_memory_HugePages_Rsvd	5s	string	I
node_memory_HugePages_Surp	5s	string	I
node_memory_HugePages_Total	5s	string	I
node_memory_Hugepagesize_bytes	5s	string	I
node_memory_Inactive_anon_bytes	5s	string	I
node_memory_KernelStack_bytes	5s	string	I
node_memory_Mapped_bytes	5s	string	I

node_memory_MemAvailable_bytes	5s	string	I
node_memory_MemFree_bytes	5s	string	I
node_memory_MemTotal_bytes	5s	string	I
node_memory_Mlocked_bytes	5s	string	I
node_memory_NFS_Unstable_bytes	5s	string	I
node_memory_PageTables_bytes	5s	string	I
node_memory_SUnreclaim_bytes	5s	string	I
node_memory_SwapCached_bytes	5s	string	I
node_memory_SwapFree_bytes	5s	string	I
node_memory_SwapTotal_bytes	5s	string	I
node_memory_Unevictable_bytes	5s	string	I
node_memory_VmallocChunk_bytes	5s	string	I
node_memory_VmallocTotal_bytes	5s	string	I
node_memory_VmallocUsed_bytes	5s	string	I
node_memory_WritebackTmp_bytes	5s	string	I
node_memory_Writeback_bytes	5s	string	I
node_netstat_Icmp6_InErrors	5s	string	I
node_netstat_Icmp6_InMsgs	5s	string	I
node_netstat_Icmp_InErrors	5s	string	I
node_netstat_Icmp_InMsgs	5s	string	I
node_netstat_Icmp_OutMsgs	5s	string	I
node_netstat_Ip6_InOctets	5s	string	I
node_netstat_Ip6_OutOctets	5s	string	I
node_netstat_IpExt_InOctets	5s	string	I
node_netstat_IpExt_OutOctets	5s	string	I
node_netstat_Ip_Forwarding	5s	string	I
node_netstat_TcpExt_ListenDrops	5s	string	I
node_netstat_TcpExt_ListenOverflows	5s	string	I
node_netstat_TcpExt_SyncookiesFailed	5s	string	I
node_netstat_TcpExt_SyncookiesRecv	5s	string	I
node_netstat_TcpExt_SyncookiesSent	5s	string	I

node_netstat_TcpExt_TCPSynRetrans	5s	string	I
node_netstat_Tcp_ActiveOpens	5s	string	I
node_netstat_Tcp_CurrEstab	5s	string	I
node_netstat_Tcp_InErrs	5s	string	I
node_netstat_Tcp_InSegs	5s	string	I
node_netstat_Tcp_OutSegs	5s	string	I
node_netstat_Tcp_PassiveOpens	5s	string	I
node_netstat_Tcp_RetransSegs	5s	string	I
node_netstat_Udp6_InErrors	5s	string	I
node_netstat_Udp6_NoPorts	5s	string	I
node_netstat_UdpLite6_InErrors	5s	string	I
node_netstat_UdpLite_InErrors	5s	string	I
node_netstat_Udp_InDatagrams	5s	string	I
node_netstat_Udp_InErrors	5s	string	I
node_netstat_Udp_NoPorts	5s	string	I
node_netstat_Udp_OutDatagrams	5s	string	I
node_network_receive_bytes_total	5s	string	C
node_network_receive_compressed_total	5s	string	C
node_network_receive_drop_total	5s	string	C
node_network_receive_errs_total	5s	string	C
node_network_receive_fifo_total	5s	string	C
node_network_receive_frame_total	5s	string	C
node_network_receive_multicast_total	5s	string	C
node_network_receive_packets_total	5s	string	C
node_network_speed_bytes	5s	string	I
node_network_transmit_bytes_total	5s	string	C
node_network_transmit_carrier_total	5s	string	C
node_network_transmit_colls_total	5s	string	C
node_network_transmit_compressed_total	5s	string	C
node_network_transmit_drop_total	5s	string	C
node_network_transmit_errs_total	5s	string	C

node_network_transmit_fifo_total	5s	string	C
node_network_transmit_packets_total	5s	string	C
node_network_transmit_queue_length	5s	string	I
node_sockstat_FRAG_inuse	5s	string	I
node_sockstat_FRAG_memory	5s	string	I
node_sockstat_RAW_inuse	5s	string	I
node_sockstat_TCP_alloc	5s	string	I
node_sockstat_TCP_inuse	5s	string	I
node_sockstat_TCP_mem	5s	string	I
node_sockstat_TCP_mem_bytes	5s	string	I
node_sockstat_TCP_orphan	5s	string	I
node_sockstat_TCP_tw	5s	string	I
node_sockstat_UDPLITE_inuse	5s	string	I
node_sockstat_UDP_mem	5s	string	I
node_sockstat_UDP_mem_bytes	5s	string	I
node_sockstat_sockets_used	5s	string	I

5.3 eNodeB and UE level data

To get an overall view of the state of a virtualized service, besides monitoring the state of the virtualization infrastructure, it is advisable to collect metrics regarding the state of the service itself. As also recommended by ETSI, most native cloud network services expose some metrics by default. This is the case, for example, of the srsRAN [srsRAN] software used in the simulations performed on the SYRROCA framework when applied to the use case of a 5G core. Table 7 and Table 8summarizes the metrics exposed by the srsRAN eNB simulator and by the srsUE User End simulator, respectively. The default metrics collection rate is 1 second in both cases, but the rate is adjustable. Furthermore, metrics are provided on a per-UE basis for the downlink (DL) and uplink (UL), respectively.

