

# Towards Trustworthy Reinforcement Learning-based Resource Management in Beyond 5G

Jordi Pérez-Romero<sup>1</sup>, Oriol Sallent<sup>1</sup>, Irene Vilà<sup>1</sup>, Elli Kartsakli<sup>2</sup>, Ömer Faruk Tuna<sup>3</sup>,  
Swarup Kumar Mohalik<sup>4</sup>, Xin Tao<sup>4</sup>

<sup>1</sup>Universitat Politècnica de Catalunya (UPC), Barcelona, Spain; <sup>2</sup>Barcelona Supercomputing Center (BSC), Barcelona, Spain; <sup>3</sup>Ericsson Research, Istanbul, Turkey, <sup>4</sup>Ericsson Research, Stockholm, Sweden.

**Abstract**— It is envisaged that future Beyond 5G (B5G) systems will make extensive use of Artificial Intelligence (AI) capabilities to achieve an efficient automated management and optimization of communication and computing resources and to support advanced data-driven applications that provide the users with highly immersive services. Ensuring the trustworthiness of the AI solutions is key for their successful introduction in B5G, as this will guarantee their robustness towards errors and potential attack threats, the privacy of the used data and trained models and the explainability of AI-based decisions, ensuring that they do not have unsafe consequences. In this context, this paper focuses on the trustworthiness of Reinforcement Learning (RL) solutions, as their inherent trial-and-error behavior during training makes them particularly challenging from the robustness perspective. Then, the paper proposes a framework for managing the lifecycle of an RL-based resource management solution for both training and inference stages to ensure its trustworthy operation. The framework relies on an RL training configuration function to specify the training conditions, a Network Digital Twin (NDT) to perform the training on a safe environment and a continuous operation function to monitor the behaviour of the trained policy during inference. The framework is illustrated with an applicability use case of capacity sharing for network slicing.

**Keywords**—AI/ML-based optimization; Trustworthy AI; B5G/6G evolution; Reinforcement Learning

## I. INTRODUCTION

Artificial Intelligence (AI) development aims to benefit society as a whole and assist humans by resolving challenging problems. However, AI might unintentionally harm humans, for example by making decisions that lead to adverse effects on a system that supports critical services (e.g. remote surgery, autonomous driving, etc.). As a result, the research community has recently given significant attention to trustworthy AI to reduce any unfavorable impacts that AI may have on people [1]. Trustworthiness becomes a prerequisite to develop, deploy and use AI systems, because without AI systems being demonstrably worthy of trust, unwanted consequences may arise, and their uptake might be hindered. The High-Level Expert Group on AI (AI HLEG) set up by the European Commission elaborated in [2] the ethics guidelines for trustworthy AI and established the existence of three components to be met throughout the system's entire life cycle. Specifically, AI should be *lawful*, complying with all applicable laws and regulations, *ethical*, ensuring adherence to ethical principles and values, and *robust* to prevent unintentional harm. The AI HLEG group also provided guidance on how to realize trustworthy AI by listing seven key requirements [2][3], including among them technical robustness and safety.

A clear application area of AI is in the context of future Beyond 5G (B5G) and 6G cellular systems, as it is commonly

agreed that, thanks to the enormous development in computational resources, edge- and cloud-computing, as well as the ever-increasing amount of available network and application data, AI will permeate almost all the layers of these systems. The vision is that B5G/6G will leverage AI for optimising the air interface and to transform the network to a powerful distributed AI platform. Hence, the AI as a Service (AIaaS) concept, which refers to the provision of AI and Machine Learning (ML) capabilities as a cloud-based service that can be consumed by the decision-making components of the network (e.g., orchestrators, controllers, infrastructure managers, etc.), will be a key B5G/6G enabler [4].

To facilitate the deployment of such AI-enabled services, architectures should be able to manage the complete lifecycle of the AI/ML models in a transparent and automated way and should encompass the mechanisms that ensure the trustworthiness of the solutions. In this direction, the VERGE project (<https://www.verge-project.eu/>) introduced in [5] an architecture for the evolution of edge computing towards B5G/6G. The proposed architecture aims to enable the seamless execution of cloud-native services, including disaggregated Radio Access Network (RAN) and core network functions, distributed AI, and big data workflows, while leveraging data-driven AI/ML-based solutions for edge and network optimization. One of the main pillars of this architecture is the so-called “Security, Privacy and Trustworthiness for AI” (SPT4AI), which encompasses a set of methods to ensure the trustworthiness in the VERGE AI solutions.

