Chapter 5

Towards Natively Intelligent Networks

By Marco Gramaglia, Xi Li, Ginés García-Aviles, et al.[1]

Copyright © 2023 Marco Gramaglia, *et al.*
DOI: 10.1561/9781638282396.ch5

The work will be available online open access and governed by the Creative Commons "Attribution-Non Commercial" License (CC BY-NC), according to https://creativecommons.org/licenses/by-nc/4.0/

Published in *Towards Sustainable and Trustworthy 6G: Challenges, Enablers, and Architectural Design* by Ömer Bulakçı, Xi Li, Marco Gramaglia, Anastasius Gavras, Mikko Uusitalo, Patrik Rugeland and Mauro Boldi (eds.). 2023. ISBN 978-1-63828-238-9. E-ISBN 978-1-63828-239-6.

Suggested citation: Marco Gramaglia, Xi Li, Ginés García-Aviles, *et al.* 2023. "Towards Natively Intelligent Networks" in *Towards Sustainable and Trustworthy 6G: Challenges, Enablers, and Architectural Design*. Edited by Ömer Bulakçı, Xi Li, Marco Gramaglia, Anastasius Gavras, Mikko Uusitalo, Patrik Rugeland and Mauro Boldi. pp. 159–232. Now Publishers. DOI: 10.1561/9781638282396.ch5.

now
the essence of knowledge

1. The full list of chapter authors is provided in the Contributing Authors section of the book.

Over the past few years, network automation has become an increasingly important topic in both research and industry. With the growing complexity of modern networks and the need for efficient network management, artificial intelligence-based algorithms have gained attention as a promising solution. Standardization efforts have also been underway to support analytics derived from network data. As we look towards the future of mobile networks, it is expected that the integration of network intelligence (NI) will be a crucial component of the next-generation 6th Generation (6G) mobile network. While the foundational pillars of 6G have been defined by recent research and standardization works (e.g., 3GPP Release 18), it will be necessary for the upcoming waves of mobile network architecture to natively integrate NI solutions, making lifecycle management of NI an integral part of network services. This will require a novel architectural framework that supports the integration of NI into the network services, as well as the development of novel algorithms that can leverage NI to optimize network performance and enable new services and applications. In this context, this chapter proposes a comprehensive framework for the integration of NI in the 6G mobile network, which includes the key components of NI functions, management and orchestration (MANO), and infrastructure. We also introduce novel algorithms that can leverage NI to optimize

network performance and enable new services and applications. Overall, the goal is to provide a roadmap for the integration of NI into the 6G mobile network, with the aim of creating a more efficient, intelligent, and flexible network architecture that can meet the demands of the next generation of mobile services.

This chapter is structured as follows: the overall set of enablers needed for the transition towards intelligent networks is discussed in Section 5.1. Then, Sections 5.2 and 5.3 discuss the overall network intelligent architectural framework that has to deal with the specificities of the application of NI to future 6G mobile network. In Section 5.4, we then draw some guidelines for the design of such network intelligence function (NIF) that empower the vision presented in this chapter. Finally, from Section 5.5 to Section 5.12, we discuss different NI solutions that will address from different perspectives the challenges posed by the management and operation of a 6G network to eventually maximize the impact envisioned in Chapter 1.

5.1 Enablers for an Intelligent Network

Top-down network architecture design process typically starts by identifying a targeted service portfolio, given specific environmental conditions and business goals. These characteristics are captured in a set of representative use cases that are then used to derive the needed network functionality. This approach is, however, limited in terms of how well expected advancements in the application technology (e.g., computing, sensing, and actuation) and how end-user needs can be envisioned. Ideally, a network architecture would be adaptable beyond its original design goals to meet the unforeseeable changes in the service portfolio and related business goals. The network should hence allow its automatic reconfiguration with minimal manual operations in response to new and changing demands. Self-adaption of the network functionality is addressed by NIs that embed and extend artificial intelligence/machine learning (AI/ML) functionality beyond orchestration and network management into network functions and services. Intelligent Networks functions reside across the common service platform, the network functions (NFs) domain, and the management, see Figure 5.1.

It consists of AI/ML-enabled closed-loop control of NFs with the necessary supporting enabler frameworks to automate and modify network operations based on given policies and goals. The first prerequisite for NIs is a common end-to-end (E2E) analytics framework that collects and provides accurate and timely data for inference and training the AI/ML functionality. The E2E analytics framework is

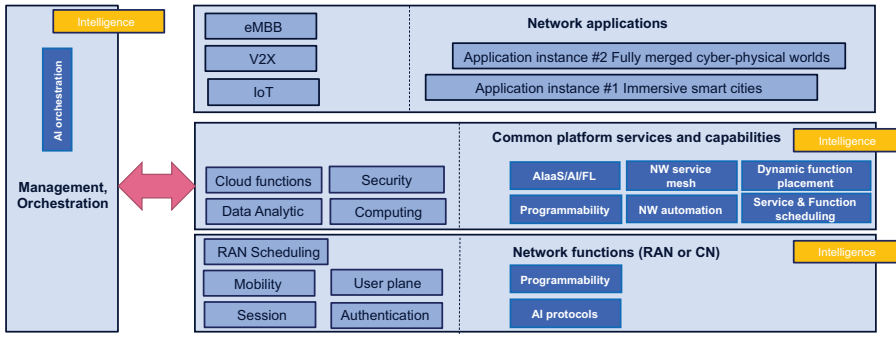


Figure 5.1. 6G enablers for the NIs in dark blue boxes in the context of the different 6G architecture domains.

responsible for exchanging knowledge across various network layers and domains, enabling AI agents and ML models training and operation. The AI-model's training depends on the concrete location where the model will be applied and the type of data that is accessible to it in a given configuration. As 6G architectures may span across a cloud continuum from the central cloud to the distributed edge and user equipment (UE), two deployment solutions are identified: first, a fully distributed edge-based solution, where all AI functions are instantiated at the edge nodes as cloud native applications; and second, a hybrid solution, where computationally expensive AI functions are trained and possibly executed in the central cloud using wider and richer data sets than would be available in a given edge cloud. As 6G UEs are assumed to consume intelligent services from the network, they can also train their own AI models collaboratively in a privacy-preserving way according to the federated learning (FL) paradigm [41]. To facilitate this, it is required to provide specific means to discover and join learning federations of UEs, allowing them to train their own AI-models collaboratively while maintaining privacy. Trained AI/ML models are stored and maintained in an AI repository, from which specific ones are installed into resource-controlling AI agents. AI agents are distributed across the network to control and adjust the resources of the network, which they are responsible for (see Figure 5.2).

Specifically, the first step is to identify the set of AI functions that are required to support this closed-loop network automation approach, possibly allowing hierarchies of closed loops, and considering their virtualization, packaging, and orchestration as in-network cloud-native functions that can be controlled and supervised by the MANO layers. To effectively address this approach, four AI functions have been identified as fundamental: the AI model repository function, the AI training function, the AI monitoring function, and the AI agent. Table 5.1 provides a short description for each of them.

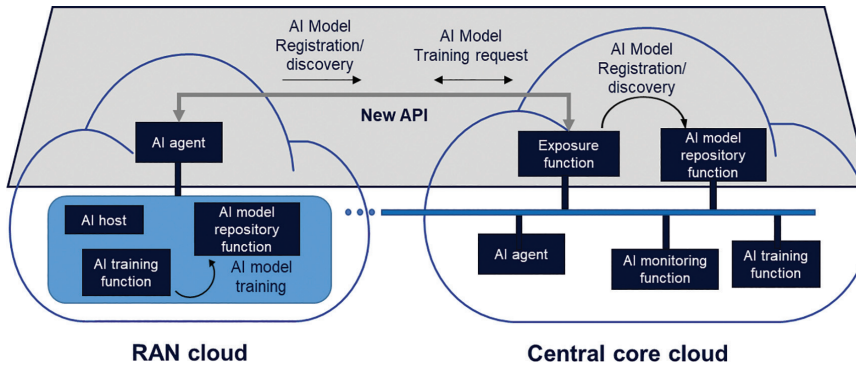


Figure 5.2. Architecture enablers for supporting AIaaS.

Table 5.1. AI functions identified as enablers for supporting AIaaS.

AI Function	Description
AI model repository function	It is the function that provides a catalogue of the available AI/ML trained models (including their metadata for capability specification), which are either already deployed or ready to be deployed within new or existing instances of AI agents. Multiple versions of the same model can be stored in the AI model repository function, e.g., related to subsequent training sessions over different datasets.
AI training function	It is the function that performs the training of AI/ML algorithms (including any required data pre-processing) and produces executable models that can be integrated in the AI agents. The AI training function is triggered either autonomously by the AI monitoring function when a performance degradation is detected, or by the management and orchestration layer whenever a new model has to be generated.
AI monitoring function	It is the function that takes care to evaluate the performance of the AI/ML models and consequently provide the trigger for training and re-training operations in the AI training function. This translates into evaluating the runtime accuracy of deployed models, as well as their performance in terms of action impact. This includes methods for identifying any potential conflict (direct or indirect) in the model inferences.
AI agent	It is the function that uses the trained AI/ML models (one or more) to perform the inference process (including any required data pre-processing functionality). According to the specific model(s) it executes, the AI agent requires specific data to be ingested. The outputs of the AI agents are then used to drive the actions and the behaviour of other functions, including 5th Generation (5G) NFs and application functions (AFs), as well as management and orchestration functions.

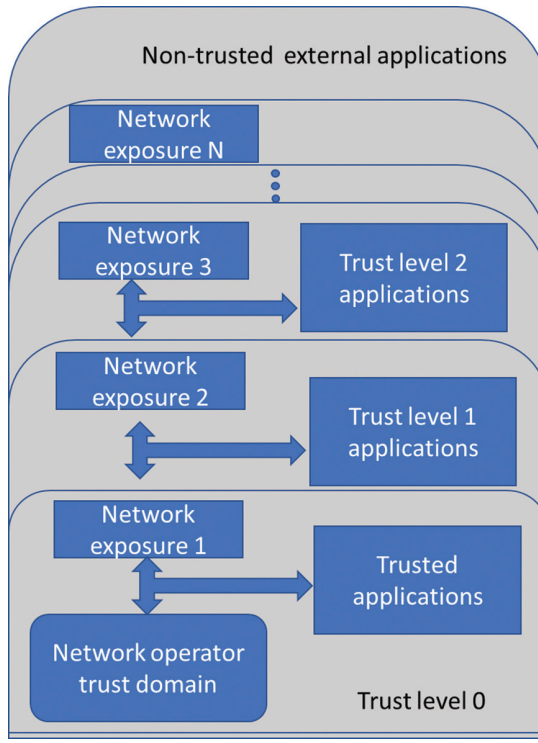


Figure 5.3. Network exposures for each trust level.

Collectively, the whole set of AI functions described in Table 5.1 form the AI as a Service (AIaaS) framework whose main purpose is to offer services to be consumed by any trusted NF, or even third-party applications that are external to the network through network exposure APIs (Figure 5.3). The consumer of the AIaaS can submit requests for inferencing decisions to AI agents for a particular action. The elements of the proposed AIaaS framework must comply with regulatory aspects of data governance that are accomplished by management of multiple trust levels between network domains to ensure data privacy requirements within each security domain. The proposed framework may be able to reduce the AI overhead, since operation of a given AI agent in one domain can be extended to different privacy domains, provided that the privacy requirements are addressed within each domain.

In addition, the underlying infrastructure layer needs to be adaptable to varying NF workloads, new functionality, and dynamic placement of NFs across the multi-cloud continuum. Dynamic instantiation of the AI agents and programmable NFs leverages AI-driven orchestration with the capability to select different processing points for network functions across the multi-cloud continuum. This is done by dynamic function placement (DFP), which introduces a two-level hierarchical orchestration solution where domain internal dynamicity is not fully exposed

externally but still network functions can be executed in a multi-domain, multi-cloud environment on-demand basis. A top-level orchestrator interacts with the domain-level orchestrator and decides which candidate domains for NFs are to be created or moved. Final deployment details and selection of the domain's internal processing point are left for the domain-specific orchestrator. The newly instantiated or reconfigured NFs and AI agents need to discover each other, and they need to communicate efficiently via a network service mesh (NSM).

In order to satisfy communication efficiency requirements, NSM must support the most common communication patterns like publish-subscribe and request-response communication. Additionally, both synchronous and asynchronous communication must be supported. NSM provides a registry (or similar) for (de-)registering services that are then exposed via service discovery, which is used to find out what services are available in NSM and how.

Networking policies also play an important role in NSM, and they must be enforced in a distributed manner, i.e., a set of networking policies distributed over the related domains is dealt with in the same manner. Additionally, OAM and AI-related knowledge may have different connectivity requirements between the related domains and their internals to harmonize management operations.

NSM-managed tunnelling between domains needs to support different types of communication patterns, and typically this is indirect communication via a communication intermediary. So tunnelled data represents *NF:intermediary* communication and could be unidirectional or bidirectional depending on how routing is arranged in the domains.

One important thing to consider is the support for traffic prioritization inside the tunnel, because it might influence how inter-domain tunnelling should be configured. For instance, one straightforward option would be to avoid mixing different quality of service (QoS) classes in the same tunnel and instead have QoS class-specific tunnels. Maintaining multiple tunnels per domain boundary should not restrict scaling too much and technically be feasible, i.e., not so much state information should be maintained per tunnel endpoint.

5.1.1 A Two-level DFP in Multi-domain Landscape

In this section, further details are provided on the two-level design for DFP, which is shortly introduced in Section 5.1. Additionally, the impacts of adding multi-domain support to orchestration and service discovery, which are closely related to DFP, are also discussed.

The proposed design has the decision-making on two levels:

- Top-level decision-making, during which the best candidate domains are searched for.

- Domain-specific decision-making, where the network function deployment will be finalized according to the domain internal resource conditions and preferences.

The decision-making structure that follows is closely related to the logic of two-level hierarchical orchestration, in which multi-domain scope is handled first, followed by domain-specific scope. The same multi-level support shall be considered in other supporting functionalities, such as monitoring and management (e.g., NF scaling), which are part of DFP. Monitoring is naturally done within a domain, and then the information is exposed externally. One critical requirement of the design is the ability to expose information from domain(s). The top-level part is dependent on the information that is exposed by individual domains. It is crucial that the validity of the information is adequately scalable. If the exposed information is too dynamic ("short-lived"), a distributed system may end up wasting time and resources trying to keep dynamic information synchronized. Aside from the information's validity and lifetime, the amount of exposed information can also be a limiting factor. Therefore, domains may need to pre-process the information and expose it as information aggregates. However, in some cases, raw information might also need to be exposed.

NF scaling is a common operation for legacy orchestration, and it is used as an example for defining the impacts of the two-level design for decision-making and operation execution. This is the reason why multi-domain and intra-domain scaling should be considered separately:

- Multi-domain: scaling over multiple domains requires technical key performance indicators (KPIs) (e.g., utilization metrics representing the current resource conditions and load estimations for the near future from each domain) to make service offering-related decisions (e.g., requesting increased service capacity or relocating service offerings) from one domain to another.
- Intra-domain: traditional NF scaling up/down or in/out based on the local needs and/or external triggers such as requests from the top-level entity.

The duality of the decision-making closely corresponds to the timing aspects of the related control loops. For instance, any control loop running on top of multiple domains (top level) presumably has longer execution delays than those running inside a domain, which is a crucial piece of information since the potential execution delay of the control loop(s) determines how fast such a system can react to changes. This reaction sensitivity also relates to the dynamic environment in which the control loop is operating, and the information based on which the control loop is executed is part of this environment.

Similarly, to the exposed domain information, it is important that the decision-making information is not too dynamic or short-lived. This implies that longer control loops, such as those used in multi-domain scope, are better suited for wide-scope predictive decision-making. In addition, faster control loops, which are used inside domains or even inside logical or physical nodes, are better for reacting to rapid changes under dynamic conditions. In this design, the top-level domain does predictive operations with a longer lifetime, and each underlying domain operates on that basis. Therefore, from a decision-making perspective, domains are able to react faster to local changes and make short-term decisions and operations until longer-term decisions are propagated from the top level. It is important to have some form of feedback channel between the levels. For instance, domains can communicate their internal actions and decisions back to the top level, where the long-term solution(s) can be adjusted accordingly, i.e., predictions have to be updated according to what has happened at the lower (domain) level. Additionally, domains can have a certain degree of autonomy by having instructions and thresholds on when to report back to the upper (top level) level. It is essential that the used service discovery solution is efficient enough and scales well. To ensure proper scaling, service discovery can also implement a two-step design: (i) resolve the destination domain, and (ii) resolve the communication endpoint (NF) inside the domain. In multi-domain environment and especially when DFP is effectively in use, it is important that the establishment of inter-NF communication session could be optimized. According to this optimization, service discovery can be a domain internal thing without requiring longer loops between multiple domains, including some centralized functionalities.

For multi-domain environments, the usage of communication intermediaries (e.g., service communication proxy (SCP) in 5G environment) can help us in making scalable inter-NF communication by using indirect communication, where an intermediary hides a set of NFs and that for such NFs the service discovery results in the communication address of the corresponding intermediary. This ensures the scaling properties of service discovery and makes the data in (top level) service registry static since all the dynamics of NF instances are masked by communication intermediaries. Typical legacy schema for masking NFs has been grouping them based on NF type behind an intermediary and at the same time enable load balancing opportunity per NF type. However, in a more heterogeneous multi-domain environment where NF instances can be found in the core, (far) edge, or even behind the radio interface, the question of whether this is the only option arises, implying that new schemes based on which NFs can be grouped together behind intermediaries will likely emerge. It is well known that direct inter-NF communication, especially between domains, potentially limits how effectively load balancing can be done for NFs.

Context-related communication (e.g., context transfer or context sharing) could follow the normal inter-NF communication patterns, e.g., indirect communication via intermediary. This may be the only available option for inter-domain cases, but for the cases where the target NF instance is in the local domain, direct *NF:NF* communication might be preferred. In this way, context-related communication does not need to share communication medium with inter-NF communication and could receive higher priority, respectively, because NF context(s) must be in place before any inter-NF communication would be meaningful. There are no technical requirements to support traffic prioritization based on its type, but using separate communication mediums could further help in isolation and ensure separate scaling as well.

Regarding shared data between old and new NF instances, it could be categorized as (i) generic and (ii) instance-specific. The former is functional or practical as such between instances, but the latter has to be updated by a new instance, which receives the data provided by a context transfer functionality.

Shared data might have different real-time (RT) requirements based on how it is distributed and kept synchronized. For near-real-time (NRT) data, any changes must be distributed immediately to ensure that data updates are propagated fast enough. Respectively, for non-NRT (NNRT) data, there is no such urgency to make distributed data changes immediately; thus, the system can collect multiple updates and then distribute them as an aggregation. There are strict timing constraints in the interaction between the DFP and the service registry that determine whether the results of the DFP operations that impact service availability and resourcing are updated in time. It should be noted that updates with indirect communication via an intermediary, which is not directly visible in the service registry, can potentially be faster than the updates in the domain-wide service registry for direct communication updates. Naturally, this requires that DFPs be interfaced for distributed operations. DFP needs to support coordinated operations across domain borders to support such advanced use cases. Last, two topics are introduced for a new type of NF instance-related operation in DFP that derives especially from multi-domain support. Namely, they are NF relocation and NF offloading. While these two operations and the already-mentioned NF scaling are quite self-explanatory, it is often hard to differentiate them from each other rationally. Their existence can be justified in a multi-domain environment for this purpose, and they are defined as follows.

Scaling is called a "legacy" operation for NFs within this context. It is primarily driven by NF consumer demand and refers to scaling existing NF producer(s) capacity (up, down, in, or out) to ensure a specific NF's "optimal" service level. KPIs for "optimal" can be obvious in many ways. It should be noted that multi-domain implies new requirements for NF scaling as well. For instance, how to

support distributed multi-domain transactions for NF scaling by coupling together domains' atomic transactions is a valid research question.

"NF Relocation" is a permanent operation that is used to change the physical location of NF instance(s) in the network topology. As soon as relocation and possible context transfers are done, the original instance is deleted. For example, a management entity might ask for an optimization to request NF relocation. The need for relocation can also be derived from the 6G service model, where it has been envisioned that end-user-specific (far) edge services that may have strict latency requirements are required to follow the end-user in the network topology.

NF offloading is the distribution of existing NF workloads to new network topology locations with newly created NF instance(s). One example of an offloading decision could be sudden changes in capacity offerings, i.e., load balancing. Offloading differs from relocation in a way it does not involve the deletion of the original NF instance(s). In other words, offloading can be considered a temporary expansion of NF capacity for offloaded workloads, which can be withdrawn once the offloaded tasks are finished. One typical scenario is NF offloading from the core domain to the (far) edge domain to meet the latency requirements.

5.1.2 AI Workload Placement Perspective

This section presents an intelligent network enabler for effectively managing and orchestrating computing resources. Computation tasks may vary, e.g., in relation to the source of the data to be processed. One critical computation tasks' category is related to AI and ML workloads. Specifically, (i) balancing the computation and communication load, (ii) moving computation, and (iii) efficient placement of computation further challenges are introduced. Physical nodes (e.g., user equipment, edge/cloud servers) that undertake the execution of AI workloads may encounter trust level problems, traffic load-related issues, or challenges/requirements related to their energy consumption.

The availability of the data, along with potential restrictions on its sharing among the various network segments/nodes due to privacy reasons, is one of the crucial factors that must be taken into account. In order to provide maximum privacy, but also network transmission-related communication overhead and, as a result scalability, the placement of AI workloads must consider the vicinity of the data source, e.g., in terms of network hops/available bandwidth for transferring the data, as well as the data volume and frequency based on which this must be transmitted. In addition to the aforementioned factors, one should also examine the special features/characteristics of AI workloads, namely different input data models, behaviour models, and knowledge sharing needs among nodes of the network, as

well as considerations related to the trust level of the nodes that undertake computation tasks.

A management methodology focusing on AI workloads for Beyond 5G (B5G)/6G architectures could target a three-fold strategy, i.e., to minimize the energy consumption of the overall network towards sustainability, minimize the processing and transmission delay in relation to the data volumes that are to be transmitted towards the processing entities, and maximize the overall trust level of the system by prioritizing nodes/AI agents with high trustworthiness indexes [1].