Table 7 SRS eNodeB raw data information

name	collection rate	type	cumulative/instant
Radio Network Temporary Identifier (UE identifier)	1s	string	I
Channel Quality Indicator reported by the UE (1-15)	1s	string	I
Rank Indicator reported by the UE (dB)	1s	string	I
Modulation and coding scheme (0-28)	1s	string	I
Bitrate (bits/sec)	1s	string	I
Number of packets succesfully sent	1s	string	C

Number of packets dropped	1s	string	C
% of packets dropped	1s	string	C
PUSCH SNIR (Signal-to-Interference-plus-Noise Ratio)	1s	string	I
PUCCH SNIR	1s	string	I
Power Headroom (dB)	1s	string	I
Buffer Status Report - data waiting to be transmitted as reported by the UE (bytes)	1s	string	I
Cpu	1s	string	I
Memory	1s	string	I

** The latest version of srsRAN (21.10 with an initial version of gNodeB) outputs the same data.

Table 8 srsUE raw data information

name	collection rate	type	cumulative/instant
Component carrier	1s	string	I
Reference Signal Receive Power (dBm)	1s	string	I
Pathloss (dB)	1s	string	I
Carrier Frequency Offset (Hz)	1s	string	I
Modulation and coding scheme (0-28)	1s	string	I
Signal-to-Noise Ratio (dB)	1s	string	I
Average number of turbo decoder iterations	1s	string	I
Bitrate	1s	string	I
Block error rate	1s	string	I
Timing advance (uS)	1s	string	I
Uplink buffer status - data waiting to be transmitted (bytes)	1s	string	I
Cpu	1s	string	I
Memory	1s	string	I
Cc	1s	string	I

5.4 WiFi AP-level data

5G-EmPower virtualizes Wi-fi access points and offer different QoS to the users. Table 9 summarizes the list of raw collected metrics from Access points. The collection rate is tunable.

Table 9 5G-EmPOWER Access Point raw data information

name	collection rate	type	cumulative/instant
Neighboring Wi-Fi Stations RSSI	Arbitrary (Min 100ms)	integer	I
Neighboring Wi-Fi Access Points RSSI	Arbitrary (Min 100ms)	integer	I
Transmitted bytes/packets	Arbitrary (Min 100ms)	integer	C
Received bytes/packets	Arbitrary (Min 100ms)	integer	C
Downlink goodput	Arbitrary (Min 100ms)	float	I
Number of transmitted frames (including failures)	Arbitrary (Min 100ms)	integer	C
Number of successfully transmitted frames	Arbitrary (Min 100ms)	integer	C

5.5 Application server-level data

These data sources are application-specific and listed for each application that the connect-compute platform supports. Table 10 lists the data sources from a video streaming application using the DASH protocol.

Table 10 Video streaming application data sources

Name	Collection rate	Type	Cumulative/Instant
#video segment requests sent out in the previous window	Configurable (2-16s)	integer	Cumulative
Bitrate of the previous segment requested	Event triggered	integer	Instant
Throughput over the last downloaded segment	Event triggered		Instant
Throughput over the last 10 s	10s	integer	Cumulative
Bytes in the video playback buffer	Configurable (2-16s)	integer	Instant
#video segments in the playback buffer	Configurable (2-16s)	integer	Instant

5.6 Network performance real-time monitoring

Table 11 lists KPI data that can be collected from the Athonet 5G core network in Prometheus format. Prometheus (https://prometheus.io/) is an open-source systems monitoring and alerting toolkit and the core exposes the unaltered upstream Prometheus API. The following table shows a brief description of the KPI in 3GPP terminology or of the alert, its type, and a reference to the 3GPP technical specification document (if any).

Table 11 Athonet core network KPI data

Description of the collectable KPI or alert	Type	3GPP specification reference
Number of UPF sessions	KPI	NA
Number of GTP-U (N3) interfaces	KPI	TS 28.552 5.4.1
Number of UPF IP (N6) interfaces	KPI	TS 28.552 5.4.2
Number of GTP-U (N3) packets	KPI	TS 28.552 5.4.1
Number of UPF IP (N6) packets	KPI	TS 28.552 5.4.2
Number of UPF PFCP (N4) messages	KPI	NA
Number of Users in the SMF	KPI	TS 28.552 5.3.1
Number of Active Session in the SMF	KPI	TS 28.552 5.3.1
Number of DNNs supported by the SMF	KPI	TS 28.552 5.3.1
NGAP RRC Establishment Causes	KPI	TS 28.552 5.2.2
Number of AMF NGAP messages received	KPI	TS 28.552 5.2.2
Number of devices registered in the AMF	KPI	TS 28.552 5.2.2
Number of active device connections in the AMF	KPI	TS 28.552 5.2.2
RAN Nodes status	KPI	TS 28.552 5.2.2
Overall CPU usage above 50% for more than a minute	Alert	NA
Memory usage above 70%	Alert	NA
Memory usage above 90%	Alert	NA
Disk usage above 70%	Alert	NA
Disk usage above 90%	Alert	NA

6. Conclusions

This deliverable reported on the achievements related to the development of systems and methods for the automation of the AI@EDGE connect-compute platform. Initial descriptions of the internal design of the Network and Service Automation Platform (NSAP) have been provided. The NSAP is a framework for automation of network management that provides an environment for data-driven methods supporting decision making.