This paper focuses on the development of trustworthy reinforcement learning (RL) solutions and how they can be supported and facilitated in the VERGE architecture. RL is a subset of ML that consists in learning a behavioral model through the dynamic interaction with an environment [6]. It provides a mathematical formalism for learning-based control that allows acquiring near-optimal behavioral skills and, for this reason, RL methods have applicability in many decision-making problems and have been proposed for many functionalities in wireless networks [7], particularly in those related with resource management, as RL allows dealing with the uncertainties of the radio environment.

In RL, an agent executes actions depending on the observed state in the environment and then obtains a reward signal as a result that quantifies how good or bad was the outcome of the selected action. This process is iteratively repeated during the training stage so that the decision-making policy at the RL agent can be progressively enhanced. To learn a robust RL decision-making policy, the selection of the actions during the training needs to balance the exploitation of the knowledge that has been acquired from previous decisions and the exploration of new

actions that are randomly selected in order to discover if they can improve the current policy. This exploration, which is essentially a trial-and-error approach, can be particularly critical depending on the considered scenario and decision-making problem (e.g. in case of safety-critical services such as healthcare or autonomous driving), because it can lead to considerable degradations in the performance of the related services, thus affecting the trustworthiness of the RL algorithm.

In this context, the main contribution of this paper is the proposal of a framework for a trustworthy lifecycle management of RL-based resource management strategies. Among the dimensions of trustworthiness, the focus is on the technical robustness, targeting that the RL decision-making policies are reliable and make accurate decisions that do not lead to adverse effects on the system. To that end, the proposed framework covers both the training and inference stages and is aligned with the architecture of the VERGE project. Workflows to illustrate the interworking between the involved components are also included.

The paper is organized as follows. Section II presents a brief summary of the VERGE architecture. Section III describes the proposed framework for trustworthy RL-based resource management and Section IV presents some illustrative results on an applicability example of RL for capacity sharing. Finally, conclusions are given in Section V.

## II. MAIN HIGHLIGHTS OF THE VERGE ARCHITECTURE

The architecture for AI-powered edge computing evolution proposed by the VERGE project is built around three main pillars. The first one is the “Edge for AI” (Edge4AI), a flexible, modular and converged edge platform design that unifies the lifecycle management and closed-loop automation for cloud-native applications and network services across a unified edge-cloud compute continuum. The second pillar is the “AI for Edge” (AI4Edge), a portfolio of AI-based solutions to manage and orchestrate the computing and network resources. The third pillar is the SPT4AI, a suite of methods and tools to ensure the privacy of sensitive data and AI models, the security of the AI-based models against adversarial attacks, their safe training and execution, and their explainability for different stakeholders.

The key building blocks of the VERGE architecture are illustrated in Fig. 1. A highly heterogeneous infrastructure is depicted at the bottom, consisting of diverse edge computing resources (from the Far Edge to the Near Edge and the Cloud) embedded in the end-to-end (E2E) B5G network. The architecture intends to provide services to users of multiple types associated with different use cases (e.g. augmented/extended reality, smart cities, automotive, etc.). Users are connected through heterogeneous RAN deployments and leverage the availability of Multi-access Edge Computing (MEC) services. The provision of services across this infrastructure is sustained on the three VERGE pillars shown in the upper part of Fig. 1 and summarized in the following paragraphs (see [5] for details).