5.2 NI Native Architecture Empowered by AI

In this section, an architectural model that stems from and integrates with current standards (e.g., O-RAN, 3GPP, and ETSI) and realizes the vision of native support for E2E NI coordination is described. The use of NI or AI to represent AI functions applied to network problems is alternated in this chapter. To achieve the specified result, the proposed architectural framework builds upon the initial guidelines provided by [2]. This previous study covered a subset of the current requirements from a higher-level perspective and focused on the relationship among NI degrees of freedom, input-output relationships, and RT constraints subject to infrastructure observability and controllability. In the following, the proposed architectural framework is discussed in detail, as depicted in Figure 5.4.

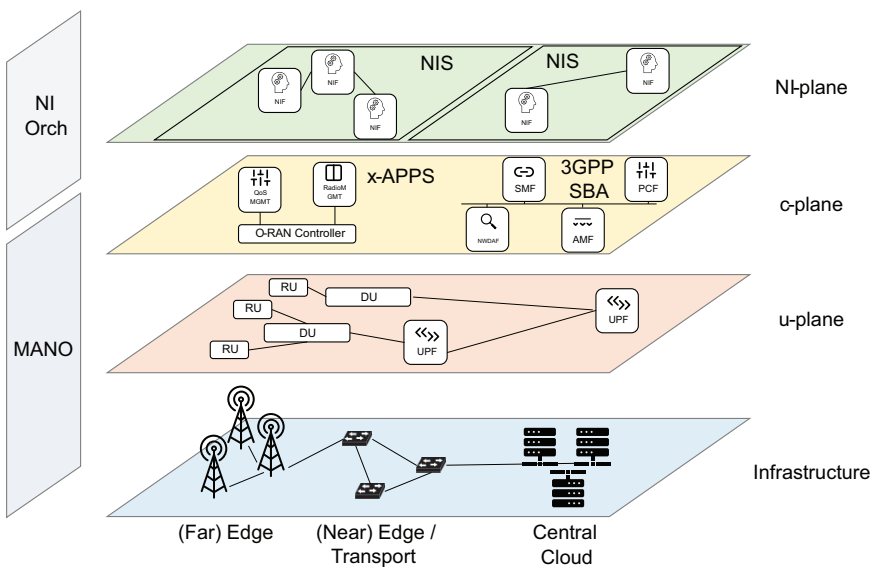


Figure 5.4. The framework for the NI integration in the overall architecture.

The network's softwarization and data-driven trends have led us to decompose the structure into four complementary layers. Along with the infrastructure layer, the control plane, and the user plane, which are the three essential building blocks of a software network like the 5G network, an additional layer is envisioned: the network intelligence plane (NIP). This NIP is introduced owing to the current trends in the industry that integrate the functions related to NI [3] in the network.

The MANO of this compound network is performed by two modules: the MANO, as traditionally done in 5G networks, which handles the typical lifecycle management of the network and network slices, and a new sibling element, the network intelligence orchestrator (NIO), which takes care of all the operations related to the management of the intelligence of the network. These operations include:

- The *selection* of the NIFs that come together to build a network intelligence service (NIS) to pursue one of the KPIs envisioned by 6G networks, such as energy efficiency.
- The *monitoring* of such functions, including the monitoring of their KPIs (e.g., their accuracy) and of the specific actions that may be taken to optimize them (e.g., meta-parameter change, re-training, or model changes).
- The specific *training* procedures in case of learning models.
- The *interaction* with the MANO to handle service and resource orchestration.

For MANO, all the definitions and functional components from ETSI can be reused, omitting these to avoid clutter. Instead, in the next sections, the internals of the novel NI orchestration, specifying interfaces and procedures, are described.

5.2.1 Detailed Architecture

In this section, an overall representation of architectural models is provided as well as the list of relevant definitions from standards. The newly proposed components (e.g., NIF, NIS, and NIO) and the explanation of how they fit such existing architectural models (e.g., NIF mapping to xApp/rApp in O-RAN and NWDAF in 3GPP) are presented in the next.

5.2.2 Taxonomy

As discussed in [3], the management of NI in a softwarized network can be performed in a similar manner as the management of network services is designed for 5G networks. This allows to reuse well-known concepts, adapting them to the context of NI. The high-level interactions are depicted in Figure 5.5, highlighting how the interactions take place.

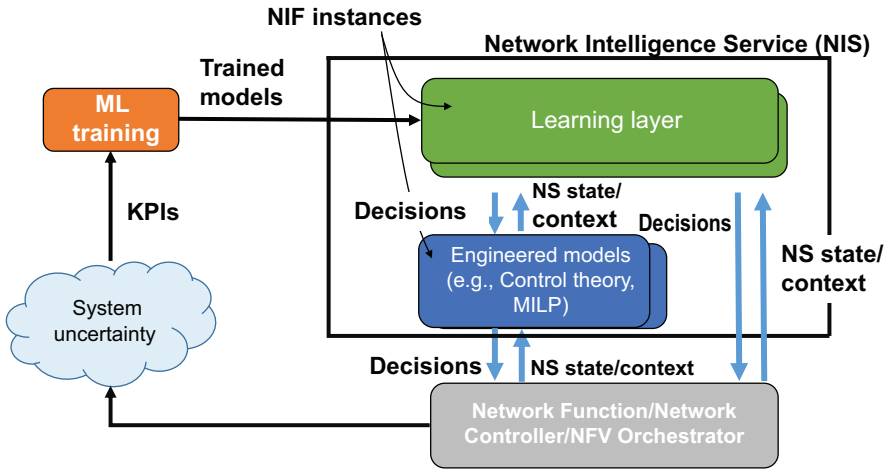


Figure 5.5. The taxonomy of the NI.

Thus, analogously to the information model specified for network management by, for instance, 3GPP, the concepts of *NI Service* (cf. Network Service, leveraging a slice such as enhanced mobile broadband (eMBB) or ultra-reliable low latency communications (URLLC)) and *NI Function* (like any function specified by, e.g., 3GPP or O-RAN) are defined as follows:

NIF: Functional block within a network (slice) that implements a decision-making functionality to be deployed in a controller, NFV orchestrator, or network function and has well-defined interfaces and behaviour.

NIS: Composition of NIFs that has a specific target, usually related to a specific set of targeted KPIs. Table 5.2 shows examples of NISs derived from NI functionalities developed in [3].

There is a one-to-many relationship between NIS and NIFs, as the former could be provided by one or more instances of the latter. Consequently, network operators or service providers can, for example, request specific sustainability and reliability services targeting one or more KPIs. The NI orchestration will take care of providing such a service by composing specific instances of NIFs.

NIFs themselves could be of different kinds. They could be learning models based on, e.g., deep neural networks (DNNs) or engineered models, or they could be built upon specific optimization algorithms such as those based on control theory or mixed-integer linear programming (MILP). These NIFs have two main interactions with the underlying layers (c-plane, u-plane, or infrastructure, proxied by a NFV orchestrator): NIFs inject decisions and receive information about the network slice state and the context of such a state. A NIS is hence a coordinated

Table 5.2. Examples of NI services derived by NI functionalities [3].

NI Service	KPIs	NIFs
Reliable virtualized RAN	Reliability	• Reliable distributed unit (DU) for RAN virtualization • Orchestration of radio and computing resources in vRANs
Sustainable network operation	Virtual network function (VNF) energy savings Compute resource savings OPEX savings	• Cloud acceleration for virtualized RAN • Compute aware scheduling analytics • AI-enhanced edge orchestration • Data-driven resource orchestration • Multi-timescale network slice reservation
Network capacity management	Wireless capacity increase	• Reconfigurable intelligent surfaces control
Edge orchestration	OPEX savings	• Network service auto-scaling • Capacity forecasting

effort of one or more NIFs that could be arranged hierarchically. For example, a NIS could be composed of a learning-type NIF sending decisions to an engineered model NIF, which in turn acts on the underlying infrastructure.

In order to manage the interaction between different NIFs, a further level of detail is needed, which decomposes each NIF into atomic elements that perform a specific operation. That is, besides the specific requirements associated with the algorithms, a mechanism to create a common framework to map the most common features of NI algorithms is needed, and subsequently integrate them into the overall architecture, and design the necessary interfaces that algorithms use to interact with their environment.

For the purpose of creating a common framework and integrating NI algorithms into the overall architecture, a methodology that is already utilized by the MAPE-K (*Monitor-Analyse-Plan-Execute* over a shared Knowledge) feedback loop is proposed. The MAPE-K feedback loop is considered one of the most influential reference control models for autonomous and self-adaptive systems [4]. This nomenclature adopted to label NI requirements allows classifying the algorithms that run at NI instances in a unified manner, based on how they interact with the other elements of the network.

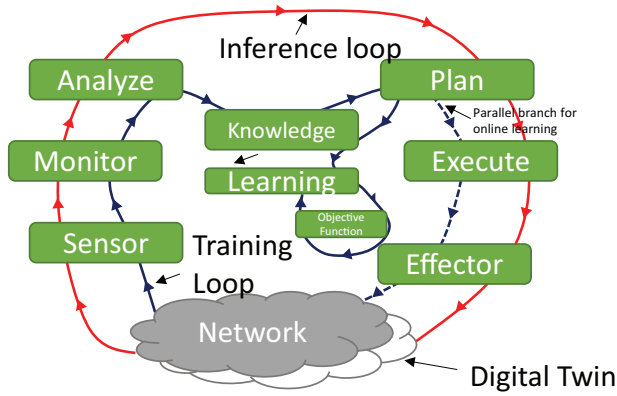


Figure 5.6. The N-MAPE-K framework.

It is worth noting that the original MAPE-K framework has limitations in the target context of mobile network functionalities supported by NI. Therefore, changes to the legacy MAPE-K are proposed to consider the specificities of the network environment, as depicted in Figure 5.6. In the figure, the different training/control loops that may be implemented by a NIF are depicted: the inference loop, the training loop, and the training loop with a branch for online learning. The model emerging from this adaptation is coined *Network MAPE-K* (N-MAPE-K). Specifically, the MAPE-K is extended along two dimensions:

- The purpose of the NIF, whether the knowledge is being trained or used in inference for the operation of the network, follows the MLOps paradigm.
- The nature of the NIF algorithm distinguishes between online learning and pre-trained/engineered models.

For the latter, the knowledge module shall be integrated with a training definition, containing all the attributes of the algorithm and thus specifying aspects such as the input data shape, batches, and most importantly, the used loss function (which could be dynamically adjusted) and the state/action representation, depending on whether the NIF algorithm belongs to the family of supervised learning or to the one of online learning. Additionally, the effector and the sensors can also be redirected to a digital twin element if needed by the specific NI instance, in order to disentangle the learning-loop process from the real operation of the network.

Hence, each NIF can be further split into NIF components (NIF-C) as follows:

- The Sensors specify all the probes that are needed to gather the input data and the kind of input data to be gathered. In principle, the APIs are specified at this level.
- The Monitor block specifies how the NIF interacts with the sensors.

- The Analyse block includes any pre-processing, summary, or preparation of the data, such as averages, autoencoders, and clustering algorithms.
- The Plan implies the specific NI algorithm that is implemented, for instance a neural network (NN) fulfilling categorization tasks.
- The Execute part specifies how the algorithm is going to interact with the system and how to possibly change configuration parameters.
- Finally, the Effector includes specific configuration parameters updated in the network function, again specifying the API.

With this framework, already presented in [5], NI algorithms are represented in a unified way and hence perform all the NI lifecycle management as discussed next.

5.3 NI Orchestrator

The structure of the proposed NIO is detailed next. The NIO is the element in charge of managing and orchestrating the NIS, NIF, and NIF-C, the elements that build the NI. The NIO structure is mutated from the layered structure of the ETSI NFV MANO framework, tailoring the components to the specificities of NI.

5.3.1 NIO Internals

The network intelligence orchestration framework, depicted in Figure 5.7, is structured over three levels, as in the ETSI NFV MANO framework. In the following,

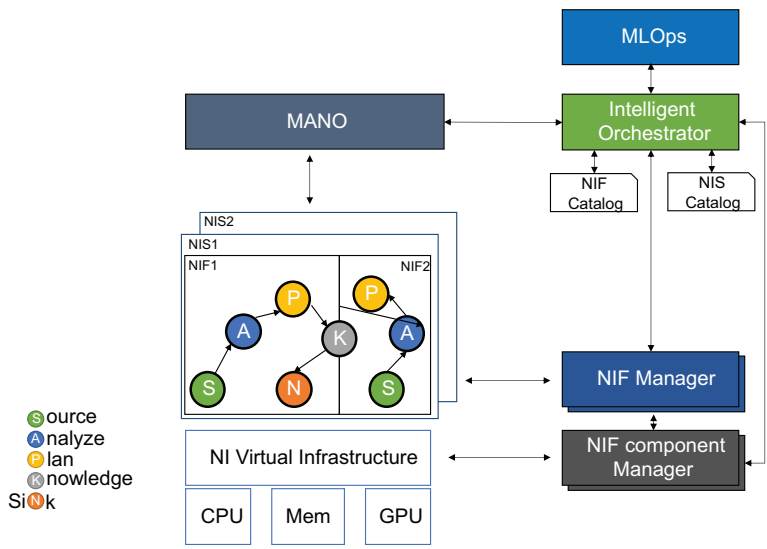


Figure 5.7. The NI orchestration framework.

references to lifecycle management refer to onboarding, instantiation, termination, scaling, and state retrieval.

NIF component manager: The NIF component manager is in charge of handling the lifecycle of the NIF-Cs, independently of their kind (i.e., independently of whether they are Source/Analyse/Plan/Knowledge/Sink) and their connection towards the infrastructure. For instance, in the case of Sources, the internet protocol (IP) addresses of the different data producers shall be provided, while in the case of Sinks, the specific configuration of application programming interface (API) endpoints has to be configured. This will have specific instantiations according to where this interaction shall take place. For instance, if the NIF is executed from the core, then Sinks and Sources shall integrate with the network registry function (NRF) and the network exposure function (NEF), properly synchronizing with the network data analytics function (NWDAF), whose analytics are captured as a set of Analyse, Plan, and Knowledge boxes. Similar considerations also apply for other network domains, such as the radio access network (RAN), where this framework can be fully integrated into the O-RAN x-Apps ecosystem.

NIF manager: The NIF manager, instead, has a global view of the set of NIF-C that composes every NIF: besides the lifecycle management of the NIF, this module is in charge of monitoring the health of the intelligence functions. This includes typical diagnostic information (e.g., constantly checking the KPIs yielded by the NIFs, like the accuracy) if the NIF is being used in inference or it is an online learning solution, or other metrics such as the loss and the training loops if the NIF is currently being trained (like the one presented in Section 5.2). The NIF manager is responsible for setting the meta-parameters of the models (through the interaction with the NIF-C manager) and reporting the health status of the NIF to the upmost module in the hierarchy, the Intelligent Orchestrator.

This module is in charge of the lifecycle management of the NIS, by properly coordinating the NIFs that build each of them. This includes the possibility of sharing NIF-C among different NIFs (e.g., two NIFs that require the same input) and also the arbitration policies in the case of two NIFs that share the same sink, that is, the configuration APIs.

Intelligent orchestrator: The intelligent orchestrator manages the connection towards the network MANO to gather important information, such as the expected network KPIs for the managed slice and service, as well as the information of the underlying network infrastructure. The intelligent orchestrator has catalogues of already-onboarded NIS and NIFs. In particular, NIFs may need to be (re-)trained to cope with changing or different conditions or on a periodic basis. In this case, the network orchestration interfaces with an external platform to build ML pipelines and perform such operations.

5.3.2 NI Distributed and Scalable MANO Framework for Massive Number of Network Slices

Sustainable, scalable, and energy-efficient massive slicing MANO in 6G systems will rely on distributed AI-native architectures, where both the analytics and the decisions will be performed close to the monitoring points via federated and multi-agent learning strategies. This will improve both scalability and reaction times, paving the way for zero-touch network service management (ZSM).

For 6G evolution, the proposed distributed and scalable MANO framework investigates zero-touch MANO in the support of network slicing at massive scales for B5G and 6G networks focusing on MANO scalability [6]. It proposes a novel autonomic MANO framework, heavily leveraging the distribution of operations together with state-of-the-art data-driven AI-based mechanisms. As shown in Figure 5.8, the proposed architecture splits the centralized management system into several management sub-systems, distributing both the intelligence as well as the decision-making across various components. Each technological domain may have one or several distributed management elements. Hierarchical, distributed, scalable, and AI-based management of a massive number of network slices across domains towards zero-touch management is achieved by distributing the management functions over all entities in charge of the life cycle management (LCM) and runtime management of network slices. This ensures the delegation of service-level management functions to be onboarded within the network slice.

The proposed distributed and scalable MANO framework has been designed for AI-driven MANO of massive numbers of network slices. It supports operations of fault management, self-healing, self-configuration, performance optimization

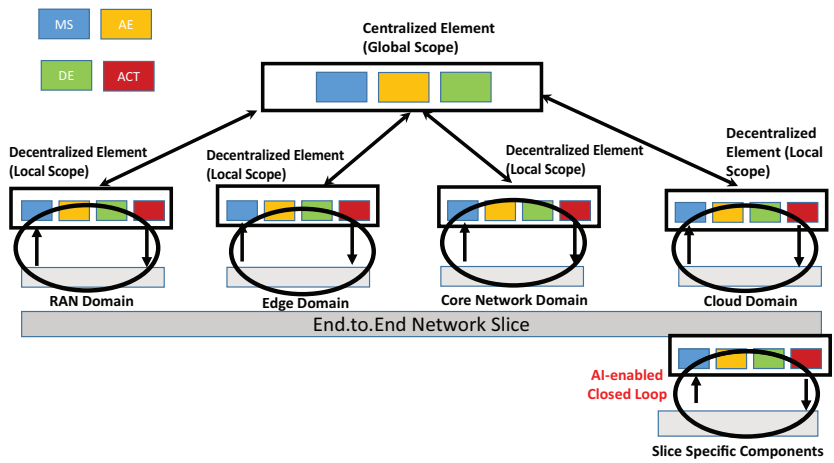


Figure 5.8. Proposed distributed and scalable MANO concept for 6G evolution – a high level outline.

(including energy-saving), and security operations. The proposed framework follows the so-called MAPE paradigm and is in line with key ETSI ZSM and ENI requirements and, hence, placed within the architecture introduced in Section 5.2. The AI-driven, multiple MAPE loops are used for level-specific, control loop-based optimization. Distributed closed control loops assist the LCM entities with state-of-the-art AI-based and data-driven mechanisms.

The proposed distributed and scalable MANO framework is based on four key component types: the monitoring system (MS), analytics engines (AE), decision engines (DE), and actuator (ACT) that present different parts of a control loop pipeline instantiation that can be used according to the target scope of analysis and decision with the goal of minimizing the raw data exchange to allow a fast decision analysis and decision. Each such loop is implemented using a pipeline composed of MS, AE, DE, and ACT. Typically, AEs and DEs are AI-driven. These four components present different levels of instantiation that can be used according to the target scope of analysis and decision, where the aim is to minimize the raw data exchange and allow a fast local analysis and decision. Their descriptions are as follows.

The MS is an entity responsible for gathering a set of different metrics from the systems that the DE is controlling. The MS can be common for multiple control loops.

- An AE performs time-series predictions as well as feature space regressions, clustering, and classification to extract insights from the measurements collected by the MS.
- A DE decides the LCM actions that need to be applied to face the issues detected by the AE. It may also control the MS measurement granularity or the AE prediction parameters.
- The ACT converts high-level (intent) reconfiguration commands obtained from the DE sublayer into a set of atomic reconfiguration commands. Hence, DEs do not have to deal with details of reconfiguration.

5.3.2.1 Distributed and scalable MANO architecture

For 6G evolution, the proposed distributed and scalable MANO architecture, its corresponding components, and interfaces are shown in Figure 5.9.

- Business layer: The layer consists of the business entities that operate the framework, provide slice management services to slice tenants, or own a slice (slice tenants).
- MANO layer: The layer consists of the core functions of the framework responsible for MANO of slices, slice LCM, and exposure of management interfaces to specific business entities.

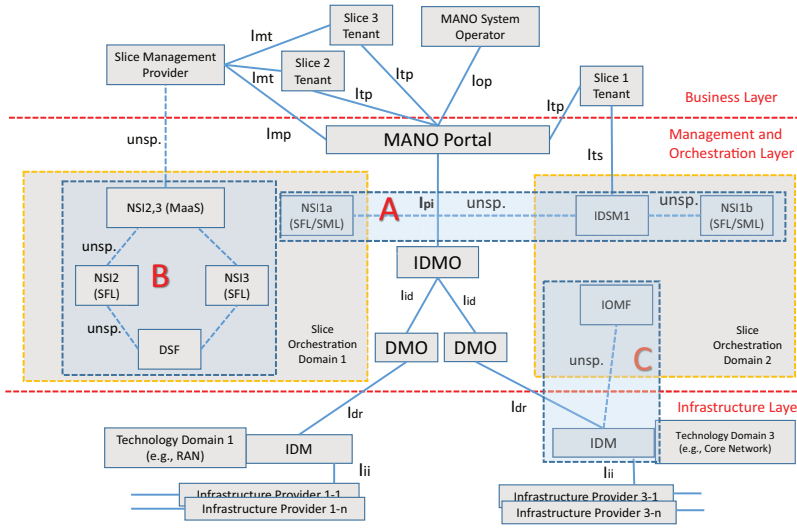


Figure 5.9. Architecture, components and interfaces of the proposed architecture distributed, and scalable MANO framework for 6G evolution.

- Infrastructure layer: This layer consists of the infrastructure, infrastructure providers, and functions enabling communication with the MANO layer and enabling optimization of the usage of infrastructural resources.

In the following, detailed descriptions are provided for each static and dynamic component of the framework belonging to the management and orchestration and infrastructure layers, with a focus on lifecycle management. The description of business entities and their interactions is presented from a high-level perspective. The components of management and orchestration and infrastructure layers are as follows:

- The MANO portal is used by slice tenants, slice management providers, and infrastructure providers to request operations regarding slice LCM, i.e., slice deployment, slice modification, and slice termination [6]. It also exposes the capabilities offered by the proposed distributed and scalable MANO framework (available slice templates, etc.) and partakes in negotiations related to the business dimension of the contract. The portal is also used to pass all the accounting and billing-related information.