The design of the NSAP embraces the concept of closed-loop control, where decision-making is automated based on data-driven AI/ML-based methods and algorithms. The overall goal is to minimise the need for the human operator to analyse the current state, take decisions, and to implement corrective actions. The data-driven methods and algorithms are supported by monitoring data and other system data in an efficient and consistent manner through a set of common data pipelines.

An initial collection of methods and algorithms for automation and learning for network management purposes are described with some preliminary results. The methods and algorithms can have different roles, for example, prediction or estimation of system parameters in the near future, which could be part of a data pipeline, network management decision-making, for example of service placement or resource allocation, or supporting the needs of an application service to meet its service requirements.

The achievements reported in the deliverable are the progress towards the overall project Objective 3 on designing and implementing a general-purpose network automation framework, capable of supporting flexible and reusable pipelines for the end-to-end creation, utilisation, and adaptation of secure and privacy-preserving AI/ML models.

Bibliography

[3GPP5GS] 3GPP, 5G, System architecture for the 5G System (5GS), Tech. Rep. 2020.

[ADEOGUN] R. O. Adeogun, P. D. Teal, and P. A. Dmochowski, ``Extrapolation of MIMO mobile-to-mobile wireless channels using parametric-model-based prediction," IEEE Trans. Veh. Technol., vol. 64, no. 10, pp. 4487-4498, Oct. 2015.

[Alawe18] Alawe I, Ksentini A, Hadjadj-Aoul Y, Bertin P. Improving traffic forecasting for 5G core network scalability: A machine learning approach. IEEE Network. 2018 Nov 29;32(6):42-9.

[An2015] An J, Cho S. Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE. 2015 Dec 27;2(1):1-8.

[ANGUS] A. Angus, C. V. Murudkar, and R. D. Gitlin, "Machine learning for qoe prediction and anomaly detection in self-organizing mobile net-working systems, "International Journal of Wireless & Mobile Networks (IJWMN) Vol, vol. 11, 2019.

[Bad20] H. Badri, T. Bahreini, D. Grosu and K. Yang, "Energy-Aware Application Placement in Mobile Edge Computing: A Stochastic Optimization Approach," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 4, pp. 909-922, 1 April 2020, doi: 10.1109/TPDS.2019.2950937.

[Balasubramanian2021] B. Balasubramanian, E. Scott Daniels, M. Hiltunen, R. Jana, K. Joshi, T., X. Tran, and C. Wan, RIC: A RAN Intelligent Controller Platform for AI-Enabled Cellular Networks. AI-POWERED 5G SERVICES

[Beh19] R. Behravesh, E. Coronado, D. Harutyunyan and R. Riggio, "Joint User Association and VNF Placement for Latency Sensitive Applications in 5G Networks," Proc. of CloudNet, 2019, doi: 10.1109/CloudNet47604.2019.9064145.

[BEH20] Behravesh, Rasoul, et al. "ML-Driven DASH Content Pre-Fetching in MEC-Enabled Mobile Networks." 2020 16th International Conference on Network and Service Management (CNSM). IEEE, 2020.

[BEH21] Behravesh, Rasoul, et al. "Time-Sensitive Mobile User Association and SFC Placement in MEC-Enabled 5G Networks." IEEE Transactions on Network and Service Management (2021).

[BEN94] Y. Bengio, P. Simard, and P. Frasconi. Learning longterm dependencies with gradient descent is difficult. IEEE Trans. on Neural Networks 15.2 (1994): 157-166.

[Bengio'21] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," European Journal of Operational Research, vol. 290, no. 2, pp. 405–421, apr 2021.

[Boasman-Patel]. Boasman-Patel, A., Sun, D., Wang, Y., Maitre, C., References Autonomous Network: Empowering Digital Transformation for The Telecom Industry. Tratto da https://www.tmforum.org/wp-content/uploads/2019/05/22553-Autonomous-Networks-whitepaper.pdf

[BOLD] M. Boldt, S. Ickin, A. Borg, V. Kulyk, and J. Gustafsson, "Alarm prediction in cellular base stations using data-driven methods," IEEE Transactions on Network and Service Management, vol. 18, no. 2, pp. 1925–1933,2021.

[Bou18] Boutaba, R., Salahuddin, M.A., Limam, N. et al. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. J Internet Serv Appl 9, 16 (2018). https://doi.org/10.1186/s13174-018-0087-2

[Bra09] Brauckhoff D, Salamatian K, May M. Applying PCA for traffic anomaly detection: Problems and solutions. InIEEE INFOCOM 2009 2009 Apr 19 (pp. 2866-2870). IEEE.

[CAdvisor] https://github.com/google/cadvisor

[CAO17] J. Cao, Y. Zhang, W. An, X. Chen, J. Sun, y Y. Han, «VNF-FG design and VNF placement for 5G mobile networks», Sci. China Inf. Sci., vol. 60, n.o 4, p. 040302, abr. 2017, doi: 10.1007/s11432-016-9031-x.

[CARPIO] Carpio, Francisco, Admela Jukan, and Rastin Pries. "Balancing the migration of virtual network functions with replications in data centers." NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2018.