The *Edge4AI* forms an AI-powered platform to facilitate the deployment and execution of cloud-native services and network functions from the *Application layer* over the heterogeneous pool of connected edge and cloud resources. The *Edge4AI virtualization layer* provides a unified view of the communication and computational resources, forming an edge-cloud compute continuum tightly integrated with the B5G

communication fabric. To flexibly deploy cloud-native functions on the infrastructure, the *Edge4AI* includes the *Orchestration, Management and Control layer*. It handles the orchestration of services and infrastructure and the control of the RAN elements. For the latter, a set of intelligent RAN controllers at single site and multi-site level is included. The lifecycle management of the AI/ML solutions is enabled by a set of Application Programming Interfaces (APIs), supporting services and toolkits under the scope of the so-called *Cognitive Framework*. Moreover, the *Distributed Knowledge Base (DKB)* contains a registry of the available AI/ML models, used datasets and associated metadata. Finally, the *Data Access layer* is in charge of gathering relevant data from the observability and telemetry stacks to monitor the underlying infrastructure and services. To that end, it includes a set of distributed agents for the ingestion of data from RAN and core, edge platform telemetry and application-related metrics. The datasets employed for the training of AI/ML models can also be stored in an *Open Dataspace*, enabling their reutilization and transparent usage.

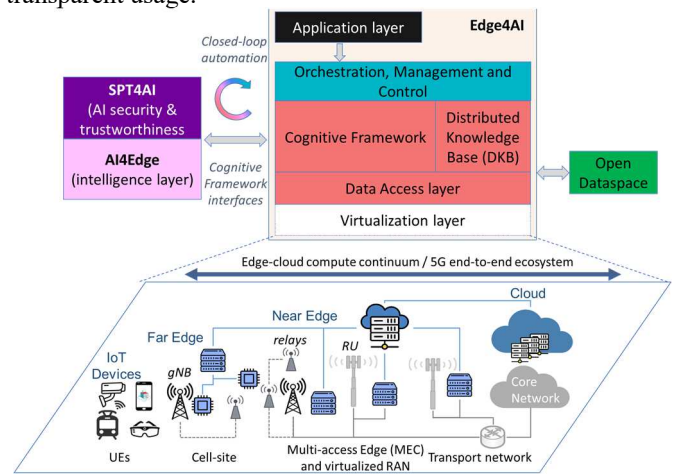


Fig. 1. Key building blocks of the VERGE architecture.

The *AI4Edge* pillar forms the intelligence layer with the AI/ML models designed for the automated management and optimization of communication and computing resources and for the support of advanced data-driven applications that provide immersive services to the users. The *AI4Edge*, which is facilitated by the cognitive framework functionalities and APIs, specifies the model-specific methods for AI/ML training and validation, AI/ML model monitoring and management and AI/ML model inference of the trained models.

Finally, the *SPT4AI* pillar provides the methodologies and tools for ensuring secure, private, safe and explainable operations of the *AI4Edge* models, thereby increasing their trustworthiness. The *SPT4AI* includes, among other functionalities, the ones for trustworthy RL that are proposed in this paper and discussed in next section.

## III. FUNCTIONAL FRAMEWORK FOR TRUSTWORTHY RL-BASED RESOURCE MANAGEMENT

### A. Architectural framework

The architectural components of the proposed solution for trustworthy RL-based resource management are illustrated in

Fig. 2 in the context of the VERGE architecture. An RL agent applies a decision-making policy to support a certain functionality as part of the AI4Edge portfolio of solutions (e.g. handover, resource allocation, power control, etc.). The decision-making policy is learnt by the RL agent during the training stage and, after this stage is completed, the resulting policy is applied in the so-called inference stage. Inference is executed at a network function referred to in Fig. 2 in a generic manner as the *controller*. It can be part of the orchestration, management and control layer or can be embedded in certain network nodes (e.g. base stations).

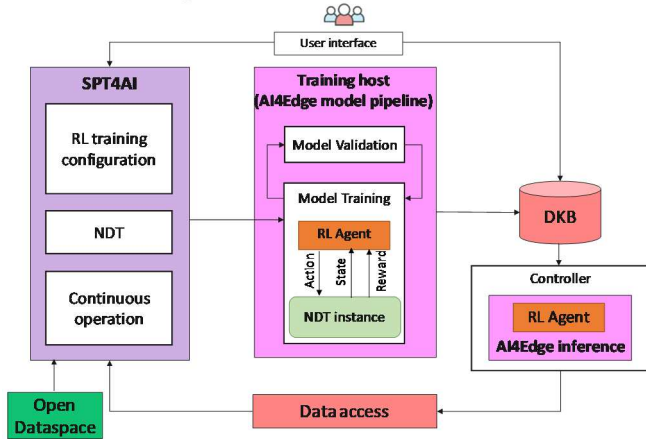


Fig. 2. Architectural components of the trustworthy RL solution.