Inter-Domain Manager and Orchestrator (IDMO) is equivalent to the 3GPP network slice management function (NSMF) [7] and exposes the northbound API (NBI) for the OSS/BSS or consumer service management function (CSMF). IDMO oversees the LCM of E2E network slices. It has full-scope slice MANO decision capabilities and takes global actions for network-wide, cross-slice, and cross-domain optimizations. The tenant or the slice owner interacts with the

OSS/BSS or CSMF to define the network slice to deploy using an already-generated blueprint to generate a network slice template (NST) that includes attributes and meta-data on the network slice (e.g., the start date and end date, slice owner, type of slice, etc.) and information on each sub-slice composing the network slice. For instance, in the case of computing resource (i.e., cloud or edge) domain, the NST may include information such as the number of CPUs, memory, and virtualization technology (i.e., virtual machine (VM) or containers) to be used. For the RAN domain, resources may be related to the functional split type [8], the MAC scheduler algorithm, the number of radio resource blocks (RB), and others. Finally, for the transport domain, resources may include the type of link (bandwidth, latency), number of virtual local area networks (VLANs), front-haul link capacity, virtual private network (VPN) links, and QoS. Each technological domain's needed resources are enclosed in the NST in the form of a technological domain-specific descriptor. For instance, for the NFVI domain, the resources are described using a network service descriptor (NSD) that includes the VNF(s) list and their descriptors.

Domain manager and orchestrator (DMO) is responsible for the orchestration and management of each of the slice orchestration domain (SOD) slices. Each technological domain is managed and orchestrated by its own entity, DMO, which is equivalent in 3GPP to the network sub-slice management function (NSSMF) [7]. Depending on the technological domain, a NSSMF may correspond to NFVO for cloud/edge, RANO for RAN, and software defined networking (SDN) controller for the case of the transport network.

Domain shared functions (DSFs) are a set of shared functions (VNFs) that can be implemented as physical network function (PNF)/VNF or CNF and can be reused by SFLs of multiple slices. The approach provides a reduced footprint of the deployed slice.

The inter-domain slice manager (IDSM) is a part of the slice template (a set of VNFs), and in some cases, it can be generated automatically by the IDMO (if IDMO is responsible for slice template split between multiple SODs). When IDSM is in use, it provides the slice tenant with a management interface. The IDSM is also responsible for the calculation of slice-related KPIs.

In the proposed distributed and scalable MANO framework slice structure, two separate layers can be distinguished – the slice management part called the slice management layer (SML) and the slice main part called the slice functional layer (SFL). The SML is an implementation of the ISM concept, having in mind the AI-based MAPE management. The SML is, therefore, split into MS sublayer (MS-S), analytic engines sublayer (AE-S), DE sublayer (DE-S), and actuating functions sublayer (ACT-S). The SFL contains a set of virtual functions that form the network slice to be deployed. The SFL part is composed of virtual functions that are dedicated solely to a slice (they are included in the slice template).

The infrastructure domain manager (IDM) provides the overall management of the infrastructure. Its interface to DMO allows for the allocation of resources (NFVI agent), the exchange of information related to the energy consumption of resources, and the exchange of the information related to the cost of resources that can be used by IDMO for resource brokering. The framework enables programmable infrastructure management. The DMO orchestrator can dynamically deploy management functions that cooperate with IDM to achieve infrastructure management. The IDM has an interface to the infrastructure provider, who can use the MANO portal asking for the deployment of additional infrastructure management functions, called IOMFs (see below). The functions are orchestrated in a similar way to slices, and LCM requests are sent by the infrastructure provider to the MANO Portal.

The infrastructure orchestrated management functions (IOMF) are specific functions that support infrastructure management. They can be orchestrated by the IDMO upon request of an infrastructure provider via the MANO portal.

Option A in Figure 5.9 concerns the deployment of a self-managed multi-domain slice. Such a type of slice requires a special component that is responsible for E2E slice management; it is worth noting that this component is implemented as a part of the E2E-slice template, not as a part of DMO. Option B shows the deployment of slices that use the PaaS approach, i.e., shared functions that implement the management as a service (MaaS) paradigm. Another set of shared functions, DSF (see further), is in this option exploited by the functional part of the slice. Option C shows infrastructure management-oriented and orchestrated DMO functions that are created in a similar way to slices, on the request of the infrastructure provider. The components that are deployed within each option are described in detail in the forthcoming sections. More details on the overall concept can be found in [9] and [6].

5.3.2.2 Monitoring system of distributed and scalable MANO framework for 6G networks

The MS of the distributed and scalable MANO framework (MANO-MS) implements the crucial concept of observing the network within the proposed architecture and gathering vital information that will feed the NI agents operating the network. In particular, MANO-MS is a distributed MS, developed within a container infrastructure system that relies on Kubernetes. Figure 5.10 illustrates the architecture of MANO-MS, with the central component being the sampling loop, implemented using sampling functions. The manager component is responsible for the life-cycle management of the sampling functions and provides access to the gathered telemetry data via *q* and *db* interfaces. The communication between these

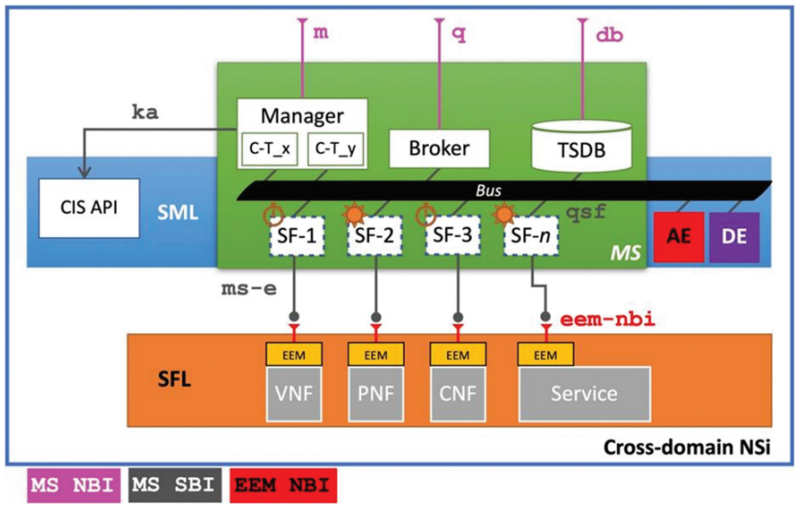


Figure 5.10. MANO-MS: Monitoring system of the proposed distributed and scalable MANO framework.

components is facilitated by the service bus of MANO-MS, which is implemented through Apache Kafka.

The platform is controlled through the *m* interface, as depicted in Figure 5.10. During deployment, the bus topics are configured, and the user then requests a sampling loop by submitting a configuration file. After validation, the manager generates the corresponding deployment instructions, which are passed on to Kubernetes (K8s) for deployment. The deployed sampling function, implemented as a CNF, enters the operational stage, and periodically samples the telemetry data as configured, publishing it to the bus.

The MANO-MS platform is versatile and capable of monitoring a wide range of telemetry data, as long as an embedded element monitoring is available (e.g., metrics from different network components as depicted in Figure 5.11).

5.3.3 Enabling SDN Control with NI

In previous sections, it has become clearer that NI can be provided at multiple network domains and through multiple orchestration layers. In this section, an architecture for enabling an SDN controller with intelligence is provided, as are several use cases.

Artificial intelligence, and more specifically machine learning, has been demonstrated as a feasible tool to support network traffic analysis in multiple scenarios. To this end, the introduction of ML in transport networks has been previously studied [10], but typically is focused on non-RT scenarios and with the usage of dedicated tools.

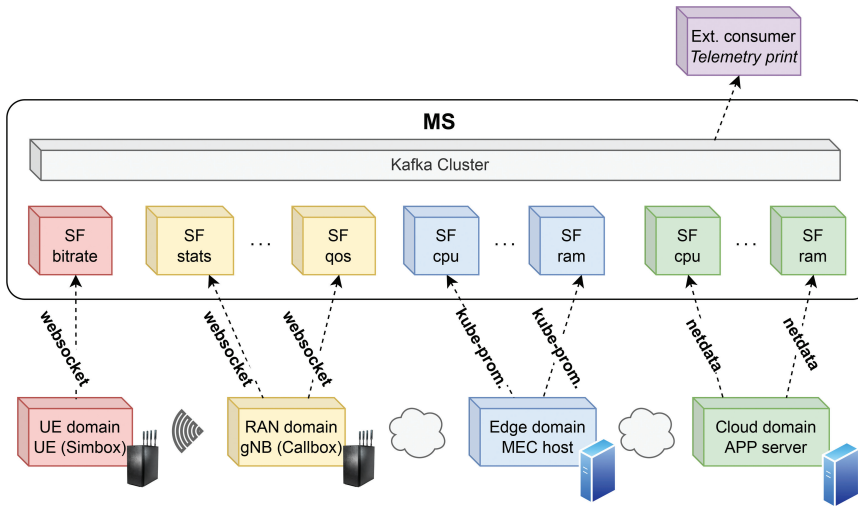


Figure 5.11. Multi-domain slice for 5G end-to-end monitoring.

Network programmability has been brought to transport network due to the introduction of SDN, which proposes the decoupling of control and data planes of underlying network equipment. To this end, it offers improved network programmability of the equipment and introduces the concept of SDN controller. The SDN controller is responsible for the control and management of the network equipment, allowing a logically centralized control and management.

TeraFlow project has proposed a new transport SDN architecture that enables an open environment for network applications and devices using full standard interfaces with container-based services, which are deployed as micro-services and managed on elastic infrastructure through agile DevOps processes and continuous delivery workflows [11].

Figure 5.12 shows the architecture for TeraFlowSDN (TFS), which is a cloud-native SDN controller composed of multiple container-based services using novel virtualization techniques, which are deployed as micro-services and managed on elastic infrastructure through agile DevOps processes and continuous delivery workflows. These micro-services structure an application as a collection of interconnected and related services using a common integration fabric. In a micro-services architecture, services are simple and detailed, and the protocols are lightweight. This architecture eases the introduction of novel services within the SDN controller that are able to provide NI. In this section, two functionalities are noted.

Figure 5.13 shows the first proposed functionality, which relates to traffic forecasting. SDN controller is able to monitor current network status and obtain timestamped traffic matrices on top of a pan-European core network (using Geant as an example [12]). Then, a traffic forecasting application (forecaster) can introduce

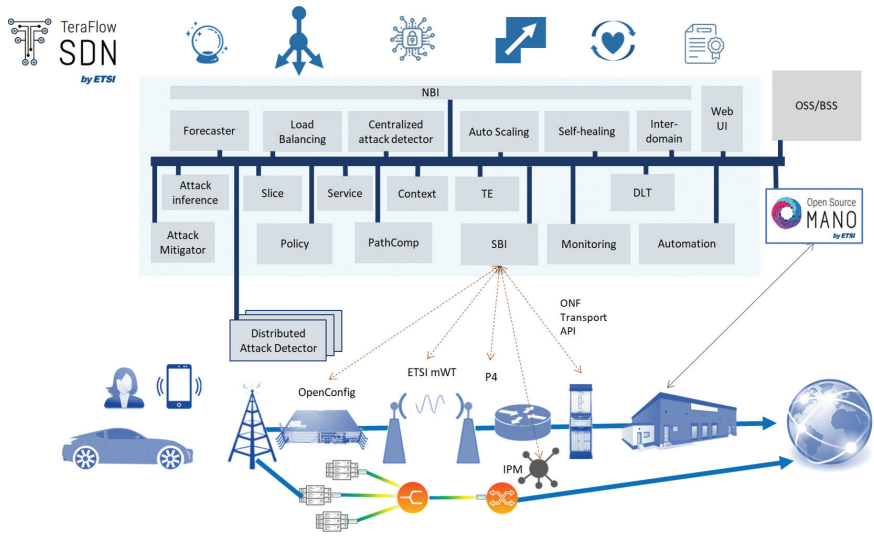


Figure 5.12. Enabling NI in the cloud-native SDN controller.

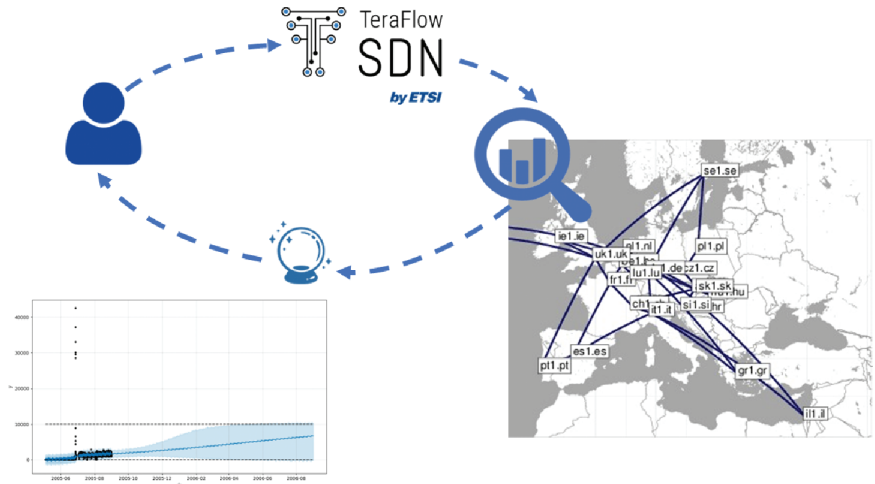


Figure 5.13. Traffic forecasting with SDN controller. Source of sample network: [12].

ML-based algorithms to predict the network status in the future. This information can be provided to the OSS/BSS in order to trigger necessary link updates, as well as provided back to the SDN controller in order to limit resources allocated to specific network links.

Forecaster is a novel TFS component that is able to perform proactive SDN traffic prediction (i.e., forecasts) by means of ML algorithms. For example, it is able to collect RT KPIs, such as link occupancy, and use ML algorithms to forecast

where and when a problem (e.g., unavailable link resource) is likely to occur so as to reroute traffic before it happens.

The interfaces provided by the Forecaster component allow to do a complete network forecast by introducing a topology identifier, or obtain specific forecast of a link, by introducing a link identifier. A forecast is a data structure that contains an array of timestamped predictions for a specified period of time for a specific link. Another provided interface allows to check if a new requested connectivity service will have resources available in the future. To this end, a link resource availability threshold is configured, and the forecast component provides a *ForecastPrediction* with the decision.

Multiple traffic forecasting libraries can be introduced in order to provide the component with multiple engines. One of the aforementioned possibilities is the Prophet library, which already includes a seasonal model for data forecasts. Another possibility could be the introduction of AutoML. AutoML does not require training, and thus, no previous modelling of the data is required.

Another features that TeraFlowSDN includes is the introduction of NI to face different threats on different planes. Management and control planes expose capacities that could be impacted by multiple vector attacks: from insiders (e.g., unauthorized configurations), to exposed interfaces (northbound, southbound, or east-west bound), to SDN applications. At the data plane, the TeraFlowSDN controller is exposed to both classical and advanced network attacks (e.g., DDoS, malware, traffic manipulation, physical-layer intrusions, etc.).

Moreover, the inclusion of ML components as applications of the SDN controller also adds a new threat surface that can be utilized by the so-called adversarial techniques that try to fool ML components by introducing small perturbations in the input that cannot be perceived by humans.

Both the massive amounts of information flowing through the network infrastructure and the very short latencies in threat detection impose limitations on current intrusion detection systems that perform ML-based network management. To cope with this problem, TeraFlowSDN includes a two-layer ML-based architecture with a central ML engine and ML-based threat detectors placed at the edge nodes.

The use of this advanced distributed architecture, supported by Google Remote Procedure Call Protocol (gRPC) telemetry data and network flows, will help to detect and mitigate the above-mentioned threats. In some specific data plane attacks, it is necessary to deploy distributed detection engines at the edge, with specific inference ML models that should be developed, to solve the attacks close to the origin.

The use of a simulated environment based on real NFV/SDN telecom infrastructures allows to generate traffic, train models, and deliver accurate inference ML engines to the edge and to the Cybersecurity net application for early mitigation.

To ensure the resilience of TeraFlowSDN ML models against adversarial attacks, multiple libraries have been included for designing defences in ML-based components and testing them against sophisticated adversarial attacks.

Finally, TeraFlowSDN includes un-, semi-, and supervised learning approaches for multi-layer network security monitoring, with additional embedded intelligence, using standard interfaces, containerization, and load balancing, while paving the way towards carrier-grade deployment of ML-based security monitoring [13].

5.4 Design Guidelines for NIFs

Artificial intelligent systems are irreversibly set on the evolutionary path of every future vertical as well as of every object and service we humans will interact with in the near future. This trend is motivated by the need to support elastic and demanding real-world use cases such as automated and cooperative mobility, e-health, gaming, entertainment, etc.

In this scenario, it is acknowledged that the telecom operators will have the opportunity to fill a central role in providing innovative solutions for application and service developers who need to combine the advanced capabilities of B5G and 6G networks with the fluid cloud-based application development processes emerged in the last decade (such as platform as a service (PaaS), continuous integration/continuous delivery (CI/CD) pipelines, and micro-service/serverless-based designs). A significant example of such an ecosystem are AI-enabled applications, which have become a major innovative force in almost any vertical and are being foreseen as one of the pillars that will boost the fourth industrial revolution. While great progress has been made during the last few years with respect to the accuracy and performance of AI-enabled applications, their integration into potentially autonomous decision-making systems or even critical applications requires E2E quality assurance, ubiquitous availability, and low latency. In the following, the reference model for NIFs is specified, as well as the implementation of some NI functions.

5.4.1 Reference Model for NIFs

The orchestration of such a new generation of AI-enabled application requires the introduction of new abstractions into the network service orchestration domain. Here, a novel concept of NIFs is introduced to refer to the AI-enabled E2E application sub-components that can be deployed across edge-enabled B5G and 6G networks. A new generation of AI-enabled applications will be obtained through the chaining of various NIFs across the various levels of the edge architecture. This will trigger the need for novel network automation platforms supporting all aspects of

network and service management, including the deployment and scaling of NIFs over a distributed facility as well as the various ancillary tasks that are to be performed to deploy such application, e.g., the creation of a new network slice.

To achieve this breakthrough, innovation is needed in two different areas: (i) the reference models for NIFs and (ii) the E2E orchestration of NIFs.

Reference models for NIFs: There is a need to define a reference model for NIFs, capable of capturing and representing the heterogeneity of NIFs at different levels of the technology stack. This can be developed as a network of interconnected modular ontologies, implemented in standard knowledge representation languages (e.g., OWL), following well-known state-of-the-art ontology engineering methodologies. Special attention needs to be put into describing NIFs not only from a functionality point of view, as available in other catalogues, but also considering other capabilities (e.g., computation, communication, storage, and hardware acceleration) that complement those of the NIFs, as well as any other aspects related to constraints necessary to support their dynamic orchestration. Similarly, the reference model shall also offer capabilities to describe methods and approaches to support the provisioning of the AI-enabled applications.

Building applications by chaining NIFs calls for the definition of a reference model for NIFs capable of capturing and representing the wide diversity of NIFs that will be deployed in B5G and 6G networks. This model is necessary to provide a common packaging for AI functions and a virtual marketplace for standardized AI components. There are several approaches in the literature to model artefacts (including datasets, pieces of software, algorithms, models, computational resources, etc.) to facilitate their understanding and reuse, as well as their potential chaining in complex applications. A relevant example is the workflow infrastructure conservation using semantics ontology network [15], focused on documenting computational scientific infrastructures, describing their relevant capabilities, and how they can be deployed, with four different ontologies: software stack, hardware specs, scientific virtual appliances, and workflow execution requirements. Other similar works in this direction are the software ontology (SWO) [16], a model for describing the software involved in the storage and management of data, with a strong focus on the biomedical domain or OntoSoft-VFF (ontology for software version, function and functionality) [17], focused on the description of the functionality and evolution through time of any software used to create workflow components. In addition to this, the state-of-the-art provides general-purpose metadata profiles and vocabularies that have been used for cataloguing different types of assets/entities: Dublin Core for all types of digital objects, DCAT and DCAT-AP for open government datasets, as well as its extensions, the DataCite metadata schema, and the research object model for scientific workflows.

E2E orchestration of NIFs: Deploying large-scale AI-enabled applications calls for the provisioning of resources across multiple administrative and technological domains. Therefore, it is of paramount importance to partition applications composed of multiple NIFs across multiple domains according to the requirements of each NIF. The proposed approach aims to cover more extreme provisioning scenarios in terms of hardware and software resources by starting from "de facto" standards for the orchestration of cloud and edge services, such as Docker and Kubernetes, and their emergent variants (e.g., FaaS). Acknowledging the heterogeneity and complexity of the currently available edge computing platforms, especially when it is as broad as the one foreseen in B5G and 6G networks, research innovative solutions for the E2E orchestration of the AI-enabled applications is needed. This goal can be achieved by leveraging novel network automation platform, in which monitoring solutions tackling the specific requirements of optimized hardware, edge devices, communication infrastructures, and cloud services will be taken into account. The monitoring subsystem will collect raw information to obtain the QoS indicators that provide insights regarding the correct behaviour of multiple NIFs orchestrated and linked to create complex AI-enabled applications. The quality indicators will cover traditional metrics for IT systems (e.g., performance) but also specific items to assess that NIFs operate according to their initial design. At the same time, the collected metrics and the quality indicators will ensure the capacity to oversee the operation of orchestrated NIFs and even to integrate AI-based functionalities to detect abnormal situations or to implement predictive maintenance.