[CAR2006] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in Proc. of ACM ICMLC, New York, NY, USA, 2006

[CHEN2016] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. of ACM SIGKDD, New York, NY, USA, 2016

[Chen18] Chen, C., Liu, Z. “Broad Learning System: An Effective and Efficient Incremental Learning System Without the Need for Deep Architecture” IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 29, NO. 1, JANUARY 2018

[CHO] Cho, Daewoong, et al. "Real-time virtual network function (VNF) migration toward low network latency in cloud environments." 2017 IEEE 10th International Conference on Cloud Computing (CLOUD). IEEE, 2017.

[Cor21] E. Coronado, F. Raviglione, M. Malinverno, C. Casetti, A. Cantarero, G. Cebrián-Márquez, R. Riggio, "ONIX: Open Radio Network Information eXchange," in IEEE Communications Magazine, [In Press].

[Crosbie95] Crosbie M, Spafford G. Applying genetic programming to intrusion detection. In Working Notes for the AAAI Symposium on Genetic Programming 1995 Nov 10 (pp. 1-8). Cambridge, MA: MIT Press.

[Cub19] Ekin D Cubuk et al. “Autoaugment: Learning augmentation strategies from data”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, pp. 113–123.

[Cub20] E. D. Cubuk, B. Zoph, J. Shlens and Q. V. Le RandAugment : Practical automated data augmentation with a reduced search space 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)

[D2.1] Secci, S. (2021). D2.1 Use cases, requirements, and preliminary system architecture. AI@EDGE Deliverable 2.1 (H2020-ICT-52-2020).

[D4.1] Enric Pages, Javier Melian (eds.) (2021). D4.1 – Design and initial prototype of the AI@EDGE connect-compute platform. AI@EDGE Deliverable 4.1 (H2020-ICT-52-2020).

[Dai’17] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, “Learning Combinatorial Optimization Algorithms over Graphs,” in Advances in Neural Information Processing Systems, vol. 2017-Decem. Neural information processing systems foundation, apr 2017, pp. 6349–6359.

[Dap15] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng. 2015. Opprentice : Towards Practical and Automatic Anomaly Detection Through Machine Learning. Proceedings of the 2015 ACM Internet Measurement Conference (IMC ’15).

[Defferrard’16] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering,” in Proc. Advances in Neural Inf. Process. Syst. (NeurIPS). NeurIPS, jun 2016, pp. 3844–3852.

[DIA20] Alessio Diamanti, José Manuel Sanchez Vilchez, Stefano Secci. LSTM-based radiography for anomaly detection in softwarized infrastructures. International Teletraffic Congress, IEEE, Sep 2020, Osaka, Japan. hal-02917660.

[DJGIT] “An MPEG/DASH client-server ns3 module,” Accessed on 23.07.2020 [Online]. Available: https://github.com/djvergad/dash

[Due07] A. Duel-Hallen, “Fading channel prediction for mobile radio adaptive transmission systems,” Proceedings of the IEEE, vol. 95, no. 12, pp. 2299–2313, Dec. 2007.

[Emu20] M. Emu and S. Choudhury, "IoT Ecosystem on Exploiting Dynamic VNF Orchestration and Service Chaining: AI to the Rescue?," in IEEE Internet of Things Magazine, vol. 3, no. 4, pp. 30-35, December 2020, doi: 10.1109/IOTM.0001.2000012.

[Ergen19] Ergen T, Kozat SS. Unsupervised anomaly detection with LSTM neural networks. IEEE transactions on neural networks and learning systems. 2019 Sep 13;31(8):3127-41.

[ETS19] ETSI GS MEC 028 V2.1.1, “Multi-access Edge Computing (MEC); Radio Network Information API”, December, 2019.

[ETSIMEC] ETSI, GRMEC. Mobile edge computing (mec); deployment of mobile edge computing in an nfv environment,”. Tech. Rep., 2018.

[ETSINFV2014] ETSI, GS, Network Functions Virtualisation (NFV), Architectural Framework}, Tech. Rep., 2014.

[ETSINFV2017] ETSI, GR, Network Functions Virtualisation (NFV), Use Cases, Tech. Rep., 2017.

[ETSI-ZSM] Zero touch network & Service Management (ZSM). Tratto da https://www.etsi.org/technologies/zero-touch-network-service-management

[FLU21] Flutd: Build your unified logging layer. Link: https://www.fluentd.org/. Accessed: 2021-12-21

[Fu09] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. 2009. Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis. In Proc. of International Conference on Data Mining (ICDM). 149–158.

[GARDNER] W. Gardner, “Simplification of MUSIC and ESPRIT by exploitation of cyclostationarity,” Proceedings of the IEEE, vol. 76, no. 7, pp. 845–847, Jul. 1988.

[George12] George A, Vidyapeetham AV. Anomaly detection based on machine learning: dimensionality reduction using PCA and classification using SVM. International Journal of Computer Applications. 2012 Jun;47(21):5-8.

[Gilmer’17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural Message Passing for Quantum Chemistry,” 34th International Conference on Machine Learning, ICML 2017, vol. 3, pp. 2053–2070, apr 2017.