To achieve a trustworthy operation of the RL agent, the proposed framework incorporates specific SPT4AI functionalities that configure and monitor the model training and inference stages associated with the AI4Edge pillar. Specifically, the proposed framework is sustained on the *RL training configuration* function, the *continuous operation* function, and on the use of a Network Digital Twin (NDT) of the RAN that provides a safe training environment.

The degree of trustworthiness of the RL agent should be assessed through a metric (or metrics) that quantifies to what extent the decision making policy is technically robust and makes accurate decisions during the inference. In general terms, this metric is referred to in this paper as *optimality level*. Its precise definition depends on the problem at hand but typically it should be computed by contrasting network performance measurements (e.g. throughput, energy consumption, etc.) against desired targets. The *optimality level* should be considered in the design and operation of the SPT4AI functions.

### 1) *RL training configuration function*

The key role of the *RL training configuration* function is to establish the proper strategy to efficiently conduct the training, trading-off aspects such as the training duration, the sustainability of the training process (connected e.g., to energy consumption), and the *optimality level* of the learnt policy. To that end, the *RL training configuration* should specify the hyperparameters of the RL technique (e.g., Deep Neural Network configuration, learning rate, etc.) and the parameters of the training environment so that the RL agent faces a diversity of situations during the training stage. These situations constitute the basis for learning a robust policy so that the agent knows how to react in each possible situation during the

inference stage on the real network. For example, for training AI models addressing radio resource allocation in the RAN, the training should be conducted under different space/time variations of the offered load of different services and cells.

Multiple datasets with different characteristics can be available to specify the situations that the RL agent will experience during training. Datasets can be extracted from different sources (e.g., RAN measurements, synthetic data, data augmentation) and can be available in the Open Dataspace of the VERGE architecture.

In order to tackle the *RL training configuration* function in a systematic way, different features can be associated to each of the available datasets. An example is the *dataset coverage* metric [8], which measures the range of situations that are captured by a dataset with respect to the total set of possible situations. A high coverage metric means that the RL agent will face a large variety of situations during the training.

Another feature that can be used to assess the quality of the dataset is the so-called *degree of coincidence*, which measures the ratio between the situations that are expected to be observed during inference and those that are included in the dataset. A low value of this metric indicates that the inference conditions are not properly captured in a dataset.

### 2) *Network Digital Twin*

In order to deal with the intrinsic randomness of RL algorithms during the training stage due to the exploration vs. exploitation trade-off, the use of an NDT is considered. An NDT provides a virtual and updated representation of the network that allows analysing, diagnosing and emulating the physical network in a zero-risk environment [9][10]. Therefore, it brings the opportunity to train RL algorithms by testing the outcomes of the different actions selected by the RL agents on a virtualized, updated, and safe version of the real network. Indeed, the use of an NDT for training RL algorithms can be the key to their practical adoption, as it assures that the outcomes of random exploration actions, which are needed to progressively enhance the decision-making policy, are obtained without having any impact on the real network.

Aligned with the terminology and concepts proposed by IETF in [10], an NDT can be composed of three main modules [11]. First, the *service mapping models* module includes the models that characterize the network nodes and functionalities (e.g. base stations, routers, propagation, mobility, etc.). The *data repository* module collects and stores data from the network used to get an accurate representation of the reality. This collection can be done through the data access layer of the VERGE architecture. The datasets selected for the training by the *RL training configuration* will also be part of the NDT *data repository*. Finally, the *digital twin management* module controls the lifecycle of the NDT enabling the configuration of the service mapping models, the deployment of NDT instances with the selected models, the control of the execution and the extraction of Key Performance Indicators (KPIs).