While the NIFs paradigm opens the door for a number of opportunities for the customers, it also poses numerous challenges for the infrastructure providers including the need for supporting properties such as QoS, scaling, and fault-tolerance for the AI-enabled applications. This problem is exacerbated by the need for supporting stringent QoS requirements such as ultra-low latency and high data rate of novel applications and services envisioned in B5G and 6G networks. The AI-enabled applications force service orchestrating mechanisms to take into account additional factors, apart from the code and data locality, in order to satisfy diverse latency and data rate requirements by making appropriate placement decisions at MEC servers, which are expected to play a pivotal role in curtailing the round-trip delay of the applications. Extensive work on orchestration has been done in the past years by the service community and by the network community. An extensive survey on an E2E MANO in multi-technology and multi-operator 5G networks can be found in [18]. A study on multi-domain network and service orchestration is conducted in [19] highlighting its challenges in data-centre networks as well as in cellular networks.

5.4.2 Customized AI Techniques that Empower Practical NI

6G network will need to rely on customized instances on AI, which natively embody the NI (in the NIP). In the following, two exemplary solutions are discussed.

5.4.2.1 Avoiding the loss-metric mismatch in NI

Loss functions drive the training process of supervised machine learning models. In the vast majority of cases, loss functions are designed to be generic enough to work well in a wide range of scenarios. In regression problems, including forecasting tasks, mean absolute error (MAE), mean square error (MSE), and mean squared logarithmic error (MSLE) are common choices for expressing the loss.

However, in many practical cases in network management, such traditional losses do not characterize well the target performance metric of forecasting tasks. For instance, in anticipatory resource allocation problems encountered across mobile network infrastructure domains, the goal is to anticipate a capacity that is *sufficient* to accommodate the future traffic demand. Indeed, under-provisioning of capacity leads to the disruption of the offered service and violations of the service-level agreements (SLAs) with the service providers, while overprovisioning causes a more affordable squandering of resources. There, it is critical that the predictor learn to forecast a minimum quantity that is always above the demand.

Using a traditional loss function to perform forecasts in cases such as those outlined above results in a so-called loss-metric mismatch, where the regression objective, represented by the loss to be minimized, does not correspond to the optimization of the actual performance metric (Figure 5.14). As a result, the AI model that implements the regression model does not learn predictions that are aligned with the expected network management objective.

Thus, the NI native architecture defined above shall *support the use of customized loss functions that are carefully developed based on expert system knowledge*, i.e., a deep understanding of the network engineering or management task at hand, as well as of the variables that affect it and how they do so. This illustrates how the tailored design of loss functions for NI shall occur.

In the left plot (a), a pure traffic predictor is trained using a legacy loss, e.g., MAE or MSE for regression. The resulting forecast cannot be used as is, but serves as an

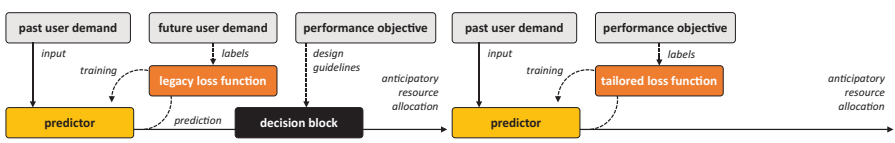


Figure 5.14. Different approaches for solving the loss-metric mismatch.

input to the actual decision block, which is manually designed by human experts to output the anticipatory MANO actions so that the target network performance objective is met. Yet, the decision block is agnostic of the inherent accuracy of the predictor and just trusts the forecast it receives. In the right plot (b), the novel approach proposed here is outlined: expert knowledge is used to directly design a dedicated loss that encodes the relationship between the prediction and the performance objective. As a result, the predictor is trained to produce forecasts that optimize the performance and can be directly used to drive the MANO actions. Importantly, the action decision is now aware of the unavoidable prediction error (e.g., lower accuracy in predicting small traffic volumes) and automatically compensates for it (e.g., by taking more conservative actions to accommodate small-traffic future demands).

5.4.2.2 Loss meta-learning for NI

The performance metric to be optimized by anticipatory MANO actions is not always known a priori by the network operator. This is the case, for instance, when the performance must be measured at the application layer (i.e., in the service provider domain) or when it concerns end user satisfaction (e.g., if it relates to mean opinion scores or quality of experience). In these situations, designing tailored loss functions is not possible, since the human expert (e.g., network manager or system engineer) does not know the exact relationship between the forecast and target performance.

6G networks shall leverage on innovative guidelines to deal with NI design in the complex situations described above. Specifically, instead of imposing a predefined expression of the loss function used to train the predictor, *a design of forecasting models is proposed that is free to meta-learn the loss function that best suits the network management objective at hand*. In practise, this is realized by combining a loss-learning block with the actual predictor, as shown in Figure 5.15. This block is responsible for learning the loss function, or, equivalently, capturing the relationship between the forecast produced by the predictor and the target management objective. Once

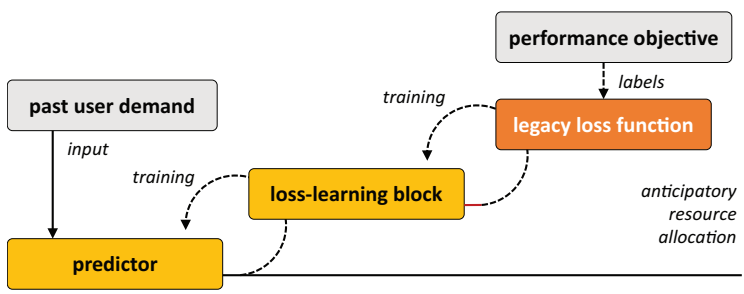


Figure 5.15. Loss meta-learning for NI.

ready, the loss-learning block can operate as a tailored loss function: it receives the output of the predictor and determines its quality for the precise management task. Therefore, it can be employed to train the predictor so as to steer the optimization of its parameters towards minimizing the actual MANO objective. The full design of the proposed framework is specified in [20].

5.4.3 Sustainable Decentralized AI Solutions

Sustainable decentralized AI-native architectures involve using energy-aware AI techniques for designing and optimizing 6G network slicing MANO in a sustainable and scalable way. The invoked architecture enables the next innovations:

- A FL-based AE is considered to reduce the amount of raw data exchange between local AEs and E2E AEs, making data analysis and prediction more energy-efficient. This technique avoids transferring the local dataset to a remote server, which leads to a high communication overhead and higher energy consumption.
- Decentralized deep reinforcement learning (D-DRL) techniques are considered in the DEs, such as multi-agent DRL (MA-DRL) and federated DRL (F-DRL). Such type of decentralized AI techniques can be utilized in the DEs in several ways: (1) To perform cross-domain joint VNF placement and energy control. The energy cost could be added in the denominator of the DE multi-objective reward function along with latency, while the throughput can be maximized by plugging it in the numerator with the costs. The multi-objective weights are fine-tuned according to operator/tenant priorities that depend on the slice type and business strategy. (2) Traffic-aware local decision agents dynamically placed in the network. These federated decision entities tailor their resource allocation policy according to the long-term dynamics of the underlying traffic, defining specialized clusters that enable faster training and communication overhead reduction.

5.4.4 Implementation of Intelligent Distribution from the Computation Perspective

From a computational point of view, the network can be seen as an entity consisting of a large number of computing nodes interconnected via communication links. The nodes are not homogeneous but range anything from high-performance data centres and edge servers down to user equipment and miniature IoT computing devices. Some of them have a more complicated internal architectural structure, featuring multiple processor cores, CPUs, processor cards, and clusters of them with

internal communication channels, as well as memory and input/output devices. Also, communication links are typically heterogeneous. In addition, neither the computational workload of the network nor its hardware components are constant but alter more or less frequently all the time, depending on the prevailing user data traffic distributions and acts of service providers. Let us call computation here parallel if it happens simultaneously in multiple units within a node and distributed if multiple nodes are involved.

In order to get the best performance out of the network, the computation needs to occupy intra-node units and be distributed among the network nodes so that the overall computing capacity would be maximized and the resource usage (e.g., time, energy, design effort, etc.) and cost would be minimized while ultimately supporting the designated use cases. A typical network node alone has more than a thousand design parameters, not to mention the plurality of nodes and fitting the subsets of the computation into them. It is therefore evident that determining the optimal distribution is a demanding multi-target Nondeterministic Polynomial (NP) complete problem, which cannot be solved with the current technology. The current 5G systems approach distribution by relying on simple offloading functionality from UE to the network, virtual machines allowing flexible allocation of resources in the cloud, containers reducing the state of computation, multicore CPUs allowing (constrained) parallel processing, and a programming paradigm relying mostly on independent sequential components and asynchronous execution of threads.

To boost the more macroscopic 6G network-level architectural design decisions, the hidden capability of low-level architectural techniques would be worth considering. For more flexible and efficient placement of computation, an architecture utilizing the following techniques to overcome the limitations of 5G network and processing solutions could be potentially applied (see Figure 5.16).

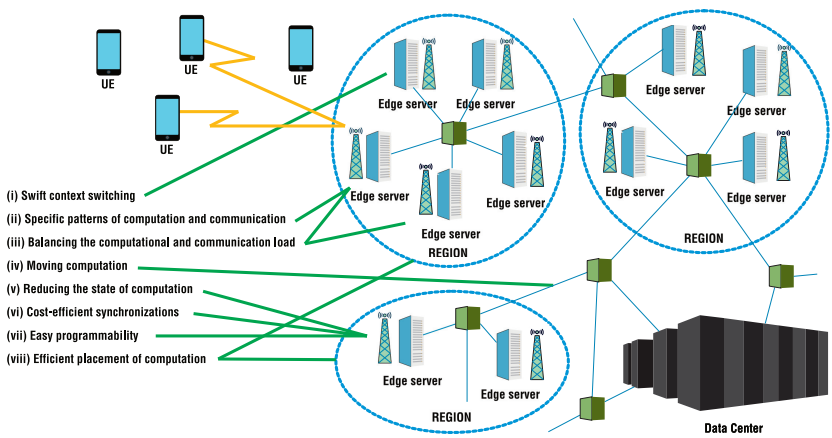


Figure 5.16. Intelligent computation distribution with low-level architectural techniques.

To support parallel and distributed execution, a programmer and the system need to divide the computation and related communication into parts (threads, processes, components, and subservices) that execute in parallel or concurrently in multiple execution units and communicate with each other. The number of computational (software) parts often exceeds the number of (hardware) execution units, introducing the need for multitasking, i.e., sharing units between the parts. In order to avoid efficiency issues, multitasking requires (i) swift context switching between the parts targeted to the execution unit. Context switches can be made faster, e.g., via multithreading [21] that uses on-chip resources to retain multiple contexts and allows for instant switching between them. If computational parts of a single service or task are executed in multiple execution units, there is a need to perform intercommunication between these units that can form (ii) specific patterns of computation and communication. Execution of these patterns can be accelerated by techniques such as multicasting and multioperations [22]. Multioperations are primitives of parallel computation where threads perform reductions, e.g., additions, on values provided by multiple threads into a single value in a constant number of steps. Often, in the case of non-uniformly distributed computing tasks, there is a need for (iii) balancing the computation and communication load between the execution units in order to manage hot spots and overcome the capacity limits of a single unit. Techniques including work sharing and work stealing between the nodes, e.g., edge servers, close to each other can be considered requiring (iv) moving computation between the execution units. Since current techniques using virtual machines and containers require moving a substantial amount of data from the source unit to the target unit as well as interrupting the execution, saving the state, and restoring it after the movement, techniques such as grouping a number of neighbouring nodes into a distributed computer would be useful to reduce the need for data block movements. Since the delay caused by moving computation is proportional to the amount of data moved, (v) reducing the state of computation can increase the performance. This is possible, e.g., by grouping homogeneous computations into flexible vector-like entities with one control flow but multiple data flows, such as thick control flows [23]. If there exist dependencies between the parallel parts, one needs to perform synchronizations to obtain the correct results. This is expensive in current multicore CPUs. A possible way to support (vi) cost-efficient synchronizations is to use the wave synchronization mechanism overlapping synchronizations with computations.

Distribution of computation can be investigated from the perspective of usability or programmability, which is now receiving more attention as a KPI for emerging 6G systems. Methods of (vii) easy programmability for parallel functionalities within the nodes include, e.g., shared memory emulation [24, 25], supporting unified and synchronous memory access with strict memory consistency and

eliminating many complications of current multicore CPU programming. Finally, the algorithms for (viii) efficient placement of computation contribute to efficient computation distribution, which makes the right data to be in the right place at the right time with minimal cost. Up to the node level, the placement of functionality and data could be done explicitly by the programming or with the help of simple hardware mechanisms, such as symmetric multiprocessing. If the node is a high-capacity data centre, more complex techniques are needed for optimum allocation of computational resources due to limitations on the scalability of the legacy techniques and the communication bandwidth between nodes.

To avoid excessive synchronization and communication delays, only nearly independent components should be distributed to separate nodes. A good strategy could be based on simple greedy algorithms placing the computation into a node with the lowest workload within the region of neighbouring nodes or more complex ML-based solutions. In this case, although computation and data placement throughout the entire network would not be feasible, the hierarchical placement for a certain range of networks (e.g., regions of edge servers, subnetworks of a certain size) would be possible in a given environment (i.e., energy limit, the number of users, existence of physical defects, and traffic congestion).

5.5 A Multi Agent Reinforcement Learning Framework

This section provides an overview of the integration of RL, a subset of machine learning, into radio networks. The advancements in device and communication protocols in 6G have enabled greater automation in managing the Core-RAN-Edge network component. The objective of RL is to maximize long-term performance by automating a complex and dynamic system. This involves multiple network segments with different scales and characteristics, including the nature of the controller, decision-making periods, and centralized or decentralized nature. As a result, the MA-DRL [26] scheme may be utilized to support the learning and deployment of RL-based agents in managing 6G systems.

5.5.1 Design Concepts of the MA-DRL Scheme

Reinforcement learning (RL) is one of the main machine learning paradigms (Figure 5.17). An RL agent is an actor in a dynamic system/environment (for instance, the RAN in 6G); it can interact with and alter the system and eventually learn through interaction data with the system. Once trained, the RL agent continuously acts with the objective of maximizing a long-term performance. RL

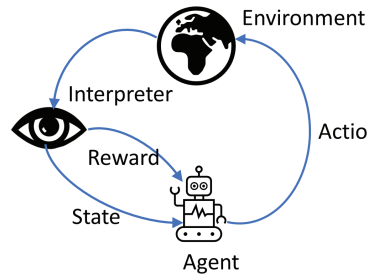


Figure 5.17. The reinforcement learning interaction loop.

is researched in many domains such as games (board games and video games), robotics, telecommunications, etc.

RL models the system as a Markov decision process (MDP). The configuration of the system is represented as a state in a state space S. The set of decision/actions the agent can take in the environment is A. A transition model, $P_a(s)$, describes the effect of the agent's action $a \in A$ on the environmental state $s \in S$. A reward signal, $R_a(s)$, represents the positive or negative feedback the agent receives after it takes an action a in a state s. In order to be applied to 6G, the system of application will need to be described as a MDP by defining the set of S, A, P, and R.

RL is composed of various families of approaches that take into account various characteristics of the problem.

- Multi-agent RL (MA-RL) deals with systems where multiple actors simultaneously compete or cooperate to solve a task. The multi-agent setting is motivated by problems where the real world imposes some communication or sensory constraints that force the agents to act in a decentralized way once deployed in the world.
- Specific RL methods deal with systems where the state is only partially observable. In a 6G system, some components might need to take decision without having a full and complete information about the system.
- Off-policy methods can learn an RL agent while executing the decisions of another fixed, predefined agent (that may not be RL related). This is of interest if one wants to learn about RL agents without having them interact in a hazardous way with a sensitive system while learning.

Traditional RL algorithms apply to small and discrete problems but do not scale when the number of observations or actions increases. The combination of RL and deep learning has somewhat reduced this issue. DNNs are a powerful and flexible tool to learn different levels of abstraction from data. The ability of DNNs to deal with high-dimensional data is game-changing to scale RL to more complex and eventually real-world problems.

5.5.2 The MA-DRL Scheme

This section outlines the functional implementation of the MA-DRL scheme in a 6G system. The proposed logical architecture, as already discussed above, divides the system functions into multiple major logical elements, including the cognitive plane, the monitoring plane, and the control plane [26].

The *monitoring plane* collects and monitors functional and performance parameters, including KPIs concerned. In this monitoring plane, there are two RL components: the *RL reward monitor* and the *resource analytics*. The RL reward monitor translates the feedback from the 6G system into a reward understandable by the RL scheme. The resource analytics is a component that analyses the feedback from the 6G system and is responsible for detecting special events such as non-stationarities.

Based on the inputs from the monitoring plane, the *cognitive plane* analyses the system and makes decisions to offer intelligent operational strategies such as optimizing resource allocation and deployment inputs to the relevant RL-driven modules such as RAN slicing, cell-free (CF) scheduling and beam steering. Most of the RL elements are therefore located in that cognitive plane. The RL simulator implements the training environment (mimicking the 6G system). The *RL agent training host* performs the RL training. The *RL agent designer* is a database of trained agents and training configurations. Finally, the *RL engine* manages the three previous blocks.

The cognitive plane outputs its intelligent decisions to other planes for actions to be applied. The control plane is called on demand to enforce RT actions decided by the cognitive plane via the *RL policy adapter*. The RL adapter is in charge of translating the MDP actions into proper control actions specific to the RL application.

For reference, the components of the MA-DRL scheme within the proposed logical architecture are reported in Figure 5.18.

On-line versus off-line learning

This section presents a comparison between the offline and online training workflows. The offline workflow involves agent training on a simulated environment, whereas the online workflow involves the agent directly training in the actual infrastructure.

The online and offline deployments emphasize some challenges of the application of RL to 6G systems, such as:

- RT control: the agent must compute its action at the required control frequency.
- Safety: the agent's actions must not be harmful to the overall system. In the online deployment, the agent should be able to perform safe exploration.

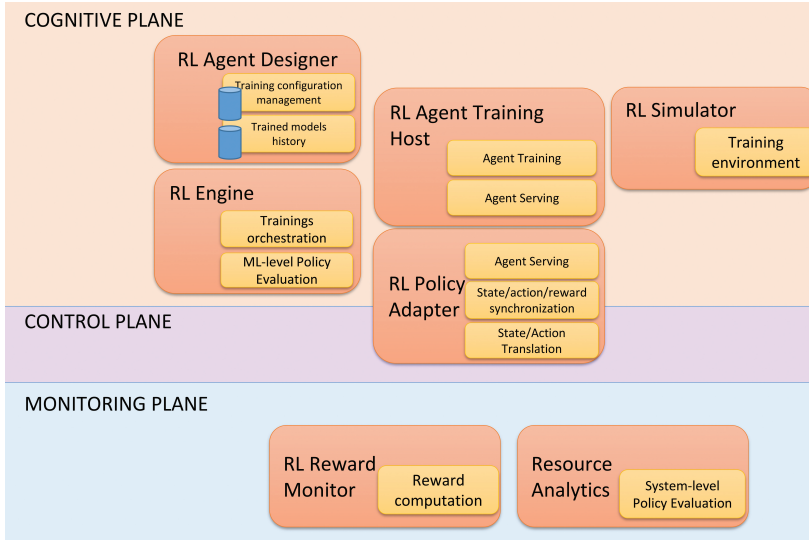


Figure 5.18. Overview of the MA-DRL scheme components.

- Robustness: the MA-DRL agents should be robust to inputs with small deviations (e.g., noisy sensors) to their training data.
- Dynamic environment: the agent should adapt to inputs whose dynamics may vary over time (see non-stationarity).

Non-stationarity, trust, and domain of validity

In addition to the above discussion, the building of a tool to instill confidence in the deployed RL agent is explored. This is accomplished by equipping the MA-DRL scheme with monitoring functions.

The validity of the trained policy deployed in the 6G system needs to be assessed by defining system-level policy evaluation rules, which will trigger alerts to the RL engine. The resource analytics component should implement these rules. If the rules exploit KPIs collected from the monitoring plane, the interface between the resource analytics component and the relevant monitoring components needs to be implemented. In some cases, the reward used during training can provide valuable information during exploitation that could be used to raise alerts or for monitoring purposes. Therefore, the reward computation from the monitoring plane needs to be implemented in the RL reward monitor component.

Finally, the handling of these alerts by the RL engine and RL agent designer will be defined. A simple approach could involve the agent designer randomly selecting trained agents until one is found that no longer triggers alerts. A more advanced approach would entail identifying a matching agent trained in a similar environment configuration based on the KPIs reported by the alerts. In this case, the

relevant KPIs and potentially threshold/tolerance values for configuration matching need to be specified.

5.5.3 MA-DRL for Joint Slicing Scheduling

This section provides an illustrative example of how the MA-DRL scheme can be utilized in an RL application by demonstrating how a joint-slicing scheduler problem can be formulated as an RL problem and incorporated into the MA-DRL scheme (Table 5.3).

Table 5.3. System components of the MA-DRL scheme.

Components	Specifications
MDP	The state space consists of three stats at each UE, namely: *WBCQI*, *Queue Length*, and *Delay of the First Packet.* The action space is the number of physical RBGs assigned to each UE, constrained by the total number of physical RBGs. The reward is a weighted sum of *Throughput*, *Latency*, and *Packet Loss* with the weights subjected to further sensitivity analysis.
Policy Adapter	• Translating between JSONs on the system side from-and-to the tensors on the agent side of the gym interface. • Time synchronization so that the corresponding <*States, Actions, Rewards*> are not messed up.
Reward Monitor	Need for tools to analyse the distribution of rewards over time under different weights for sensitivity analysis.
Training Host	Use vanilla policy gradients with pure MLPs for policy representation. All training is done offline.
Simulator	The simulator consists of: • OpenAI's gym interface sitting between the DRL agent and the RabbitMQ message queue. • The OAI's system with FlexRIC wrapping OAI's 5GNR sitting behind RabbitMQ • A telnet that programs OAI's 5G-L2SIM to replay the provided CQIs and traffics trace as UEs to the OAI's 5GNR base station. The four components (DRL agent, gym interface, RabbitMQ, and FlexRIC) are interacting as described in the "training scheme" section. **Interfaces** The gym interface will provide callable Python functions (e.g., **.step()**) on the agent side to enable the use of standard DRL implementations. The gym interface and the OAI's system will communicate through the RabbitMQ message queue.

This use case addresses connectivity in factories in the context, where more machines with diverse operational modes will be integrated together, making QoS-guaranteed wireless connectivity desirable. To this end, they address the challenge of supporting multiple connectivity "slices" with diverse requirements and ask how to do it optimally.