[Goo14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville et Yoshua Bengio Generative Adversarial Networks Advances in Neural Information Processing Systems 27, 2014

[GR ZSM 009-1]. Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers. Tratto da https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/00901/01.01.01_60/gs_ZSM00901v010101p.pdf

[GR ZSM 009-2] Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 2: Solutions for automation of E2E service and network management use cases. Tratto da https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=58055

[GR ZSM 009-3] Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 3: Advanced topics. Tratto da https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=58201

[GRAY21] Graylog: Log Management Done Right. Link: https://www.graylog.org/. Accessed: 2021-12-21

[GOA21] GoAccess: open source real-time web log analyzer. Link: https://goaccess.io/. Accessed: 2021-12-21

[Hagberg’08] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring network structure, dynamics, and function using networkx,” in Proceedings of the 7th Python in Science Conference, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[HAWILO] Hawilo, Hassan, Manar Jammal, and Abdallah Shami. "Orchestrating network function virtualization platform: Migration or re-instantiation?." 2017 IEEE 6th International Conference on Cloud Networking (CloudNet). IEEE, 2017.

[He16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[He17] He D, Chan S, Ni X, Guizani M. Software-defined-networking-enabled traffic anomaly detection and mitigation. IEEE Internet of Things Journal. 2017 Apr 17;4(6):1890-8.

[Hessar‘19] Hessar, Mehrdad, Ali Najafi, and Shyamnath Gollakota. "Netscatter: Enabling large-scale backscatter networks." 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). 2019.

[Ho19] Daniel Ho, Eric Liang Ion Stoica, Pieter Abbeel, Xi Chen, Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules, ICML 2019

[Hoch97] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation 9.8 (1997): 1735-1780.

[Huang2020] Huang, Y., Su, Y., Ravi, S., Song, Z., Arora, S., Li, K., Privacy-preserving Learning via Deep Net Pruning. https://arxiv.org/abs/2003.01876

[Huyn’18] N. V. Huynh, D. T. Hoang, D. Niyato, P. Wang, and D. I. Kim, “Optimal time scheduling for wireless-powered backscatter communication networks,” IEEE Wireless Commun. Lett., vol. 7, pp. 820–823, 2018.

[AMINA] Ibrahimpašić, Amina Lejla, Bin Han, and Hans D. Schotten. "AI-Empowered VNF Migration as a Cost-Loss-Effective Solution for Network Resilience." 2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). IEEE, 2021.

[ISA20] M. Isaksson y K. Norrman, «Secure Federated Learning in 5G Mobile Networks», en GLOBECOM 2020 - 2020 IEEE Global Communications Conference, dic. 2020, pp. 1-6. doi: 10.1109/GLOBECOM42002.2020.9322479.

[ZAPG] Zhen Ming Jiang, Ahmed E Hassan, Parminder Flora, and Gilbert Hamann. 2008. Abstracting execution logs to execution events for enterprise applications. In Proc. of the Eighth International Conference on Quality Software (QSIC).

[JIANG1] Wei Jiang and H. D. Schotten, "Neural Network-based Fading Channel Prediction: A Comprehensive Overview", IEEE Access, vol. 7, Aug. 2019, pp. 118112 – 118124

[JIANG] W. Jiang and H. D. Schotten, "Deep Learning for Fading Channel Prediction", IEEE Open Journal of the Communications Society, vol. 1, Mar. 2020, pp. 320-323

[KARIM] M. Karimzadeh, Z. Zhao, L. Hendriks, R. D. O. Schmidt, S. La Fleur,H. Van Den Berg, A. Pras, T. Braun, and M. J. Corici, “Mobility and bandwidth prediction as a service in virtualized LTE systems,” in Proc. of IEEE CloudNet, Niagara Falls, ON, Canada, 2015.

[Kellog’16] Kellogg, Bryce, et al. "Passive wi-fi: Bringing low power to wi-fi transmissions." 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16). 2016. [KB] Konečný, J., Brendan McMahan, H., Yu, F. X., Richtárik, P., Theertha Suresh, A., and Bacon, D., “Federated Learning: Strategies for Improving Communication Efficiency”, arXiv:1610.05492, 2016.

[Kingma’15] D. P. Kingma and J. Lei Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Representations (ICLR), 2015.

[Kipf’17] T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” in 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings. International Conference on Learning Representations, ICLR, sep 2017.

[Kong] J. Kong, Protecting the confidentiality of virtual machines against untrusted host, International Symposium on Intelligence Information Processing and Trusted Computing (2010), doi: 10.1109/IPTC.2010.11.

[KoNoSeTu] Z. Kotulski, T. Nowak, M. Sepczuk, M. Tunia, R. Artych, K. Bocianiak, T. Osko and J.-P. Wary, "On end-to-end approach for slice isolation in 5G networks. Fundamental challenges," in Proceedings of the Federated Conference on Computer Science and Information Systems, Prague, 2017.

[KoZb] Kotulski, Zbigniew, et al. "5G networks: Types of isolation and their parameters in RAN and CN slices." Computer Networks 171 (2020): 107135.

[LAM20] Lameh, S.F., Noble, W., Amannejad, Y., Afshar, A., "Analysis of Federated Learning as a Distributed Solution for Learning on Edge Devices," in 2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), pp. 66–74, 2020. doi:10.1109/idsta50958.2020.9264060.