As seen in Fig. 2, the training of the RL model is conducted at the training host that includes the RL agent. The environment used for the training is an instance of the NDT defined with the adequate service mapping models set in accordance with the scenario where the RL agent should operate. The NDT instance also uses the training dataset selected by the *RL training configuration* function to produce the situations specified in this

dataset e.g., in terms of traffic generation. During the training, the RL agent iteratively makes observations of the state of the environment and chooses an action that results in the modification of certain parameters at the NDT. Then, the NDT simulates the behavior of the network with the new configuration and provides as a result the reward value to be used by the RL agent to progressively improve the policy. This is iteratively repeated until reaching a termination condition specified by the *RL training configuration* function (e.g. having a variation of a loss function lower than a threshold). After the training execution terminates, the resulting model is validated to check its behavior under specific scenario conditions emulated at the NDT. Then, if the validation is successful, the training is considered completed and the resulting RL model (i.e. the learnt decision making policy) can be stored in the DKB and later on deployed in the controller for the inference stage.

### 3) Continuous operation function

When the decision-making policy is used at the inference stage to make decisions over the real network the *continuous operation* function at the SPT4AI monitors the behaviour of the policy to detect degradations, analyse the root cause of such degradations and, if needed, decide to perform a retraining or a change of policy as a fall-back mechanism. The performance of the policy during inference will be highly related with the similarity between the conditions (e.g. load levels) included in the training dataset and those experienced during inference. Moreover, it will also depend on the generalization capability of the policy to adapt to potentially new situations not captured in the training dataset. The operation of the *continuous operation* function can be based on monitoring different aspects. On the one hand, it should assess the *optimality level* of the policy based on the performance that results from the selected actions. On the other hand, it should measure the similarity between the conditions experienced during the training and those experienced during inference, using e.g. the *degree of coincidence* explained in Section III.A.1. In this way, if performance degradations are observed in the network, the *continuous operation* will be able to understand if these are due to substantial differences between the training and the inference conditions. In such a case, a retraining can be needed. As seen in Fig. 2 the *continuous operation* function uses network metrics and KPIs collected through the data access layer.

### B. Workflows

To illustrate the interworking between the different components of the solution, Fig. 3 shows the workflow of the procedure for the initial deployment of an RL model. First, the user of the VERGE platform, e.g., the Mobile Network Operator (MNO), accesses the RL training configuration function (step 1) to specify the hyperparameters and the training dataset (step 2) that determines the situations to be experienced by the RL agent. The corresponding training configuration will be enforced at the training host through one of the APIs of the cognitive framework in the VERGE architecture (step 3). At the same time, the user will also configure the NDT (steps 4, 5) by selecting the models to be used during the training in terms of e.g., mobility, propagation, network topology. As a result, the NDT instance that will be used in the training will be deployed at the training host (step 6). It is worth mentioning that, although the workflow of Fig. 3 assumes that the steps 1 to 6

are controlled by the user, an automated operation would also be possible. In that case, the RL training configuration function should include automated reasoning mechanisms to select the training datasets and configure the NDT while the role of the user would be limited to specifying high level policies (e.g. optimality level target, maximum training duration, etc.).

After configuring and deploying the NDT, the RL training process starts (step 7) and lasts until reaching a termination condition. Afterwards, the resulting model is validated (step 8) by testing it under a specific configuration of the NDT instance used for evaluation purposes. If the validation is satisfactory, the training will be considered completed and the resulting trained model will be stored at the DKB (step 9), together with the used training datasets and the training data (i.e., the hyperparameters, visited states, actions and rewards during the training). At this point, the user will be able to request the deployment of the RL model stored at the DKB on the controller through the corresponding API of the cognitive framework (steps 10, 11). Finally, the deployed RL model will start to operate in the inference stage (step 12).