This RL application is tailored towards a deployment in a Bosch factory where the technical requirements of the two slices are diverse: the PLC offloading and video processing offloading:

- Slice PLC: Delay requirement is tight while data rate is normal.
- Slice video processing: Delay is not as tight but uplink data rate is high.

In the formalization, an RL problem will be solved for different slicing configurations, and the best configuration will be then selected. The specific components of the system in the following table.

5.6 AI-Driven Air Interface Design

AI-driven air interface design is a novel methodology for designing RANs, and it is envisioned to be one of the enabling technologies of the 6G intelligent networks. It relies on artificial intelligence techniques to learn single functionality or multiple functionalities in a transmitter or receiver of a radio communication link from training data. This chapter provides sample AI-driven air interface designs targeting hardware impairment compensation or mitigation, spectral efficiency enhancement, channel estimation, and radio resource allocation. The section is organized as follows: sub-section 5.6.1 presents an AI-driven receiver method for radio frequency (RF) hardware impairment compensation; sub-section 5.6.2 presents a fully learned receiver; sub-section 5.6.3 is about AI-native air interface design with constellation shaping to mitigate hardware impairments; sub-section 5.6.4 addresses AI-native channel estimation; sub-section 5.6.5 presents AI-driven channel estimation for reconfigurable intelligent surface (RIS)-enabled networks; and sub-section 5.6.6 presents AI-driven resource allocation methods in cell-free MIMO networks.

5.6.1 AI-driven Receiver Methods for RF Hardware Impairment Compensation

Data traffic over cellular communication networks has an increasing trend, which imposes requirements on the 6G radio cellular networks to support it even higher than their 5G counterparts. However, developing techniques to fulfil these requirements in the presence of limitations in RF hardware is challenging. Signal

transmission is subject to distortions due to RF hardware impairments which have an even higher impact in high-throughput scenarios. For example, the wideband transmission is severely impacted by power amplifier (PA) non-linearity; high-band transmission (e.g., sub-THz communication) is degraded by oscillator phase noise; and high-throughput transmissions are limited by the quantization noise of digital-to-analogue converters (DACs). Therefore, it is required to develop techniques to compensate the impact of RF hardware impairments and enable high-throughput communication under extreme conditions.

Several solutions have been investigated to reduce the impact of hardware impairments, e.g., improved oscillator circuit design to reduce the phase noise which increases overall hardware cost and adjusting the operating point of the PA by applying power backoff which reduces the energy efficiency of the transmitter and reduces coverage. In addition, there have been techniques to linearize the transmitter, e.g., by performing digital pre-distortion (DPD). These techniques usually require an estimate of PA characteristics based on a feedback loop using a sampling receiver circuit at the transmitter side to measure the PA's output and down convert it to a baseband signal for signal processing to estimate PA's characteristics. The DPD performs pre-distortion of the signal based on the estimated PA's characteristics so that the overall RF-chain resembles approximately linear characteristics. The DPD processing is overall complex and energy consuming task, especially for wide-band signal transmissions.

Artificial intelligence techniques enable a new design paradigm, where instead of mitigating RF hardware impairments or trying to linearize the transmitter, a receiver can be trained to learn signal detection in the presence of RF hardware impairments. An AI-empowered receiver based on artificial neural networks (ANN) is proposed in [27]. The radio transceiver architecture with AI-empowered receiver is outlined in Figure 5.19. The AI-empowered receiver learns the optimized demapper to compute soft bits as input to the soft decoder (low-density parity-check (LDPC)) in the presence of RF hardware impairments (e.g., phase noise or PA non-linearity). The demapper can be implemented using a fully connected NN with real

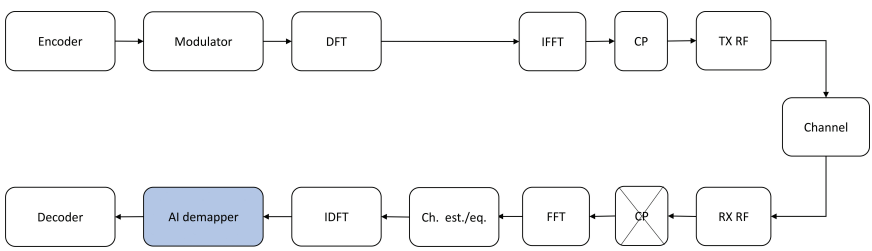


Figure 5.19. The radio transceiver architecture with AI-empowered receiver.

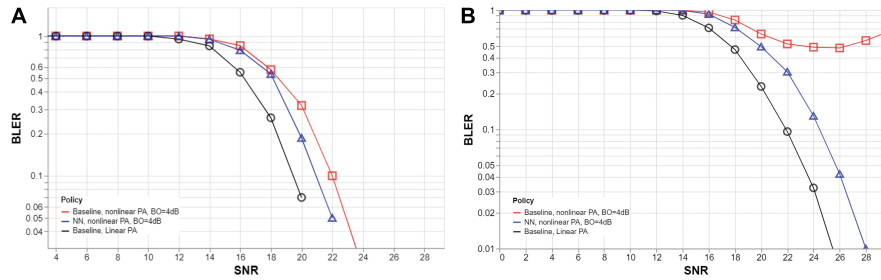


Figure 5.20. Performance of the AI-empowered receiver.

and imaginary parts of the equalized received signal and signal-to-noise ratio (SNR) estimate as inputs and the log likelihood ratio (LLR) values corresponding to the transmitted information bits as outputs.

The simulation results for the block error rate (BLER) performance of the AI-empowered receiver with a demapper composed of NN with five hidden layers each with 64 neurons are shown in Figure 5.20. For 64QAM-modulated signals with modulation and coding scheme (MCS) index 19, the AI-empowered receiver achieves roughly 1 dB performance gain compared to the legacy receiver method at 10% BLER operating point. For 256QAM-modulated signals, even for the lowest defined code rate in the standard (MCS index 20), the legacy receiver cannot successfully detect the signals and fails to reach 10% BLER. However, the AI-empowered receiver provides satisfactory results without any visible error-floor. This confirms that AI-empowered receiver can enable using higher-order modulations while providing signal reception with desired reliability, hence they can be used in high-throughput scenarios where higher-order modulations are desired.

The achievable throughput for the AI-empowered receiver and the legacy receiver with link adaptation at the transmitter is shown in Figure 5.21. The legacy receiver in the presence of linear PA provides an upper bound on throughput. In the presence of PA non-linearity, the legacy receiver achieves lower throughput compared to the upper bound, and the performance gap becomes larger at high SNR. The AI-empowered receiver achieves higher throughput compared with legacy receiver. The performance gain is larger at higher SNR values, and the achieved throughput saturates at the same level as the one for the upper bound on throughput. Hence, the AI-empowered receiver can partially compensate the impact of distortions due to PA non-linearity of a transmitter and can increase throughput and/or extend the coverage of a communication link.

The AI-empowered receiver can be used to relax the requirements on inband distortions (e.g., the requirements on maximum error vector magnitude, EVM) and the linearization methods to be performed at the transmitter side, e.g., DPD. The receiver method can be used in uplink communication of a cellular network,

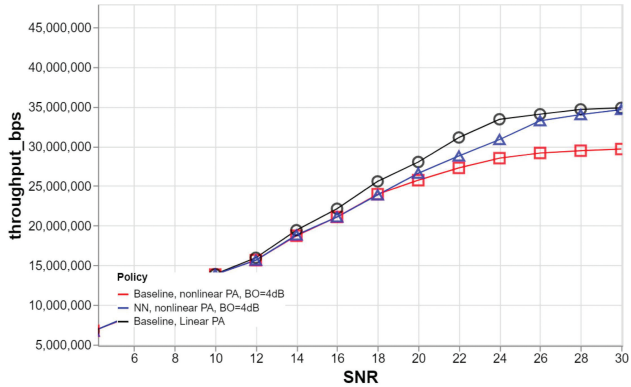


Figure 5.21. Achievable throughput with adaptive modulation and coding and for AI-empowered receiver.

which is usually coverage limited, and performing linearization techniques (e.g., DPD) at the UE is challenging, and it is desired to increase the energy efficiency of the UE to increase battery lifetime. Alternatively, the proposed method can be used in a downlink scenario to increase throughput and/or extend the coverage area of the base station (BS), enhancing the energy efficiency of BSs.

5.6.2 DeepRx: A Fully Learned Air Interface Receiver

A key component of the air interface is the receiver. Traditionally, the receiver has many processing blocks, including raw channel estimation from the reference signals, smoothing and interpolation, equalization, demapping, and decoding. The traditional receiver therefore splits the processing of the channel estimate (via known and received reference signals) from the processing of the rest of the signal and only combines them in the equalization step. This means that the receiver's performance is limited by the accuracy of the channel estimation, which is inherently limited, e.g., for high mobility UEs. This limitation can be circumvented by training the receiver algorithm from the antenna signal to the uncoded bits, as demonstrated in the DeepRx receiver shown in Figure 5.22. By also using the unknown received data, such an ML receiver learns an algorithm that can track the changes in the channel response in time and frequency, even in the case of sparse reference signals, leading to improved performance, as shown in Figure 5.23.

A similar DeepRx-type receiver can also be extended to support MIMO detection. Conventional MIMO detection algorithms contain an equalization block that relies on input-input multiplications. In order to train a NN to replace such algorithms efficiently, it is necessary to equip the NNs with the ability to represent input-input multiplications. Otherwise, the NN must learn to approximate them, which can be inefficient. The DeepRx MIMO [29] has different ways to represent

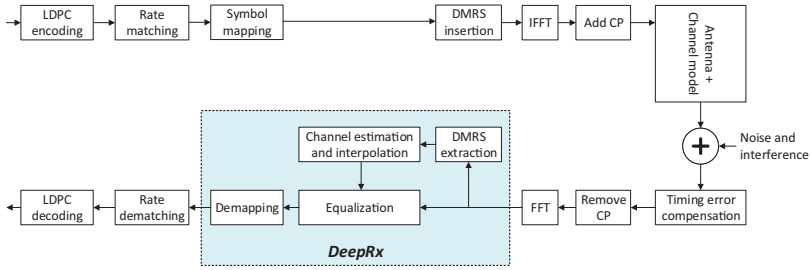


Figure 5.22. DeepRx replaces most of the processing blocks in the frequency domain receiver processing in the physical layer. The blocks inside the DeepRx box describe the traditional receiver processing blocks that DeepRx replaces with a trained neural network.

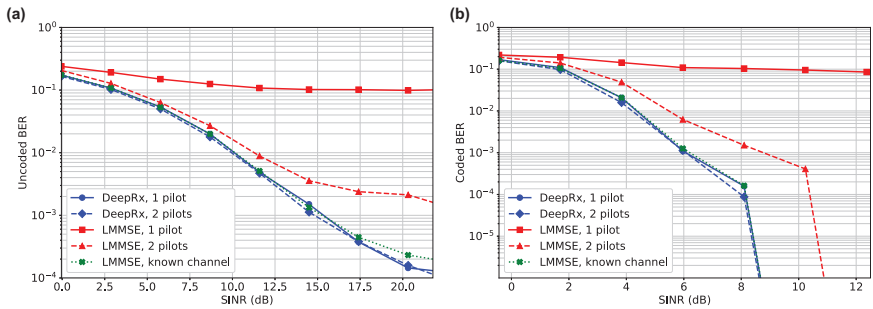


Figure 5.23. DeepRx SIMO reaches linear minimum mean square error (LMMSE) with known channel performance, even for highly mobile UEs of 0–130 km/h as in these figures.

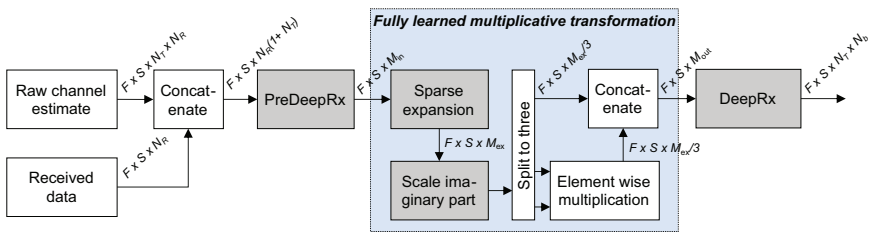


Figure 5.24. DeepRx can be extended to MIMO by adding the capability of input-input multiplication to the neural network, which allows more efficient learned equalization.

the multiplications. One example is shown in Figure 5.24, where the network learns which inputs should be conjugate multiplied and which should be multiplied. The performance results of such a DeepRx MIMO network are shown in Figure 5.25.

DeepRx shows that it is possible to learn state-of-the-art receiver algorithms based on data. Since the learned method relies on very few assumptions on the actual waveforms, it is possible to utilize this kind of receiver for other types of waveforms, e.g., ones that utilize novel constellation shapes.

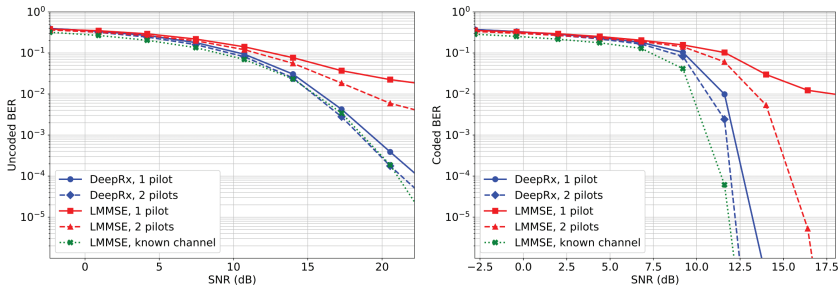


Figure 5.25. DeepRx MIMO can surpass the LMMSE receiver in TDL-E channels and reach near LMMSE with known channel performance.

5.6.3 AI-native Air Interface Design with Constellation Shaping and Hardware Impairment Mitigation

One step towards a truly AI-native air interface is to allow ML to design certain aspects of the physical layer. In practise, this means that the chosen properties of the air interface will be natively tailored for ML-based processing, e.g., on the receiver side. Ideally, this will result in enhanced spectral efficiency as the air interface is optimized with the help of ML.

In this approach to air interface design, ML is employed to learn the constellation shape and a transformation function for the time-domain waveform. These elements are learned in conjunction with an ML-based receiver, which has a similar architecture to DeepRx [30]. The benefits of such an approach are twofold. First, as opposed to 5G and earlier network generations, where the receiver must be provided so-called demodulation reference signals (DMRSs) for channel estimation and consecutive equalization, the proposed AI-based approach with learned constellation shape can operate without any pilots. This naturally removes the overhead they incur, as all the resource elements (REs) can be used to carry useful data. Second, the learned transformation of the time-domain waveform makes the whole system more resilient against PA-induced nonlinear distortion. Namely, using such a transformation, the power of out-of-band emissions can be reduced, while at the same time facilitating more accurate detection at the receiver side.

Figure 5.26 illustrates the overall approach on a conceptual level. In this example, an otherwise conventional orthogonal frequency-division multiplexing (OFDM) transmitter is assumed, with the exception that the bits are mapped to symbols using the learned constellation. In addition, as mentioned above, a convolutional neural network (CNN) is added to the transmitter's inverse fast Fourier transform (IFFT) output in order to ensure resilience against PA-induced nonlinearities.

The complete transmission link is trained E2E, which can be done with supervised learning. This requires a differentiable model between the bit source and the receiver output, including the transmitter, multipath channel, and the receiver.

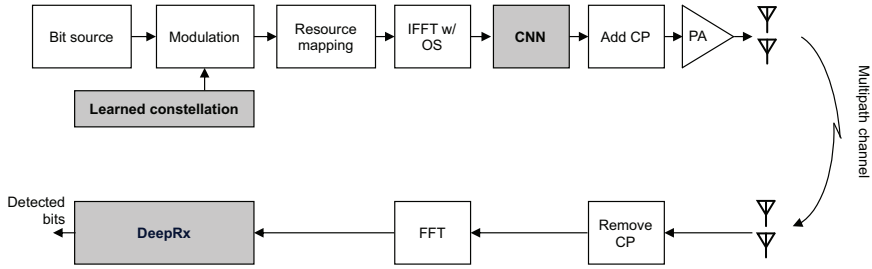


Figure 5.26. E2E learning for constellation shaping and out-of-band emission reduction.

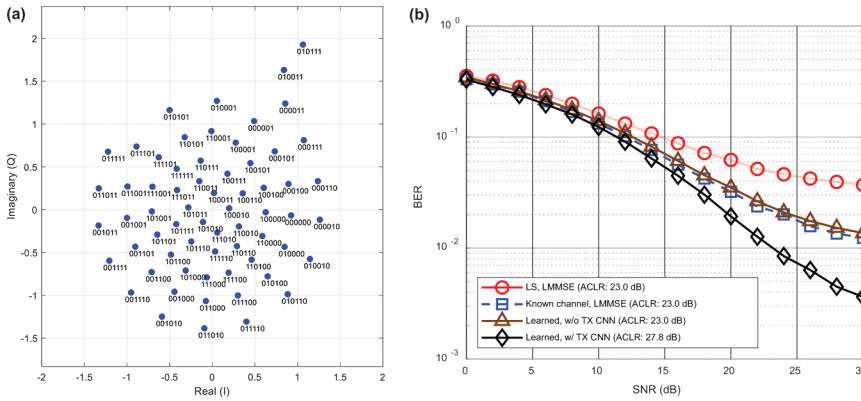


Figure 5.27. (a) An example of a learned constellation shape, and (b) the corresponding BER under a nonlinear PA.

Moreover, to ensure that a generic scheme is learned, the training is done using several randomized PA responses. Binary cross entropy between the transmitted bits and the detected bit estimates is used as the primary training loss function, which is equivalent to maximizing the throughput between the transmitter and receiver. In addition, the out-of-band emissions at the PA output are also added to the loss term, which incentivizes the transmitter to produce a waveform with as little emissions as possible. With this, the transmitter and receiver are incorporated into a single E2E model and can thereby learn jointly the constellation shape, transmission scheme for minimizing out-of-band emissions, and the required receiver-side processing for detecting the modulated bits.

Figure 5.27 shows an example of a learned constellation for a multipath scenario, as well as the achieved uncoded bit error rate (BER). The training is carried out with the TensorFlow library, where the full link is implemented as an E2E model. This also includes the channel model, which is based on precomputed channel coefficients generated under an urban micro (UMi) simulation scenario. In these

results, the transmitter PA is operating near saturation. The proposed ML-based solution is compared to two baselines:

- A practical receiver, which estimates the channel from pilots using least squares and interpolates it linearly over the whole slot (extrapolation of the channel estimate beyond the last pilot symbols is done with the nearest neighbour rule). The symbol estimates are obtained with LMMSE equalization, after which the soft bits are calculated using the log maximum A-posteriori (log-MAP) rule.
- A genie-aided receiver, which has perfect channel knowledge and performs LMMSE equalization followed by log-MAP demapping.

Both benchmarks utilize a conventional quadrature amplitude modulation (QAM)-OFDM waveform with two OFDM symbols per slot (14 OFDM symbols) dedicated to pilot transmissions. It is evident from Figure 5.27(b) that the learned pilotless scheme is superior to the benchmark schemes in terms of the achievable spectral efficiency, as it achieves a lower BER with less overhead. The pilotless operation is made possible by the learned asymmetric constellation shape, shown in Figure 5.27(a), which the receiver can use for blind detection of transmitted symbols. The lower BER of the learned scheme is mostly due to the nonlinear distortion within the RX signal, which the baseline schemes are not capable of suppressing.

To investigate the impact of PA-induced nonlinearities, Figure 5.28(a) shows first the BER of the different schemes under a fixed SNR for varying PA input backoff values. It is evident that the learned system with the TXCNN achieves the lowest BER when the backoff is small (this is the most nonlinear operating point

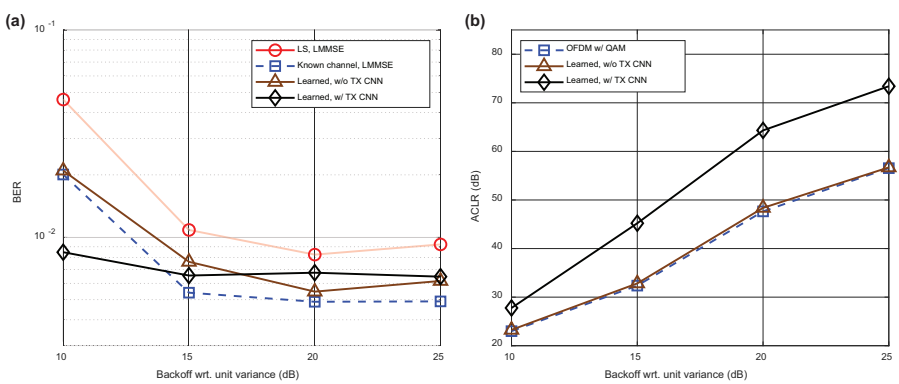


Figure 5.28. (a) BER of the different schemes with respect to the PA input backoff when the SNR is fixed at 24 dB, and (b) ACLR of the different schemes with respect to the PA input backoff. Note that here the PA backoff is defined with respect to unit variance, i.e., it represents the power of the PA input signal (not to be confused with PA backoff with respect to 1 dB compression point).

of the PA). However, when the nonlinearities are less severe, the benefit of the CNN diminishes, as expected. Figure 5.28(b) shows the adjacent channel leakage ratio (ACLR) of the learned waveform, compared to a conventional QAM OFDM waveform, measured at the PA output. Here, ACLR is defined as the ratio of the power of the in-band desired signal to the power of the out-of-band emissions. The effect of the TX CNN is again clear, as it achieves a clearly superior ACLR compared to the other schemes.

Altogether, these findings demonstrate the potential of an AI-native air interface in reducing BER, improving the spectral efficiency and reducing the signalling overhead over the current systems. Moreover, the benefits of ML-based processing when dealing with hardware impairments are also evident. The proposed approach facilitates more accurate detection of distorted signals while reducing the out-of-band emissions.

5.6.4 AI-driven Channel Estimation

Channel estimation (ChE) is among the first functions of a radio receiver that has been enhanced by AI/data-driven methods [57]. The ChE problem has been targeted by many tools stemming from AI.