[Le21] Le, J. Lei, X., Mu, N., Zhang, H., Zeng, K., Liao, X., "Federated Continuous Learning With Broad Network Architecture" IEEE TRANSACTIONS ON CYBERNETICS, VOL. 51, NO. 8, AUGUST 2021

[LCLWC] Suyi Li, Yong Cheng, Yang Liu,Wei Wang, and Tianjian Chen. 2019. Abnormal client behavior detection in federated learning. In Proceedings of the NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality. 740–750.

[Li'18] Z. Li, Q. Chen, and V. Koltun, "Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search," in Advances in Neural Information Processing Systems (NeurIPS), vol. 31, 2018.

[LIANG] K. Liang, J. Hao, R. Zimmermann, and D. K. Yau, "Integrated prefetching and caching for adaptive video streaming over HTTP: an online approach," in Proc. of ACM MM, Portland, Oregon, USA, 2015.

[Liu'13] Liu, Vincent, et al. "Ambient backscatter: Wireless communication out of thin air." ACM SIGCOMM Computer Communication Review 43.4 (2013): 39-50.

[LIU20] Y. Liu, J. Peng, J. Kang, A. M. Iliyasu, D. Niyato, y A. A. A. El-Latif, «A Secure Federated Learning Framework for 5G Networks», IEEE Wirel. Commun., vol. 27, n.o 4, pp. 24-31, ago. 2020, doi: 10.1109/MWC.01.1900525.

[LOG21] LogParser: Toolkit and benchmark for automated log parsing. Link: https://logparser.readthedocs.io/en/latest/README.html. Accessed: 2021-12-21

[Mai19] A. M. Maia, Y. Ghamri-Doudane, D. Vieira and M. F. de Castro, "A Multi-Objective Service Placement and Load Distribution in Edge Computing," Proc. of IEEE GLOBECOM, 2019, doi: 10.1109/GLOBECOM38437.2019.9014303.

[MAIM] L. F. Maimó, Á. L. P. Gómez, F. J. G. Clemente, M. G. Pérez, and G. M.Pérez, "A self-adaptive deep learning-based system for anomaly detection in 5g networks," IEEE Access, vol. 6, pp. 7700–7712, 2018.

[Mak09] A. Makanju, A. Zincir-Heywood, and E. Milios, "Clustering event logs using iterative partitioning," in KDD, 2009.

[Mal15] Malhotra P, Vig L, Shroff G, Agarwal P. Long short term memory networks for anomaly detection in time series. In Proceedings 2015 Apr 22 (Vol. 89, pp. 89-94).

[MAMUN] S. A. Al Mamun and J. Valimaki, "Anomaly detection and classification in cellular networks using automatic labeling technique for applying supervised learning," Procedia Computer Science, vol. 140, pp. 186–195,2018.

[MAO] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in Proc. of ACM SIGCOM COMM, New York, NY, USA, 2017.

[Mou21] A. Moubayed, A. Shami, P. Heidari, A. Larabi and R. Brunner, "Edge-Enabled V2X Service Placement for Intelligent Transportation Systems," in IEEE Transactions on Mobile Computing, vol. 20, no. 4, pp. 1380-1392, 1 April 2021, doi: 10.1109/TMC.2020.2965929.

[Mse19] A. Mseddi, W. Jaafar, H. Elbiaze and W. Ajib, "Intelligent Resource Allocation in Dynamic Fog Computing Environments," Proc. of CloudNet, 2019, doi: 10.1109/CloudNet47604.2019.9064110.

[Munz07] Münz G, Li S, Carle G. Traffic anomaly detection using k-means clustering. InGI/ITG Workshop MMBnet 2007 Sep 2 (pp. 13-14).

[NARA] A. Narayanan, E. Ramadan, R. Mehta, X. Hu, Q. Liu, R. A. Fezeu, U. K.Dayalan, S. Verma, P. Ji, T. Liet al., "Lumos5g: Mapping and predicting commercial mmwave 5g throughput," in Proceedings of the ACM Internet Measurement Conference, 2020, pp. 176–193.

[Nethercote'07] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack, "MiniZinc: Towards a standard CP modelling language," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 4741 LNCS. Springer Verlag, 2007, pp. 529–543.

[NMMF] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, and Ahmad-Reza Sadeghi. 2019. DÏoT: A federated self-learning anomaly detection system for IoT. In Proceedings of the IEEE International Conferenceon Distributed Computing Systems. 756–767.

[NodeExporter] https://github.com/prometheus/node_exporter

[Ola20] Mohammad Al Olaimat et al. "A Learning-based Data Augmentation for Network Anomaly Detection". In: 2020 29th International Conference on Computer Communications and Networks (ICCCN) IEEE. 2020, pp. 1–10.

[ORAN1] O-RAN Alliance, "O-RAN Near-RT RAN Intelligent Controller Near-RT RIC Architecture v2.00", March 2021

[ORAN2] O-RAN Alliance, "O-RAN Non-RT RIC: Functional Architecture v1.01", March 2021

[ORAN3] O-RAN Alliance, "O-RAN A1 interface: General Aspects and Principles v2.03", July 2021

[ORAN4] O-RAN Alliance, "O-RAN AI/ML Workflow Description and Requirements v1.03", July 2021

[Pasch10] Paschalidis IC, Chen Y. Statistical anomaly detection with sensor networks. ACM Transactions on Sensor Networks (TOSN). 2010 Sep 8;7(2):1-23.