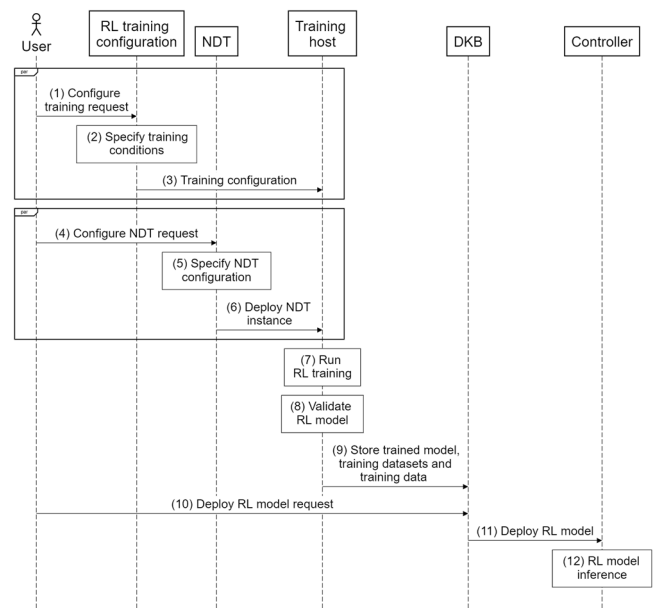


Fig. 3. Workflow of the RL model initial deployment.

The continuous operation function follows the workflow shown in Fig. 4, which illustrates the case that a retraining is required and this retraining can be handled automatically without user intervention. During inference stage (step 1), the controller provides monitoring data to the continuous operation function (step 2). This data includes performance KPIs together with the states, actions and measured rewards. At the same time, the continuous operation function also gets from the DKB the stored training data of the model (e.g., visited states during training, actions, rewards, etc.) (step 3). In this way, it will be able to monitor the performance of the RL model during inference and compare it against the one that was obtained during the training (step 4). This will allow detecting discrepancies between the behaviour of the model during the training and the inference stages, e.g., due to changes in the network, due to the NDT instance not being properly configured, due to different traffic conditions, etc. In case that significant discrepancies are detected, the continuous operation

function will issue a retraining request to the RL training configuration function (step 5). Consequently, the training configuration function will request monitoring data from the controller to better characterise the real network operation (steps 6, 7). This new information will be used to better specify the training conditions (steps 8, 9) and to better configure the NDT (steps 10, 11), so that a new NDT instance for training is generated (step 12). Then, the model will be retrained and validated at the training host (steps 13, 14) and a new version of the model will be stored at the DKB (step 15) and deployed at the controller (steps 16, 17) for continuing with the inference (step 18). Although the workflow presented here assumes that the retraining is automatically handled, other possibilities could exist in which the user of the platform could specify the training conditions and/or the NDT based on the monitoring data.

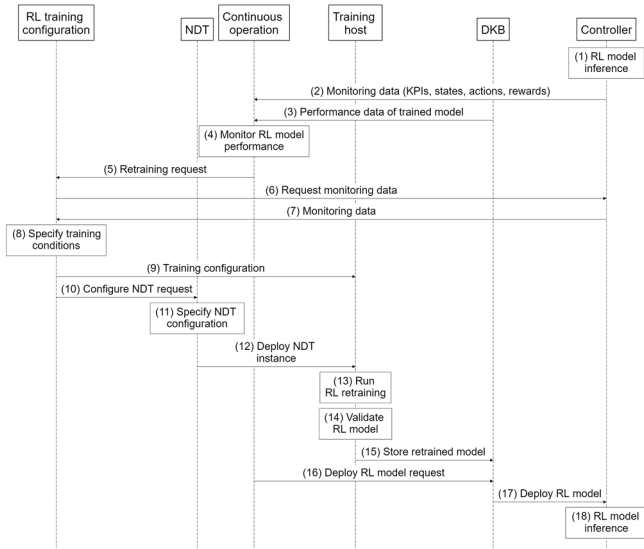


Fig. 4. Workflow of the continuous operation in case of retraining.

#### IV. APPLICABILITY EXAMPLE

This section presents an applicability example of the proposed framework to gain insight in the role of the defined metrics and how they can drive the definition of strategies for the RL training configuration and the continuous operation functions. The considered RL strategy is the Deep Q Network (DQN) Multi Agent Reinforcement Learning (MARL) algorithm from [12] that dynamically allocates the capacity of a 5G New Radio (NR) RAN to a set of RAN slices. It intends to fulfil the Service Level Agreement (SLA) with each slice, defined in terms of the aggregate bit rate to be provided across the involved cells, and to maximize the utilization of the available Physical Resource Blocks (PRBs) in each cell. Thus, DQN-MARL deals with the offered load fluctuations of RAN slices to make assignments that fulfil at least the SLA of the slices when this is demanded and can absorb excess loads above the SLA if there is spare capacity.