For ChE problem, the minimum mean squared (MMSE) estimator is optimal in the Neyman-Pearson-Lemma sense. However, in practise, MMSE is not utilized widely, due to the following reasons: the first one is computational complexity. The MMSE requires inversion of the covariance matrix of the first input, which can grow with o(n^3). The second reason is the run-time sample complexity, i.e., the number of samples required to construct an accurate sample covariance matrix. For ChE, the processing delay that's required to accumulate pilots to build an accurate sample covariance matrix is not tolerated. Thus, an NN inspired by the mathematical representation of the MMSE estimator is proposed [57–59]. This NN consists of two layers that represent the MMSE estimator and a multiplication layer that passes the input to the estimator.

The proposed NN architecture is shallow and therefore more manageable for mathematical analysis, although it may not scale well with input size. If this NN architecture implicitly implements the MMSE, the well-known covariance matrix decomposition can be utilized, in which the large covariance matrix is represented in terms of Kronecker product smaller matrices. To handle data with spatial (vertical and horizontal), temporal, and frequency dimensions, the large input covariance matrix is decomposed into four smaller matrices. This enables the use of the same NN architecture on these four slices of data. At each stage, a slice of the 4-dimensional input vector is fed to an NN tailored for that slice of data. This has been portrayed well in Figure 5.29.

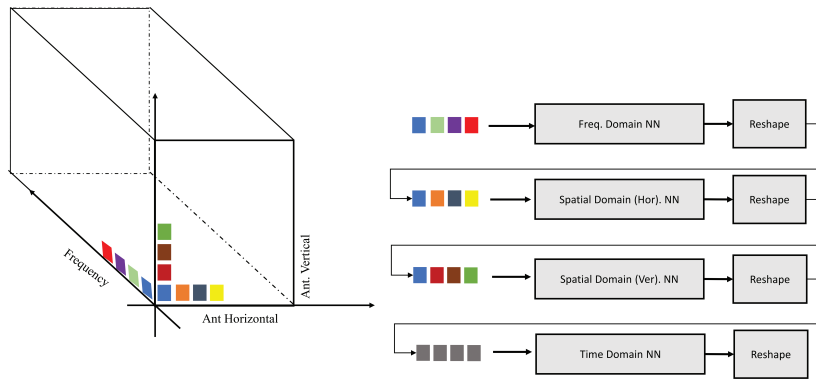


Figure 5.29. Data slices in the frequency and spatial directions of vertical and horizontal.

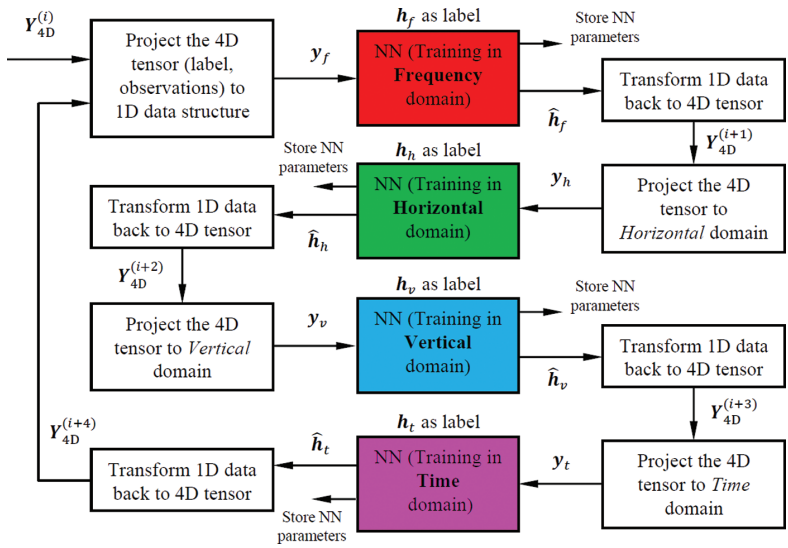


Figure 5.30. The neural network architecture consists of multiple smaller neural networks with reshaping operations in between. This design allows for processing large input vector size, without the need to large NNs.

The overall NN structure is depicted in Figure 5.30. For each of the NNs, the input data have to be reshaped to fit the NN. This reshaping acts as an additional re-shuffling, which further helps the generalization of the learned weights.

The performance of this NN is compared to that of the optimal MMSE solution. In Figure 5.31, the coloured graphs are the result of turbo-AI. The 3D turbo-AI illustrates three stages of turbo-AI, namely frequency, horizontal, and vertical, while the 4D turbo-AI benefits from all the dimensional processing. This further shows the trade-off for computational complexity versus performance in terms of normalized mean squared error (NMSE).

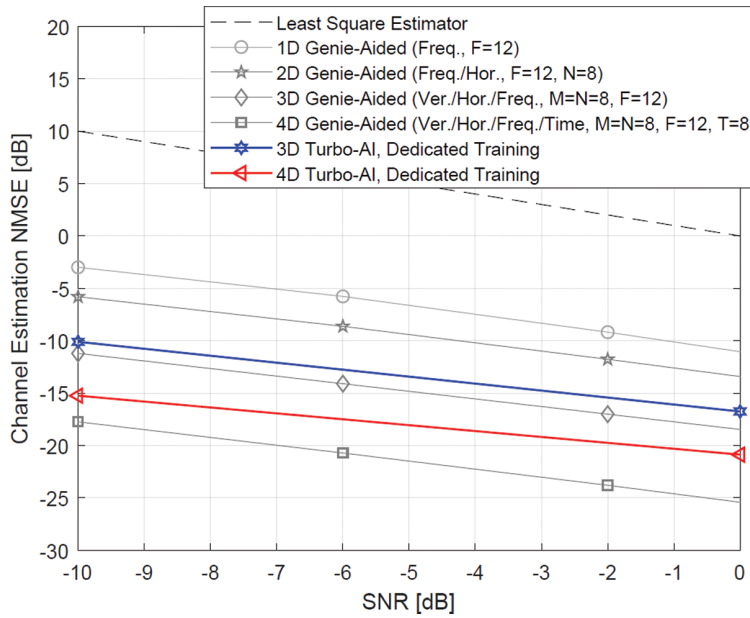


Figure 5.31. Comparison of the proposed ChE schemes, turbo-AI, with the statistically optimal solution MMSE-based estimator. The MMSE estimator for the above results included a 6144 × 6144 matrix inversion operation.

5.6.5 AI-based Sparse Channel Estimation for RIS-aided Communications Networks

A reconfigurable intelligent surface is composed of a large number of passive elements, each of which can reflect back incoming signals while incurring a phase shift. The wireless propagation environment can be reconfigured based on this property of RISs, and the phases of individual elements can be configured to optimize a certain objective function that corresponds to some aspect of system performance. However, most optimization solutions need channel state information to be known at the entity controlling the RIS (the assumption is that the RIS is controlled by the BS, where BS estimates the channel, calculates optimal phase shifts, and sends control signals to the RIS to configure the phase shifts). However, channel estimation is difficult as RIS consists of a large number of passive elements, which requires measurements at the BS with long pilot sequences due to large dimensionality. Also, it is necessary to estimate both the direct channel and the reflected channel through RIS.

The channel estimation for the reflected channel through the RIS requires the design of phase configuration of the RIS elements. It has been studied in [31], and an optimal codebook is proposed based on a minimum variance unbiased estimator. The dimensionality of the parameter space can be reduced by considering the

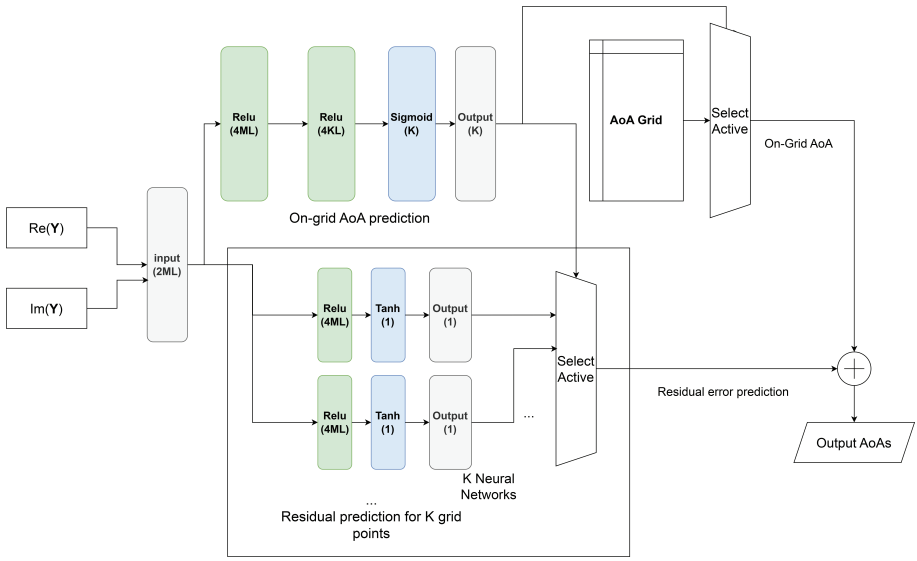


Figure 5.32. NN architecture for AoA prediction.

angular domain sparsity of the mmWave channels, resulting in improved accuracy with reduced pilot overhead. In [32], a sparse representation of the concatenated BS-RIS-user channel is derived, where channel estimation is converted into a sparse signal recovery problem. A double-structured orthogonal matching pursuit (DS-OMP)-based cascaded channel estimation scheme is proposed in [33], based on the double-structured sparsity of the angular cascaded channels. In this work, a channel estimation method based on the sparse representation of the channel is proposed, and angular parameters are estimated using a NN. The NN-based solution is a one pass method compared to iterative traditional sparse estimation techniques, while results show better accuracy compared to [31].

The uplink channel estimation of an RIS-assisted mmWave system is considered, where the direct link is assumed to be non-line-of-sight (NLoS) and the RIS has line-of-sight paths with both the user and BS. A compact representation for the RIS channel is derived, where the angle of arrivals (AoAs) is discretized. First, the case where AoAs lie exactly on the discrete grid (on-grid) is considered, and a sparse estimation method is proposed based on OMP. Also, the results are compared with NN-based predictions. Then, the case where AoAs can take any discrete value deviating from the discrete grid (off-grid) is considered, where the residual error from on-grid points is expressed separately. The NN architecture shown in Figure 5.32 is used for the prediction of AoAs, which consists of several NNs. The top NN is used to predict the on-grid AoA points, and the residual error is calculated by the NNs shown in the bottom. This work has been published in [34].

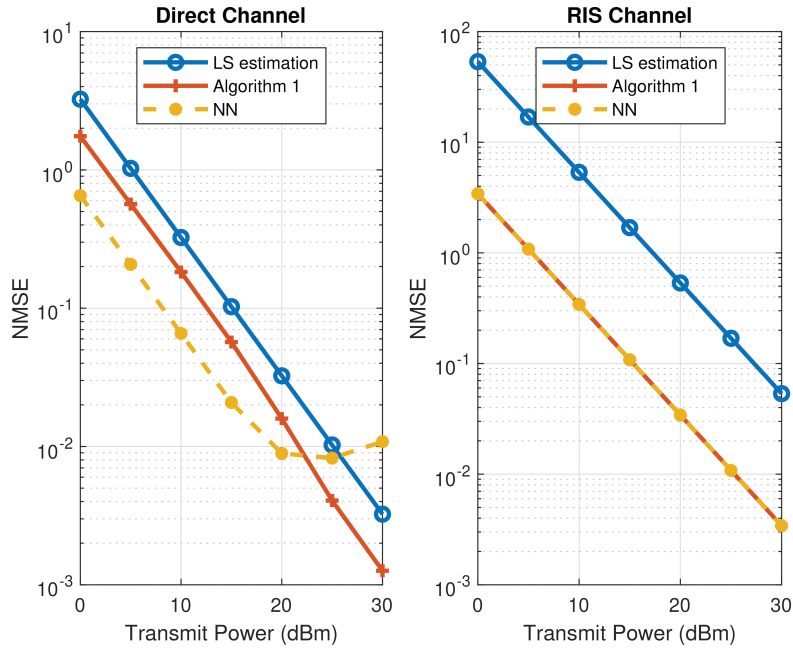


Figure 5.33. Comparison of performance of proposed methods with LS estimation for both direct and reflected channels, in the off-grid case.

Numerical simulations are performed to evaluate the results, and channel estimation error is considered as the KPI. Comparison of the performance of the proposed methods is shown in Figure 5.33, where the NMSE of direct channel and cascaded channel through RIS are considered separately. The performance of the NN is compared with least squares estimation and sparse estimation algorithms with and without perfect AoAs. Results show that NN outperforms all the other methods; however, a saturation effect is seen with increased transmit power due to leakage from grid imperfections.

5.6.6 AI-based Radio Resource Allocation for Cell-free Massive MIMO Networks

Proper radio resource management (RRM) is of much importance in improving the system performance in communication networks. However, the classical optimization or heuristic-based resource allocation algorithms have several challenges, such as high computational complexity, suboptimal solutions in complex and non-convex problems, lack of flexibility and parameter sensitivity, and inaccuracy of the model-based resource allocation methods [37]. Novel communication architectures such as cell-free massive MIMO networks and high-frequency communication systems have an increased system complexity due to the large number of

antenna elements in the transceivers and the increased AP deployment density for a high-frequency radio access technology. RRM becomes more challenging in such systems due to the increased system complexity and high dimensionality of the resource allocation problems. In recent literature, the learning capability of ML algorithms is exploited to overcome the above challenges associated with conventional approaches in complex communication systems [36–40]. Most of the existing studies focus on a supervised learning approach where a model is trained to learn the mapping between the inputs (user locations or channel statistics) and the optimal outputs (power allocations) obtained by an optimization algorithm.

On the other hand, an unsupervised learning-based resource allocation approach does not require the optimal resource allocations to be known during model training as in supervised learning; hence, it alleviates the need of generating a large dataset with thousands of samples by solving the computationally complex optimization problem [38]. Thus, it makes the data preparation and model training simpler, more practical, and more flexible since the deep learning model can be easily retrained in a changing environment over time. An unsupervised learning-based resource allocation scheme for a cell-free massive MIMO network is proposed here to learn the resource allocations in a data-driven manner with lower computational complexity than an optimization-based algorithm. Specifically, joint optimization of user power allocations and fronthaul capacity allocations (between CSI and data) to maximize the network sum throughput in an uplink of a limited-fronthaul cell-free massive MIMO network is considered. A DNN is directly trained using a custom loss function to optimize the sum throughput objective instead of training with labelled data. The large-scale channel coefficients between the users and the access points are used as the DNN input.

The loss function for model training is defined as the negative value of the sum rate, which is a function of large-scale channel coefficients and the trainable parameters θ of the DNN. It is differentiable with respect to the trainable parameter set θ which allows training the model via mini-batch gradient descent method. In each iteration of the training, a set of channel realizations is generated from its distribution, and the average loss is calculated over the mini-batch. During training, the model learns parameters θ to minimize the loss that maximizes the sum rate and outputs the power allocations and fronthaul capacity allocations. Furthermore, the DNN could be used in two modes: (1) offline training mode, where the model is trained offline for a large dataset with different channel instances; and (2) online training mode, where the offline trained model is retrained in each channel instance, allowing further customization and fine-tuning of model parameters based on large-scale channel inputs in each channel realization, to further optimize the sum rate performance.

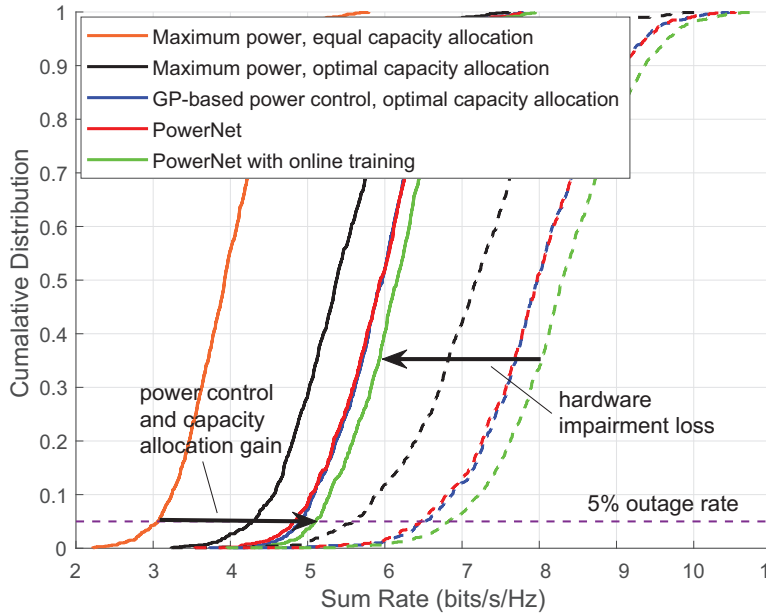


Figure 5.34. Sum rate performance comparison between proposed method (PowerNet) and baseline (optimization-based) with and without hardware impairments in the transceivers. Dashed lines: Ideal transmitters and receivers without hardware impairments. Solid lines: With hardware impairments in the transmitters and receivers.

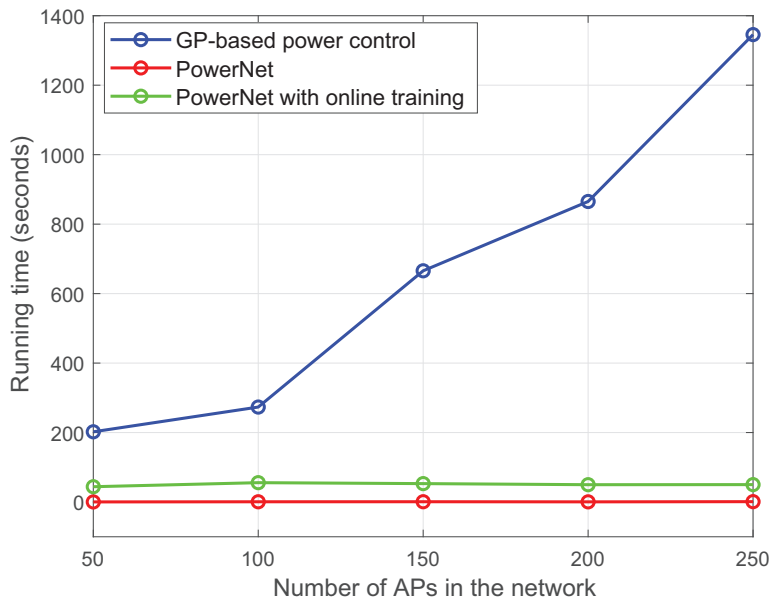


Figure 5.35. Recorded CPU timing for the CVX solver and the PowerNet to produce outputs for 100 channel realizations for different network configurations.

As seen from Figures 5.34 and 5.35, numerical simulations have shown the proposed ML approach to learn resource allocation vectors, resulting in similar sum throughput performance compared to optimization-based baseline while having lower computational complexity. The time complexity of the DNN does not drastically scale with the number of users and APs, as the optimization-based algorithm does. While the time complexity of the optimization-based algorithm exponentially scales when the number of APs or users is increasing, the time complexity of the DNN only increases slightly when the system parameters are scaled. In contrast to the iterative algorithms used in the optimization-based approach, the ML approach only requires one-shot output calculation, which only involves matrix multiplication and addition operations to produce the DNN outputs; hence, it has a significantly reduced time complexity of the DNN.

5.7 Statistical Federated Learning for Resource Provisioning

A paramount feature of 6G systems is its massive and highly heterogeneous network slicing, which enables tenants to not only target specific industries, but also bring digitalization to end users through the expected widespread adoption of novel, advanced, and diverse digital services (e.g., holographic communications, next-generation extended/virtual reality applications, tactile internet, industry 5.0). This fact will require 6G networks to manage a large number of slices, which could potentially span multiple technology domains, such as the RAN, edge, cloud, and core. This fact poses a major challenge to traditional centralized MANO methods, which are vulnerable to single points of failure and require extensive monitoring. This results in a large communications overhead and delayed, heuristic-based decisions, leading to increased energy costs and reduced scalability and sustainability for the network. To address this issue, sustainable and scalable 6G network slicing must rely on distributed AI architectures, where local computation tasks are performed closer to the monitoring points, reducing energy costs and exchanging only some local updates (e.g., the model weights or compressed/pre-processed data) or compressed statistics through federated and multi-agent learning strategies.

5.7.1 AI for SLA Management in RAN

AI techniques will be a crucial factor in the automation of resource allocation for dynamic network slicing and its associated SLA. In order to reach both scalability and sustainability in network slicing, AI-enhanced analytic algorithms must

be brought closer to the distributed monitoring points throughout the network. This approach significantly reduces the raw data exchange, allowing only certain AI model parameters to be transmitted for coordination or collaboration purposes. In such a scenario, FL-based analysis is an attractive solution that enables: (i) local learning at each network element, whether virtual or physical, by exchanging only the weights of its model with the aggregation server; and (ii) addressing the absence or inadequate distribution of local datasets by capitalizing on the knowledge of other elements participating in the FL task. The challenge in this case is twofold: first, to ensure that the outputs of the models learned offline conform to a predetermined statistical distribution, such as the slice-level SLAs, when deployed in a testing scenario; and second, to ensure that these models converge when faced with non-independent and identically distributed datasets in live network environments.

This section presents a new decentralized management and analytics framework designed to handle a massive number of dynamic slices in Beyond5G and 6G scenarios, enhancing scalability, sustainability, and responsiveness of self-management and self-configuration of network slices for ZSM. The framework aligns with both ZSM [42] and enhanced network management interface (ENI) objectives [43]. Additionally, the framework is implemented in three technological domains: cloud, edge, and RAN [44].