[Perez-Penichet'16] Pérez-Penichet, Carlos, et al. "Augmenting IoT networks with backscatter-enabled passive sensor tags." Proceedings of the 3rd Workshop on Hot Topics in Wireless. 2016.

[Perez-Penichet'20] Pérez-Penichet, Carlos, et al. "TagAlong: efficient integration of battery-free sensor tags in standard wireless networks." 2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). IEEE, 2020.

[Pol20] M. Polese, R. Jana, V. Kounev, K. Zhang, S. Deb and M. Zorzi, "Machine Learning at the Edge: A Data-Driven Architecture with Applications to 5G Cellular Networks," in IEEE Transactions on Mobile Computing, doi: 10.1109/TMC.2020.2999852.

[RACA2] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, and V. Gopalakrishnan, "Back to the future: Throughput prediction for cellular networks using radio KPIs," inProc. of ACM MOBICOM, NewYork, NY, USA, 2017.

[RACA3] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic,R. Jana, V. Gopalakrishnan, B. Bathula, and M. Varvello, "Incorporating prediction into adaptive streaming algorithms: a QoE perspective," in Proc. of ACM SIGCOM NOSSDAV, New York, NY, USA, 2018.

[RACA] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, V. Gopalakrishnan, B. Bathula, and M. Varvello, "Empowering video players in cellular: Throughput prediction from radio network measurements," inProc. of ACM MM, New York, NY, USA, 2019.

[Sak14] Sakurada M, Yairi T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. InProceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis 2014 Dec 2 (pp. 4-11).

[Sal21] N. Salhab, R. Langar, R. Rahim, S. Cherrier and A. Outtagarts, "Autonomous Network Slicing Prototype Using Machine-Learning-Based Forecasting for Radio Resources," in IEEE Communications Magazine, vol. 59, no. 6, pp. 73-79, June 2021, doi: 10.1109/MCOM.001.2000922.

[SAMBA] A. Samba, Y. Busnel, A. Blanc, P. Dooze, and G. Simon, "Instantaneous throughput prediction in cellular networks: Which information is needed?" inProc. of IFIP/IEEE IM, Lisbon, Portugal, 2017.

[SAN18] A. Santoyo-González and C. Cervelló-Pastor, "Edge Nodes Infrastructure Placement Parameters for 5G Networks," in 2018 IEEE Conference on Standards for Communications and Networking (CSCN), Oct. 2018, pp. 1–6. doi: 10.1109/CSCN.2018.8581749.

[SAT17] Kaz Sato, Cli Young, and David Patterson. 2017. An in-depth look at Google's rst Tensor Processing Unit (TPU). Google Cloud Big Data and Machine Learning Blog 12 (2017).

[Scarselli'2009] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61–80, jan 2009.

[ScMoHa] F.B. Schneider, G. Morrisett, R. Harper, A language-based approach to security, in: Informatics: 10 Years Back, 10 Years Ahead, Lecture Notes in Computer Science, Vol, Springer-Verlag, Heidelberg, 20 0 0, pp. 86–101.

[Shaw02] Shaw M. " Self-healing": softening precision to avoid brittleness: position paper for WOSS'02: workshop on self-healing systems. In Proceedings of the first workshop on Self-healing systems 2002 Nov 18 (pp. 111-114).

[Shu16] R. Shu, et al., A Study of Security Isolation Techniques, ACM Computing Sur- veys (CSUR) 49 (3) (December 2016) 50.

[srsRAN] https://docs.srsran.com/en/latest/index.html

[Stan02] Staniford S, Hoagland JA, McAlerney JM. Practical automated detection of stealthy portscans. Journal of Computer Security. 2002 Jan 1;10(1-2):105-36.

[SUB21] Subramanya, T., Riggio, R., "Centralized and Federated Learning for Predictive VNF Autoscaling in Multi-Domain 5G Networks and Beyond," *IEEE Transactions on Network and Service Management*, vol. 18, pp. 63–78, 2021. doi:10.1109/tnsm.2021.3050955.

[SUND] T. Sundqvist, M. H. Bhuyan, J. Forsman, and E. Elmroth, "Boosted ensemble learning for anomaly detection in 5g ran," in IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, 2020, pp. 15–30.

[Say12] Syarif I, Prugel-Bennett A, Wills G. Unsupervised clustering approach for network anomaly detection. InInternational conference on networked digital technologies 2012 Apr 24 (pp. 135-145). Springer, Berlin, Heidelberg.

[TANG] Tang, Lun, et al. "Virtual network function migration based on dynamic resource requirements prediction." IEEE Access 7 (2019): 112348-112362

[TZQ] Tao, Zeyi and Qun A. Li. "eSGD: Communication Efficient Distributed Deep Learning on the Edge." HotEdge, 2018.

[Vaa03] R.Vaarandi,"A data clustering algorithm for mining patterns from event logs," in IPOM, 2003.

[Vaa15] R. Vaarandi and M. Pihelgas, "Logcluster - a data clustering and pattern mining algorithm for event logs," in CNSM, 2015, pp. 1–7.

[Velickovik'18] P. Velickovic, A. Casanova, P. Lio, G. Cucurull, A. Romero, and Y. Bengio, "Graph attention networks," in 6th International Conference on Learning Representations, ICLR 2018.

[VERGAD16] D. J. Vergados, A. Michalas, A. Sgora, D. D. Vergados, and P. Chatzimisios, "Fdash: A fuzzy-based mpeg/dash adaptation algorithm," IEEE Systems Journal, vol. 10, no. 2, pp. 859–868, 2016.