The operation of DQN-MARL is based on an RL agent per slice that, based on a state that captures the PRB occupation, the cell capacities and the SLA requirements, decides an action that consists in increasing, maintaining or decreasing the fraction of assigned PRBs to the slice in each cell. In turn, the reward captures both the SLA satisfaction and the capacity overprovisioning. The decision-making policy to select the

actions is a Deep Neural Network (DNN) whose weights are adjusted during the training (see [12] for details).

The considered scenario assumes a 5G NR cell with capacity  $\sim 120$  Mb/s and two RAN slices. Without loss of generality, the SLA of slice 1 is given by an aggregate bit rate corresponding to 60% of the capacity, while the SLA of slice 2 is the 40%. The DQN-MARL strategy makes decisions in periods of 3 min and modifies the assigned fraction of PRBs per slice in steps of 0.03. The DNN has an input layer with 7 neurons, a single layer with 100 neurons, and an output layer with 3 neurons. The rest of DQN parameters are those of [12].

The *optimality level* in this example is defined as the ratio between the reward obtained with the learnt policy and the optimum reward, which is obtained from an optimum ideal policy derived after analyzing all the possible PRB allocations to the two slices. The *optimality level* is computed for each pair of offered loads of the two slices observed during the inference stage as well as on average terms.

##### A. RL training configuration: impact of dataset selection

This section intends to assess the importance of the dataset selection in the *RL training configuration* function in terms of the resulting *optimality level*. To that end, 16 different policies of the DQN-MARL algorithm have been obtained, each one resulting from an execution of the training with a different training dataset. Each dataset is defined by a dynamic variation of the offered load in the two slices for a duration of  $2E6$  decision making periods and has a different *dataset coverage* metric given by the percentage of combinations of offered loads of the two slices included in the dataset with respect to the total number of possibilities.

For each policy the inference stage has been executed by applying the policy with a given pattern of the offered load variation per slice covering a total of 38400 decision making periods (i.e. 80 days). Fig. 5 illustrates the *optimality level* obtained with two policies, one learned with a training dataset of coverage 92% (Fig. 5a) and another with a much smaller coverage of 16% (Fig. 5b). It is observed that the policy trained with the dataset of large coverage exhibits a high *optimality level* close to 1 for most of the observed offered loads during inference. In contrast, the policy trained with the dataset of less coverage only has high *optimality level* for medium/average load levels, while for reduced loads poorer values between 0.3 and 0.6 are observed, reflecting that the policy is not making optimum decisions.

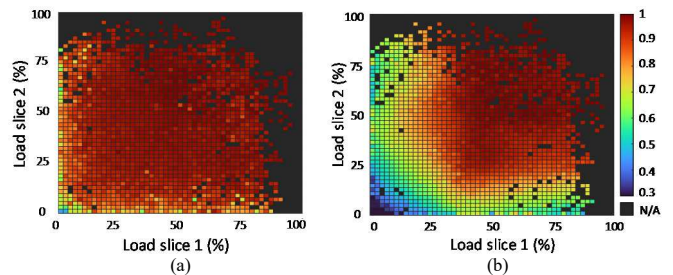


Fig. 5. Optimality level for training datasets of coverage (a) 92%, (b) 16%.

Fig. 6 quantifies the relationship between the *dataset coverage* and the *optimality level*. To that end, the 16 policies have been grouped in four ranges according to their *dataset coverage*. Then, Fig. 6 plots, on the one hand, the average

*optimality level* for the policies in each range and, on the other hand, the percentage of offered loads with *optimality level* higher than 90%, computed also as an average for all the policies in a range. The results reflect that, in this study, datasets with coverage lower than 25% only reach average *optimality levels* around 85% and only in 50% of the loads the *optimality level* is higher than 90%. In contrast, this significantly improves when the coverage is higher than 25%. In this case, the average *optimality level* is around 93-94% and in approximately 80% of the loads it is higher than 90%, thus reflecting a much more trustworthy RL policy. So this means that a proper design of the RL training configuration function should carefully consider the *dataset coverage* as it will have a significant influence on the model *optimality* after training.