5.7.2 Statistical FL-based Policy for RAN

To demonstrate the scalability and ZTM capabilities of the aforementioned architecture, consider the deployment solution illustrated in Figure 5.36. It shows a disaggregated RAN architecture with a central unit (CU)-distributed unit (DU) functional split, in which each transmission/reception point (TRP) is co-located with its DU, which is connected to the corresponding CU using a fronthaul link. Each CU runs a VNF at the edge on top of commodity hardware and includes a co-located MS and an AI-enhanced AE, which are instantiated per slice. For each central unit i and slice j, the MS (i, j) performs the data collection to build a local (small) dataset D_(i,j) of size Di,j, consisting of a set of input features vector metrics x_(i,j)^((n)) (e.g., over-the-top traffic patterns, average channel quality indicator (CQI), average number of active users, etc.) and the corresponding outputs y_(i,j)^((n)) (e.g., physical resource blocks (PRBs) occupancy, CPU load, etc.). These local datasets are generally insufficient to train precise analytical models. As a result, local AEs participate in a FL task, where an E2E slice-level AE acts as an aggregation server. Note that TRPs are often deployed in areas with varying traffic patterns, both spatially and temporally, which are strongly affected by the user distribution and depend on the context in which the TRPs are deployed (e.g., residential areas, commercial districts, and entertaining events such as concerts or football matches). Furthermore,

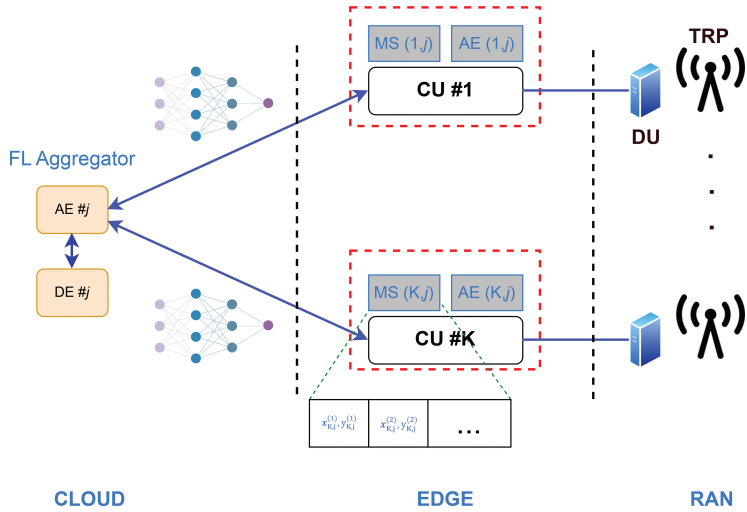


Figure 5.36. Cross-domain statistical FL-based with decentralized MS/AE.

the radio performance indicators are associated with time-dependent channel conditions. As a result of these aforementioned aspects, the local mini-data datasets are inherently non-independent and identically distributed.

The benefits of the proposed management framework can be demonstrated through a use case of slice-level resource prediction with SLA constraints [45]. The main objective is to respect the SLA violation rate for each slice while minimizing the total communication management overhead. A typical SLA is established between the tenant of slice j and the infrastructure provider, to ensure that the utilization of slice resources (e.g., CPU load or number of occupied PRBs) does not exceed the range $[\alpha_j, \beta_j]$ with a probability given by a specified upper-bound threshold δ_j. These SLAs can be defined in terms of percentiles and the empirical cumulative density function (ECDF) constraints and solved via a proxy-Lagrangian and two-player game [45]. In practise, the infrastructure provider and the tenant of slice j may agree that the Q-th percentile of utilization a specific resource must be lower than some given agreed bound to ensure isolation, where the Q-th percentile refers to the value below which $Q\%$ of the samples of this resource are distributed.

In order to prevent the exchange of raw monitoring data with the cloud domain, each AE performs SLA-constrained resource prediction locally. This involves learning an AI-based resource provisioning model, under long-term SLA constraints, considering space-time-varying input features such as slice traffic and radio conditions, which depend on the CU locations at the RAN and slice type. Given that the local MS datasets are not exhaustive, the local AEs participate in a FL task to enhance their predictions, acting as local clients. Only the j-th slice model weights and SLA violation rates achieved by the local models are communicated to the E2E

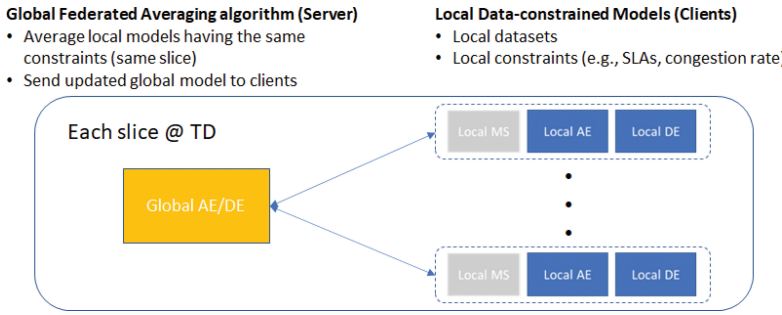


Figure 5.37. FL-based decentralized AEs for slice n.

slice-level AE, located at the cloud, which plays the role a FL aggregator. At each FL round, the E2E AE averages the received model weights and broadcasts the result to the local AEs that update their corresponding local resource provisioning models. The whole process is illustrated in Figure 5.37.

The algorithmic statistical FL (StFL) architecture and algorithmic framework for decentralized resource allocation have been presented in [46] and [45], respectively. As shown therein, the StFL solution enables SLA enforcement, while significantly reducing the overall network overhead, when compared to the centralized counterpart or with FedAvg algorithm [46]. Furthermore, as analysed in [44], the StFL algorithm achieves more than $\times 10$ times energy efficiency, paving the way for 6G sustainable massive slicing networks. To further reduce the communications overhead, minimize the computation and management of data, and ensure scalability under massive slicing, a subset of the active AEs can be selected in each FL round. Towards this goal, a stochastic policy has been proposed in [47] and [48] to select a subset of m clients out of the total set of K (where $m < K$) local AEs to take part in the FL optimization, based on their received SLA metrics. This approach uses these metrics to compute the violation rate probability distribution over the local AEs based on a softmin activation layer, and a stochastic policy is considered to randomly select the clients in the next round of the FL training based on the violation rate distribution. As a result, AE that reach lower SLA violation rates can participate in the FL round with a higher probability. In practise, the whole process can be summarized as follows: during each FL round t, the E2E DE selects stochastically m local AEs to participate in the FL task, based on the violation rate probability distribution, which is obtained using the softmin activation layer. Then, the E2E DE provides feedback in the form of an activation bit to inform the participating AEs. In this scenario, only the AEs selected would participate in the training process and transmit their weights to the FL aggregator that will average them. Finally, the resulting global model is broadcast to all the local AEs involved in the training. This procedure is repeated until convergence.

5.8 Network Slicing-Driven by Deep Reinforcement Learning

Network slicing enables multiple virtual networks to be instantiated and customized to meet heterogeneous use-case requirements over 5G and beyond network deployments. However, most of the solutions available today face scalability issues when considering many slices, due to centralized controllers requiring a holistic view of the resource availability and consumption over different networking domains. To address this challenge, a hierarchical architecture is designed to manage network slice resources in a federated manner [49]. Motivated by the advancement in DRL schemes and the open RAN paradigm, a set of traffic-aware local decision agents (DAs) is proposed to be dynamically placed in the RAN. These federated decision entities tailor their resource allocation policy according to the long-term dynamics of the underlying traffic, defining specialized clusters that enable faster training and communication overhead reduction.

Emerging use cases such as vehicle-to-everything (V2X) communication, the Internet of Things (IoT), and augmented/virtual reality (AR/VR) are some of the examples of 5G/6G use cases that need to co-exist on a shared physical infrastructure. However, the diverse demands for bandwidth, latency, and reliability increase the need for effective orchestration solutions to manage these services effectively and efficiently. Network slicing holds great potential as a solution to this challenging scenario. Network slicing creates an all-encompassing environment to underpin a plethora of network services by running fully or partly isolated logical networks, namely slices, on the same physical infrastructure. The zero-touch network is conceived as a next generation of network management that leverages the principles of NFV and SDN to be the cornerstone for supporting fully automated operations and on-demand configuration without the need for fixed contractual agreements and manual intervention. To handle these radical changes, the ZSM framework reference architecture has been designed by the ETSI [50]. It is designed to support fully automated network and service management. The architecture supports a set of architectural design principles, including:

- Modularity aspects for creating self-contained and loosely coupled services to prevent monoliths and tight coupling.
- Extensibility enables the network to extend new services and service capabilities.
- Scalability fulfils increasing or decreasing demands to deploy managed entities, and modules can be independently scaled.
- Resilience aspects cope with the degradation of the infrastructure and other management services as well as simplicity makes minimal complexity while still meeting the functional and non-functional requirements.

The modular characteristic is paired with intent-based interfaces, closed-loop operations, and AI/ML techniques to empower the full automation of the management operations. One of the essential building blocks in the ZSM is intelligent decision-making elements.

The fluctuating nature of traffic demand dramatically complicates the process of resource planning and allocation, particularly in the RAN domain. Resource allocation decisions, such as bandwidth allocation, must consider the added variability of the wireless channel and mobile users. Traditional RAN slicing solutions envision a centralized controller with a comprehensive, RT view of the network and its resources. However, this approach needs to work on scalability issues in actual implementations due to the large volume of monitoring information that must be exchanged and the high number of BSs. As a result, it becomes challenging to implement optimal resource allocation strategies in a timely and efficient manner.

Despite the innovative approach, it is still necessary to determine an effective way of managing slicing scenarios with many vertical services. In this regard, a hierarchical architecture is proposed for network slice resource orchestration. The framework considers the variable distribution of mobile traffic demands and sets up a network of local decision agents as virtual software instances within the NRT RAN intelligent controller (RIC). These DAs can access local RAN monitoring information and extract local knowledge without relying on a centralized entity for decision-making.

The framework utilizes a dynamic agent selection mechanism based on local traffic conditions similarity, allowing for more efficient information exchange and collaboration among local DAs. The benefits of this approach include: (i) resource allocation at the edge of the network, leading to more timely and accurate information; (ii) reduced control information that needs to cross the network to reach the central controller, reducing overhead towards the core network; and (iii) provisioning of FL schemes that enhance the capabilities of DAs. DAs will not only learn from a local observation space but also from information from other RAN nodes, thus improving the generalization of the learning process. The main innovation of this approach is based on utilizing the distributed RAN information to create a new type of specialized agents that work together in homogeneous clusters through a federation layer. This results in a scalable and stable decision-making process under rapidly changing traffic conditions. The proposed framework is also compatible with O-RAN.

5.8.1 Framework Overview

This solution is based on the slicing concept in mobile networks, where multiple network tenants share a portion, namely slice, of the common mobile network infrastructure. Each slice has dedicated networking resources to meet its service level

agreement (SLA). The focus is on the RAN domain, where the SLAs are expressed in terms of maximum slice throughput and transmission latency. The term transmission latency refers to the average time that traffic belonging to a slice must wait within the BS transmission buffers before being processed due to inter-slice scheduling procedures.

The mobile network infrastructure consists of a set of BSs with slices deployed on them. Each BS has a capacity expressed regarding PRBs of fixed bandwidth. The available PRBs must be divided among the slices according to their RT traffic demand and SLA requirements. It is assumed that each network slice has predefined latency and throughput requirements as part of the SLA between the network operator and the slice owner. The focus is on the RAN domain, and the latency is considered as the queuing delay experienced by traffic while passing through the scheduling processes of each BS. The system operates in time slots, which represent a decision interval. The PRB allocation decisions can only be made at the beginning of each decision interval, with an interval determined by the infrastructure provider ranging from a few seconds to several minutes.

The allocation of radio resources to end users is considered a two-step process, assuming the existence of a preliminary admission and control mechanism. First, once network slices are admitted into the system, the infrastructure provider schedules the allocation of radio resource slots for each tenant. Then, based on the slice's resource availability, each tenant may enforce its own scheduling solutions for its end users, depending on their use case or business requirements.

An F-DRL-based architecture is leveraged to handle the RAN slicing scenario. This architecture involves the local DAs operating as software instances within each BS, as depicted in Figure 5.38. The agents are responsible for enforcing slice PRB allocation decisions based on the local monitoring information received from the network MS. However, the distributed nature of RAN deployments and the varying spatiotemporal behaviour of mobile traffic traces make it challenging for an agent trained on complex and multi-variate monitoring metrics. To address the aforementioned challenges, an FL layer has been introduced to enable inter-agent communication and expedite the learning process through knowledge sharing.

The quest for intelligent and optimal control in massive telecommunication environments has aroused intensive research on the applications of DRL methods. The DRL can provide a promising technique to be incorporated into network slicing and solve the control and optimization issues. DRL combines RL and DNN to extract knowledge based on experience gained by interacting with the environment (network slicing). To solve slice issues such as admission control, congestion control, energy efficiency, resource management, service creation, dynamic network configuration, anomaly and fault detection, security, and reliability, the agent of network slicing should update the Q-function for optimal actions. Deep Q-learning

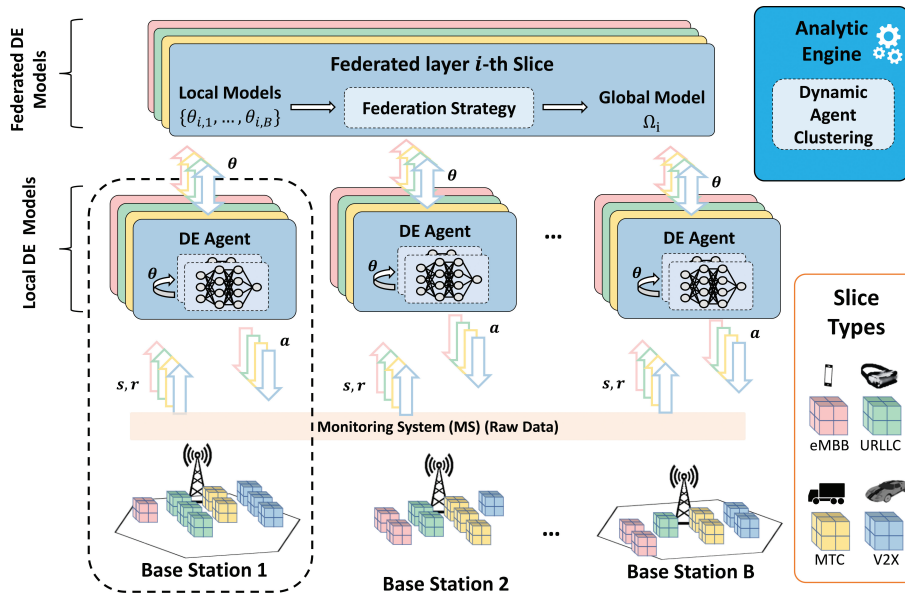


Figure 5.38. Generic federated DRL architecture for RAN slicing [49].

benefits from a NN to approximate the Q-value function. The state can be considered as input, and the Q-value of all possible actions is generated as the output. An agent (NN) in DRL generates the dataset on-the-fly, continuously interacts with a network, and is well-suited to problems with numerous possible states with high dimensions. This model-free method stores the experiences of each interaction with the environment in a replay buffer, which enables the network parameters to be updated without any prior knowledge of the environment's statistics. Figure 5.39 illustrates the local double DQN (DDQN) agent workflow.

5.8.2 Federated DRL for RAN Slicing

FL enables the training of machine learning models across decentralized entities that only have limited data access. Unlike multi-agent RL, which requires a centralized data source for all agents to observe, select actions, and receive rewards, FL allows for a collaborative learning process by aggregating multiple model updates. It results in a shared prediction model that is more refined, improves the learning rate, protects privacy, and offers better generalization. In the F-DRL-based framework, each agent trains a local DDQN model and shares its experience in the form of hyperparameters with other entities in the same federation layer. This iterative process leads to a global updated model that is stored in a cloud platform or an edge platform for faster feedback. To enhance efficiency and minimize communication overhead, the federation layer collects and shares the local models only every

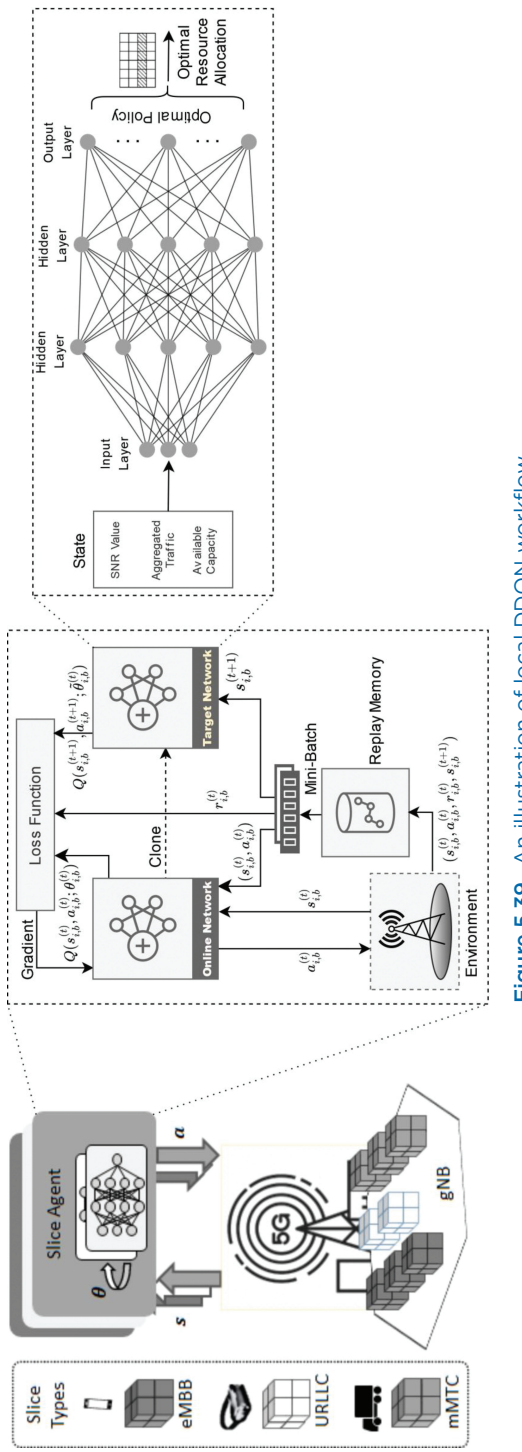


Figure 5.39 An illustration of local DDQN workflow.

T decision interval, referred to as a "federation episode." Different strategies can be applied to derive the global federated model.

Mobile traffic demands follow repetitive patterns linked to human activities and have a spatiotemporal nature. Characterizing these patterns can lead to improved network utilization forecasting and more effective resource allocation planning. However, more than simply relying on the geographical locations and proximity of BSs is needed to accurately understand traffic demands, as land usage may vary even within BS in the same area. It presents a challenge in this framework, as not all federated agents should exchange information, and it should not be limited to just nearby entities. To address this issue, a clustering algorithm is proposed that utilizes network monitoring traces and their similarity to guide the formation of DA subsets.

5.9 Analytics Engine and Interpretable Anomaly Detection

Network slicing will remain a fundamental concept in 6G networks, with an even more increased scale. While slicing increases the flexibility of the system in order to support different types of deployments and services, it also increases the complexity of the management solutions manyfold. Current standardization directions have focused on centralized management of such virtualized systems, such a solution is becoming extremely complex and hard to scale to the number of slices that are envisioned. Rather than a centralized solution, as it is considered by both ETSI NFV [51] and 3GPP management architectures [52], a solution based on the three decentralized components, namely MS, AE, and DE, offers many advantages, as described in the first section. These components are deployed on the different entities taking part in the management process, i.e., MANO, OSS/BSS, NSMF, and the in-slice management plane, distributing the management functions among these entities, in order to ensure a scalable management system.

The AE is designed to analyse the status of network slices using telemetry data collected by the MS, predict slice KPIs, and detect anomalies in KPI trends. The AE operates per slice, supporting the distributed architecture defined above. Its outputs, i.e., predictions of slice KPIs or anomaly results, are then reported to DE as key indicators to learn from and infer actionable decisions to maintain and optimize the slice performance as defined in SLAs. AE exploits the proposed distributed and scalable MANO architecture in Figure 5.40 to push the analysis close to the data collection MS in each domain (i.e., RAN, edge, and cloud), minimizing the need to transfer raw slice performance and configuration data across the different network domains and slices. The E2E slice KPIs to be predicted by the AE

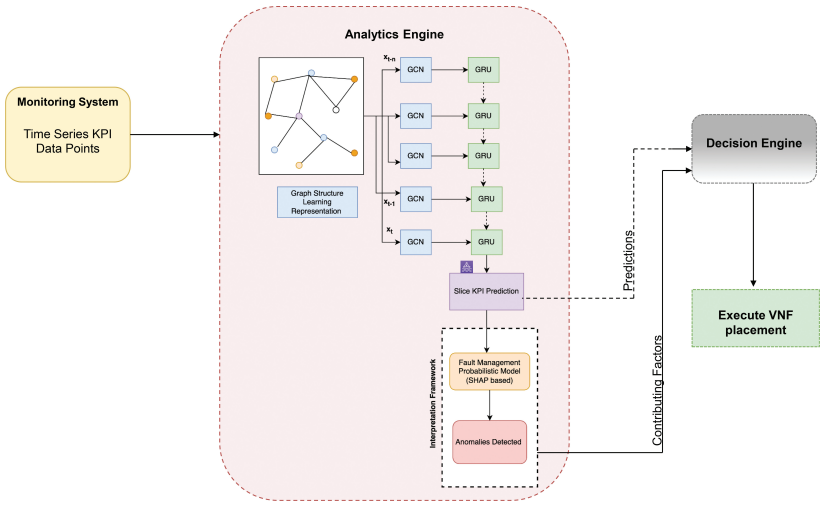


Figure 5.40. Proposed graph-based interpretable anomaly detection (G5IAD) framework.

have been defined in line with standards specifications [53] and reflect the slice performance, including, for example, uplink/downlink throughput for network slice instance (NSI), average E2E uplink/downlink delay, virtualized resource utilization per NSI, etc. Compared to KPI prediction, focus is placed on considering the unique characteristics of 5G and 6G systems when implementing KPI prediction for a slice level AI engine, as opposed to previous systems.

5.9.1 Fault Management Probabilistic Model

Multivariate time-series analysis is an important model that simultaneously analyses multiple measurements to study the behaviour of time-dependent data, and forecasts future values depending on the history of variations in the data. Figure 5.40 illustrates the AE workflow, which incorporates the input data points from MS and is further trained using graph theory to produce the slice KPI predictions and perform fault detection on predictions. In the following, the AE and design of an interpretation framework are introduced, as shown in Figure 5.40.