[Vesselinova'20] Vesselinova, Natalia, et al. "Learning combinatorial optimization on graphs: A survey with applications to networking." IEEE Access 8 (2020): 120388-120416. [ViArNe] Viswanathan, Arun, and B. C. Neuman. "A survey of isolation techniques." Information Sciences Institute, University of Southern California (2009).

[Vinyals'15] O. Vinyals, G. Brain, M. Fortunato, and N. Jaitly, "Pointer Networks," in Advances in Neural Information Processing Systems 28 (NIPS 2015), 2015, pp. 2692–2700.

[Wan17] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer and K. K. Leung, "Dynamic Service Placement for Mobile Micro-Clouds with Predicted Future Costs," in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 4, pp. 1002-1016, 1 April 2017, doi: 10.1109/TPDS.2016.2604814.

[WAN19] S. Wang et al., "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019. doi: 10.1109/JSAC.2019.2904348.

[WU20] X. Wu, J. Duan, M. Zhong, P. Li, and J. Wang, “VNF Chain Placement for Large Scale IoT of Intelligent Transportation,” Sensors, vol. 20, no. 14, Art. no. 14, Jan. 2020, doi: 10.3390/s20143819.

[XIA] Xia, Jing, Zhiping Cai, and Ming Xu. "Optimized virtual network functions migration for NFV." 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2016.

[Yang’11] L. Yang, J. Han, Y. Qi, C. Wang, T. Gu, and Y. Liu, “Season: Shelving interference and joint identification in large-scale RFID systems,” in Proc. Int. Conf. Comput. Commun. (INFOCOM). IEEE, April 2011, pp. 3092–3100.

[YANG] K. Yang, H. Ma, and S. Dou, “Fog intelligence for network anomaly detection,”IEEE Network, vol. 34, no. 2, pp. 78–82, 2020.

[YE01] Ye N, Chen Q. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. Quality and reliability engineering international. 2001 Mar;17(2):105-12.

[YE20] Ye, Y., Li, S., Liu, F., Tang, Y., Hu, W., “EdgeFed: Optimized Federated Learning Based on Edge Computing,” *IEEE Access*, vol. 8, pp. 209191–209198, 2020. doi:10.1109/access.2020.3038287.

[YI20] Yi, L., Yuan, X., Xiong, Z., Kang, J., & Wang, X. (2020). Federated learning for 6G communications: Challenges, methods, and future directions. *China Communications*, 105-118.

[Yoon21] Yoon, J., Jeong, W., Lee, G., Yang, E., Hwang, S., “Federated Continual Learning with Weighted Inter-client Transfer” Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

[Yoon21Code] Federated Continual Learning with Weighted Inter-client Transfer (Github project). Link https://github.com/wyjeong/FedWeIT. Acessed: 2021-09-27

[YU21] Yu, S., Chen, X., Zhou, Z., Gong, X., Wu, D., “When Deep Reinforcement Learning Meets Federated Learning: Intelligent Multitimescale Resource Management for Multiaccess Edge Computing in 5G Ultradense Network,” *IEEE Internet of Things Journal*, vol. 8, pp. 2238–2251, 2021. doi:10.1109/jiot.2020.3026589.

[Yue’12] H. Yue, C. Zhang, M. Pan, Y. Fang, and S. Chen, “A time-efficient information collection protocol for large-scale RFID systems,” in Proc. Int. Conf. Comput. Commun. (INFOCOM). IEEE, March 2012, pp. 2158–2166.

[YUE] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, and W. Wei, “LinkForecast: cellular link bandwidth prediction in LTE networks,” IEEE Transactions on Mobile Computing, vol. 17, no. 7, pp. 1582–1594, 2018.

[Yur] What is multi-tenancy security? (and how does it impact embedded analytics). Yurbi. (2021, March 27). Retrieved October 5, 2021, from https://www.yurbi.com/blog/what-is-multi-tenancy-security-and-how-does-it-impact-embedded-analytics/.

[YAGGHK] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian nonparametric federated learning of neural networks. In Proceedings of the International Conference on Machine Learning.

[Zang09] Zhang W, Yang Q, Geng Y. A survey of anomaly detection methods in networks. In2009 International Symposium on Computer Network and Multimedia Technology 2009 Jan 18 (pp. 1-3). IEEE.

[Zha17] L. Zhao, J. Liu, Y. Shi, W. Sun and H. Guo, "Optimal Placement of Virtual Machines in Mobile Edge Computing," Proc. of IEEE GLOBECOM, 2017, doi: 10.1109/GLOCOM.2017.8254084.

[ZCW] Y. Zhao, J. Chen, and D. Wu. 2019. Multi-task network anomaly detection using federated learning. In Proceedings of the International Symposium on Information and Communication Technology.

[ZhYuGl] Z. Zhou, M. Yu, V.D. Gligor, Dancing with Giants: Wimpy Kernels for On-Demand I/O Isolation, IEEE Security & Privacy 13 (2) (2015) 38–46.

[ZHO17] C. Zhou et al. Anomaly detection with robust deep autoencoders. ACM SIGKDD KDD 2017.

[Zuh2019] Zuh, L., Liu, Z., Han, S., Deep Leakage from gradients 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.