Graph neural networks model the relationship between a set of objects (nodes or vertices V) and their connections/interrelationships (given by a set of edges E linking the respective nodes/vertices) in non-Euclidean space (data points, which may or may not have the same underlying domain, i.e., same graph structure defined by an adjacency matrix). A combination of graph CNN and recurrent connections is proposed, which uses multiple time series as input to the network. This approach helps in capturing both the spatial and temporal relationships among the network attributes in a slice. The work in this proposed framework adopts

the hybrid graph theory and recurrent neural network-based architecture to learn meaningful network information in an unsupervised manner. To discover hidden associations among network nodes (representing the input features), a graph learning layer computes the graph adjacency matrix, which is later used as an input to all graph convolution modules. The graph learning layer learns a graph adjacency matrix that captures the hidden relationships among the multitude of network slice KPIs. This is then fed into the gated recurrent units to capture the temporal dependencies. The advantages of both of these methods are drawn upon to model both temporal and intra-feature dependencies.

The model receives time series input slice KPIs as input, along with time sequences of slice KPI values for the KPI to be predicted (or analysed for anomalies). The sequences for the input model, as shown in Figure 5.40, are passed through the recurrent neural network GRU layers, while the correlation matrices are processed by graph convolutional networks (GCN). Here, the initial node features (time series data points) are provided as an input to the GCN, and then, the node embeddings are computed by applying the series of convolutional modules. The proposed method utilizes historical time series data as input and employs a graph CNN to capture the topological structure of the network, thereby obtaining relational dependencies. Then, the time series data with relational dependencies are input into gated recurrent units to capture the temporal features. The final results are obtained through a fully connected layer, which predicts the slice KPIs.

Recurrent neural networks and hidden Markov model are used to estimate the probability of a sequence (e.g., sliceLatency KPI) occurring using these probability distributions. Note that $p(x_i|x_{1:\,i-1})$ is the probability of the integer x_i occurring after the sequence $x_{1:\,i-1}$. A language model for sequences specifies a probability distribution for the next in a sequence, given the set of previous sequences. Using a training set of well-known normal sequences, the probabilistic NN is then trained to generate this probability distribution, as shown in Figure 5.41. Given a set of

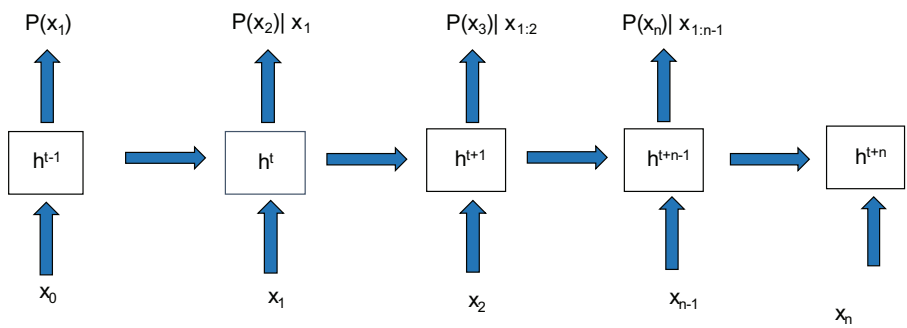


Figure 5.41. Probabilistic classifier for predicting anomalous sequences.

normal sequences, the anomaly detection algorithm evaluates the test instances as anomalous or normal. As the algorithm implemented is cost efficient (reduced training and testing times), it consumes much less computational resources in comparison to the traditional DNN algorithms.

5.9.2 Interpretation Framework

Most DNNs are "black box" that cannot provide easily interpretable insights into the relationship between input and output. Particularly when there is high-dimensional data and multiple layers in the NN, there is a need for a method (post-hoc, after training) that can be used to provide the interpretability of models on the datasets where the ground truth of interpretation results is not available. To overcome this, a relatively new technique in machine learning, known as a supports the interpretation of the NN, or any complex machine learning model, by determining how input features contribute to the value of output features. The SHAP framework unifies methods such as LIME and DeepLIFT [54] under the class of additive feature attribution methods. Reference [55] demonstrates the game theory-based SHAP framework, and the authors in [56] explained how Shapley values could be used for explaining the anomalies detected by a trained NN-based model.

An interpretation framework is provided to enable better insight and automation for the AE, as it is based on deep learning. The DE can use this information to make appropriate decisions on the next steps to take, such as reconfiguring the slice based on the features that have the most impact on the KPI that needs to be brought back into an acceptable range.

The deep explainer SHAP function takes as input the model, an instance of anomalous input x, and a set of background instances. For a particular feature x'i, it calculates a set of SHAP values that measure the importance of each of the features x_1, x_2, …, x_n in predicting x'i. This local interpretability can be represented graphically by using force plots, decision plots which enable us to pinpoint the SHAP value of features with respect to each other. In the anomalies interpretation procedure, the anomalous sequences are first filtered based on the probability score. Then, for each anomalous row, the top features with the lowest probability score are selected. For each feature in the list of top features, the model weights are set, and the Deep SHAP explainer is used to calculate the SHapley values. The deep explainer SHapley function outputs allFeatures, shapValue, and inputValue for anomalous KPI sequences. The shapValue corresponds to the impact of each feature on all other features in the input. The features are sorted based on the shapValue in descending order. Then, the positive shapValue features and their corresponding inputValue are stored as effect (featureEffect) and value (featureValue), respectively.

5.10 Summary and Outlook

The integration of NI into the network architecture will enable the realization of the 6G vision, which aims to provide unprecedented connectivity, capacity, and low latency. To achieve this goal, standardization work has been performed to support analytics coming from network data, and a plethora of artificial intelligence-based algorithms have been proposed. However, there is still a need for a novel architectural framework to integrate NI into major network architectures and enable specific AI-based enablers for the network operation towards 6G. In this chapter, a common architectural framework for the native integration of NI has been proposed towards the fully automated and scalable operation of 6G networks.

This novel NI framework includes an NI stratum that embraces different parts of the network architecture, from access to core, from infrastructure to management, and ensures that AI algorithms are handled with specific criteria related to data acquisition, policy enforcement, and decision making.

Also, several algorithms have been discussed in this chapter, highlighting the capabilities of such an approach based on AI. Once accomplished, AI and 6G networks will be integrated together as a single technology to meet the 6G vision.

References

[1] Hexa-X "D4.2 AI-driven communication & computation co-design: initial solutions". Accessed: April 6, 2023. [Online]. Available: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5e1787205&appId=PPGMS.

[2] Ivan Paez, Luca Cominardi, Miguel Camelo, Paola Soto-Arenas, Nina Slamnik-Krijestorac, Marco Fiore, Marco Gramaglia, Albert Banchs, Maria Molina, Ana Hernandez, Danny de Vleeschauwer, Chia-Yu Chang, Andres Garcia-Saavedra, Andra Lutu, Lidia Fuentes, Joaquín Ballesteros, Aspa Skalidi, Alexandros Kostopoulos, & Georgios Iosifidis. (2021). DAEMON Deliverable 2.1: Initial report on requirements analysis and state-of-the-art frameworks and toolsets. Zenodo. https://doi.org/10.5281/zenodo.5060979.

[3] DAEMON "D3.1: Initial design of real-time control and VNF intelligence mechanisms (Version 1)," 2021. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.5745433.

[4] O. Gheibi, D. Weyns, and F. Quin, "Applying machine learning in self-adaptive systems: A systematic literature review," In *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 15, no. 3, pp. 1–37, 2021. Accessed: April 4, 2023. [Online]. Available: https://doi.org/10.1145/3469440.

[5] M. Camelo, L. Cominardi, M. Gramaglia, M. Fiore, A. Garcia-Saavedra, L. Fuentes, D. De Vleeschauwer, P. Soto-Arenas, N. Slamnik-Krijestorac, J. Ballesteros, C. Y. Chang, G. Baldoni, J. M. Marquez-Barja, P. Hellinckx, and Latré, "Requirements and Specifications for the Orchestration of Network Intelligence in 6G," In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, pp. 1–9, 2022. Accessed: April 4, 2023. [Online]. Available: https://doi.org/10.1109/CCNC49033.2022.9700729.

[6] S. Kukliński, L. Tomaszewski, R. Kolakowski, A. M, Bosneag, A. Chawla, A. Ksentini, S. Ben Saad, X. Zhao, L. A. Garrido, A. Dalgkitsis, B. Bakhshi, and E. Zeydan, "AI-driven predictive and scalable management and orchestration of network slices," In *ITU Journal on Future and Evolving Technologies – Integrated and autonomous network management and control for 6G time-critical application*, vol. 3, no. 3, pp. 570–588, 16 November 2022. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.52953/IPUI5221.

[7] *Management of network slicing in mobile networks; Concepts, use cases and requirements*, Technical Specification (TS) 28.530, v17.1.0, 3GPP, Apr. 2021.

[8] C.Y. Chan, N. Nikaein, O. Arouk, K. Katsalis, A. Ksentini, T. Turletti, and K. Samdanis, "Slice Orchestration for Multi-Service Disaggregated Ultra-Dense RANs.", In *IEEE Communications Magazine*, vol. 56, no. 8, pp. 70–77, 2018. https://doi.org/10.1109/MCOM.2018.1701044.

[9] MonB5G, "D2.4, Final release of the MonB5G architecture (including security)," 2019. Accessed: April 6, 2023. [Online]. To appear in: https://cordis.europa.eu/project/id/871780/results.

[10] G. Choudhury, D. Lynch, G. Thakur, and S. Tse, "Two use cases of machine learning for sdn-enabled ip/optical networks: Traffic matrix prediction and optical path performance prediction," In *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D52–D62, 2018. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1364/JOCN.10.000D52.

[11] R. Vilalta, R. Munoz, R. Casellas, R. Martínez, V. López, O. G. de Dios, A. Pastor, G. P. Katsikas, F. Klaedtke, P. Monti, A. Mozo, T. Zinner, H. Øverby, S. Gonzalez-Diaz, H. Lønsethagen, J. M. Pulidot, and D. King, "Teraflow: Secured autonomic traffic management for a tera of sdn flows," In *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)* pp. 377–382, 2021. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/EuCNC/6GSummit51104.2021.9482469.

[12] S. Orlowski, M. Pioro, A. Tomaszewski, and R. Wessaly, "SNDlib 1.0–Survivable Network Design Library," In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007)*, Spa, Belgium, April

2007. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1002/net.20371.

[13] M. Furdek, C. Natalino, F. Lipp, D. Hock, A. Di Giglio, and M. Schiano, "Machine learning for optical network security monitoring: A practical perspective," In *Journal of Lightwave Technology*, vol. 38, no. 11, pp. 2860–2871, 2020. Accessed: April 6, 2023. [Online]. Availabe: https://doi.org/10.1109/JLT.2020.2987032.

[14] *Management of network slicing in mobile networks; Concepts, use cases and requirements*", Technical Specification (TS) 28.530, v17.1.0, 3GPP, Apr. 2021.

[15] I. Santana-Perez, R. F. da Silva, M. Rynge, E. Deelman, M. S. Pérez-Hernández, and O. Corcho, "Reproducibility of execution environments in computational science using semantics and clouds," In *Future Generation Computer Systems*, vol. 67, pp. 354–367, 2017. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1016/j.future.2015.12.017.

[16] J. Malone, A. Brown, A. Lister, J. Ison, D. Hull, H. Parkinson, and R. Stevens, "The Software Ontology (SWO): a resource for reproducibility in biomedical data analysis, curation and digital preservation", In *Journal of Biomedical Semantics*, vol. 5, no. 25, 2014. Accessed: April 4, 2023. [Online]. Available: doi:10.1186/2041-1480-5-25.

[17] L. Carvalho, D. Garijo, C. B. Medeiros, and Y. Gil, "Semantic Software Metadata for Workflow Exploration and Evolution," In *Proceedings of the Fourteenth IEEE International Conference on eScience*, Amsterdam, The Netherlands, 2018. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/eScience.2018.00132.

[18] R. Guerzoni, I. Vaishnavi, D. Pérez-Caparrós, A. Galis, F. Tusa, P. Monti, A. Sganbelluri, G. Biczók, B. Sonkoly, L. Toka, A. Ramos, J. Melian, O. Dugeon, F. Cugini, B. Martini, P. Iovanna, G. Giuliani, R. Figueiredo, and L. Murillo, "Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey," In *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 4, 2017. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1002/ett.3103.

[19] K. Kostas, N. Nikaein, and A. Edmonds, "Multi-domain orchestration for NFV: Challenges and research directions," In *Proc. of IEEE International Symposium on Cyberspace and Security*, Granada, Spain, 2016. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/IUCC-CSS.2016.034.

[20] A. Collet, A. Banchs, and M. Fiore, "LossLeaP: Learning to Predict for Intent-Based Networking," In *IEEE INFOCOM 2022 – IEEE Conference on Computer Communications*, pp. 2138–2147, 2022. Accessed: April 6,

2023. [Online]. Available: https://doi.org/10.1109/INFOCOM48880.2022.9796918.

[21] J. Kowalik, S. Janusz, *Parallel MIMD computation: the HEP supercomputer and its applications*, MIT Press, Cambridge, 1985.

[22] M. Forsell, J. Roivainen, V. Leppänen, and J. L. Träff, "Implementation of multioperations in thick control flow processors," In *IEEE International Parallel and Distributed Processing Symposium (IPDPS) Workshops*, 2018. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/IPDPSW.2018.00121.

[23] V. Leppänen, M. Forsell, and J-M. Mäkelä, "Thick Control Flows: Introduction and Prospects", In *Proceedings of the 2011 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'11)*, 2011.

[24] J. Keller, C. W. Keßler, and J. L. Träff, *Practical PRAM Programming*, John Wiley & Sons, 2001.

[25] J. Jaja, *Introduction to Parallel Algorithms*. Addison-Wesley, Reading, 1992.

[26] 6G BRAINS "D2.3, Multi-agent Deep Reinforcement Learning Scheme Specification," 30 September 2021. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.5786348.

[27] H. Farhadi and M. Sundberg, "Machine learning empowered context-aware receiver for high-band transmission," In *IEEE Globecom Workshops*, Taipei, Taiwan, Dec. 2020. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/GCWkshps50303.2020.9367518.

[28] H. Farhadi, J. Haraldson, and M. Sundberg, "A deep learning receiver for nonlinear transmitter," In *IEEE ACCESS*, vol. 11, pp. 2796–2803, Oct. 2022. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3234501.

[29] D. Korpi, M. Honkala, J. M. J. Huttunen, and V. Starck, "DeepRx MIMO: Convolutional MIMO detection with learned multiplicative transformations," In *Proc. IEEE International Conference on Communications (ICC)*, Jun. 2021. Available: https://arxiv.org/abs/2010.16283.

[30] M. Honkala, D. Korpi, and J. M. J. Huttunen, "DeepRx: Fully convolutional deep learning receiver," IEEE Transactions on Wireless Communications, vol. 20, no. 6, pp. 3925–3940, Jun. 2021. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/TWC.2021.3054520.

[31] T. L. Jensen and E. De Carvalho, "An Optimal Channel Estimation Scheme for Intelligent Reflecting Surfaces Based on a Minimum Variance Unbiased Estimator," In *ICASSP 2020 – 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5000–5004, 2020. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/ICASSP40776.2020.9053695.

[32] P. Wang, J. Fang, H. Duan, and H. Li, "Compressed Channel Estimation for Intelligent Reflecting Surface-Assisted Millimeter Wave Systems," In *IEEE Signal Processing Letters*, vol. 27, pp. 905–909, 2020. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/LSP.2020.2998357.

[33] X. Wei, D. Shen, and L. Dai, "Channel Estimation for RIS Assisted Wireless Communications – Part II: An Improved Solution Based on Double-Structured Sparsity," In *IEEE Communications Letters*, vol. 25, no. 5, pp. 1403–1407, May 2021. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/LCOMM.2021.3052787.

[34] D. Dampahalage, K. B. Shashika Manosha, N. Rajatheva, and M. Latva-Aho, "Supervised Learning Based Sparse Channel Estimation For RIS Aided Communications," In *ICASSP 2022 – 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8827–8831, 2022. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/ICASSP43922.2022.9746793.

[35] T. V. Chien, T. N. Canh, E. Björnson, and E. G. Larsson, "Power control in cellular massive MIMO with varying user activity: A deep learning solution," In *IEEE Transactions on Wireless Communications*, vol. 19, no. 9, pp. 5732–5748, 2020. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/TWC.2020.2996368.

[36] C. D'Andrea, A. Zappone, S. Buzzi, and M. Debbah, "Uplink power control in cell-free massive MIMO via deep learning," In *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 554–558, 2019. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/CAMSAP45676.2019.9022520.

[37] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, "Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges," In *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1251–1275, Second quarter 2020. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/COMST.2020.2964534.

[38] N. Rajapaksha, K. B. S. Manosha, N. Rajatheva, and M. Latva-aho, "Deep learning-based power control for cell-free massive MIMO networks," In *ICC 2021 – 2021 IEEE International Conference on Communications (ICC)*, 2021. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/ICC42927.2021.9500734.

[39] L. Sanguinetti, A. Zappone, and M. Debbah, "Deep learning power allocation in massive MIMO," In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pp. 1257–1261, 2018. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/ACSSC.2018.8645343.

[40] Y. Zhao, I. G. Niemegeers, and S. H. De Groot, "Power allocation in cell-free massive MIMO: A deep learning method," In *IEEE Access*, vol. 8, pp. 87 185–87 200, 2020. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2992629.

[41] G.A. Reina, A. Gruzdev, P. Foley, O. Perepelkina, M. Sharma, I. Davidyuk, I. Trushkin, M. Radionov, A. Mokrov, D. Agapov, J. Martin, B. Edwards, M.J. Sheller, S. Pati, P. Moorthy, W. Narayana S. Shih-han, P. Shah, and S. Bakas, "OpenFL: An open-source framework for Federated Learning", In *arXiv*, 2021. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2105.06413.

[42] ETSI, "Zero Touch Network and Service Management (ZSM)," 2023. Accessed: April 6, 2023. [Online]. Available: https://www.etsi.org/committee/ZSM.

[43] ETSI, "Experiential Networked Intelligence (ENI)," 2023. Accessed: April 6, 2023. [Online]. Available: https://www.etsi.org/technologies/experiential-networked-intelligence.

[44] H. Chergui, L. Blanco, L. A. Garrido, K. Ramantas, S. Kukliński, A. Ksentini, and C. Verikoukis, "Zero-Touch AI-Driven Distributed Management for Energy-Efficient 6G Massive Network Slicing," In *IEEE Network*, vol. 35, no. 6, pp. 43–49, November/December 2021. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/MNET.111.2100322.

[45] H. Chergui, L. Blanco, and C. Verikoukis, "Statistical Federated Learning for Beyond 5G SLA-Constrained RAN Slicing," In *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 2066–2076, March 2022. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/TWC.2021.3109377.

[46] H.-B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data." In *20th International Conference on Artificial Intelligence and Statistics (AISTATS'2016)*, 2016.

[47] H. Chergui, A. Ksentini, L. Blanco, and C. Verikoukis, "Toward Zero-Touch Management and Orchestration of Massive Deployment of Network Slices in 6G," In *IEEE Wireless Communications*, vol. 29, no. 1, pp. 86-93, February 2022. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/MWC.009.00366.

[48] S. Roy, H. Chergui, L. Sanabria-Russo, and C. Verikoukis, "A Cloud Native SLA-Driven Stochastic Federated Learning Policy for 6G Zero-Touch Network Slicing," In *ICC 2022 – IEEE International Conference on Communications*, May 2022. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/ICC45855.2022.9838376.

[49] F. Rezazadeh, L. Zanzi, F. Devoti, H. Chergui, X. Costa-Pérez, and C. Verikoukis, "On the Specialization of FDRL Agents for Scalable and Distributed 6G RAN Slicing Orchestration," In *IEEE Transactions on Vehicular Technology*, 2022. Accessed: April 6, 2023. [Online]. Available: https://doi.org/10.1109/TVT.2022.3218158.

[50] *Zero-touch Network and Service Management (ZSM)*, Reference Architecture ETSI GS ZSM 002, 2019.

[51] ETSI, "Network Functions Virtualisation (NFV)," 2023. Accessed: April 6, 2023. [Online]. Available: https://www.etsi.org/technologies/nfv.

[52] *Management and orchestration; 5G end to end Key Performance Indicators (KPI)*, Technical Specification (TS) 28.554, v18.0.1, 3GPP, March 2023.

[53] 3GPP, "System architecture for the 5G System (5GC)," 3GPP TS 29.520 v17.1.0, Dec 2020.

[54] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim, "Towards a rigorous evaluation of XAI methods on time series". In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 4197–4201, Oct. 2019. Accessed: April 6, 2023, [Online]. Available: https://doi.org/10.1109/ICCVW.2019.00516.

[55] L. Merrick and A. Taly, "The explanation game: Explaining machine learning models using shapley values," In *Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020*, Dublin, Ireland, August 25–28, 2020, Proceedings 4 (pp. 17–38). Springer International Publishing.

[56] L. Antwarg, R. M. Miller, B. Shapira, L. and Rokach, "Explaining anomalies detected by autoencoders using Shapley Additive Explanations," In *Expert systems with applications*, vol. 186, pp. 115736, 2021.

[57] D. Neumann, T. Wiese, and W. Utschick, "Learning the MMSE Channel Estimator," In *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2905–2917, June 1, 2018, doi: 10.1109/TSP.2018.2799164.

[58] Y. Chen, J. Mohammadi, S. Wesemann, and T. Wild, "Turbo-AI, Part I: Iterative Machine Learning Based Channel Estimation for 2D Massive Arrays," In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, Helsinki, Finland, 2021, pp. 1–6, doi: 10.1109/VTC2021-Spring51267.2021.9449026.

[59] Y. Chen, J. Mohammadi, S. Wesemann, and T. Wild, "Turbo-AI, Part II: Multi-Dimensional Iterative ML-Based Channel Estimation for B5G," In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, Helsinki, Finland, 2021, pp. 1–5, doi: 10.1109/VTC2021-Spring51267.2021.9448950.