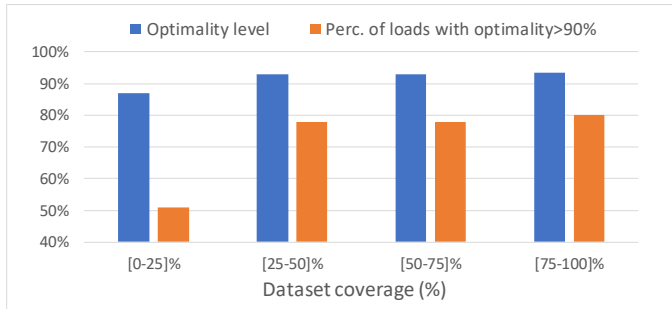


Fig. 6. Relationship between optimality level and training dataset coverage.

### B. Continuous operation: impact of the degree of coincidence

Results in this section illustrate how the *degree of coincidence* metric can be used to support the *continuous operation* function. Here this metric is computed as the fraction of offered load combinations of the two slices observed during the inference that were also included in the training dataset. Fig. 7 depicts the *optimality level* as a function of the *degree of coincidence* obtained when the inference is executed with three policies trained with datasets of different coverage. Each point in the figure corresponds to the average of one day of the inference stage. It is observed that the days with the poorer optimality (e.g. below 70%) are those in which the degree of coincidence is small, i.e. lower than ~10%, and they occur mostly with the policy trained with the dataset 3 of low coverage of 16%. This reflects that this policy is not performing adequately during these days. Thus, the continuous monitoring function would detect on the one hand a degradation of the *optimality level* and on the other hand that this degradation is associated with a low *degree of coincidence*. This would be an indication that a retraining with a better dataset would be appropriate.

## V. CONCLUSIONS

The availability of trustworthy AI solutions is envisaged as a key aspect for the successful introduction of AI in future cellular systems. In this context, this paper has focused on the mechanisms to achieve a trustworthy operation of RL-based solutions, as they can be used for different decision-making problems in B5G. Specifically, the paper has proposed an architectural framework based on an RL training configuration function to specify the training conditions, on the use of a Network Digital Twin for safely conducting the training and on a continuous operation function to monitor the trained policies

during inference. The interworking between these components has been elaborated by means of different workflows that reflect the initial model deployment and the continuous operation. The paper has also presented different metrics to support the operation of these functions, such as the dataset coverage, the degree of coincidence or the optimality level. These have been illustrated using a DQN-MARL algorithm for RAN slicing.

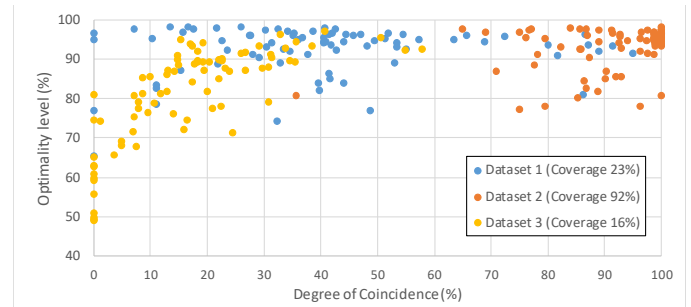


Fig. 7. Optimality level and degree of coincidence for different trained policies.

## ACKNOWLEDGEMENT

This work is part of VERGE project, which has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101096034. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. The work of Irene Vilà has been funded by a Margarita Salas Grant (ref. 2022UPC-MS- 94079).

## REFERENCES

- [1] H. Liu, et al. "Trustworthy AI: A Computational Perspective", CoRRabs/2107.06641 2021.
- [2] High-Level Expert Group on Artificial Intelligence, "Ethics Guidelines for Trustworthy AI", European Commission, April, 2019.
- [3] L. Floridi, "Establishing the rules for building trustworthy AI", Nature Machine Intelligence, May, 2019.
- [4] 5G PPP Architecture WG, "The 6G Architecture Landscape. European perspective. Version 1.0", White paper, December, 2022.
- [5] E. Kartsakli, et al. "An Evolutionary Edge Computing Architecture for Beyond 5G Era", IEEE Int. Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Nov. 2023.
- [6] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd edition, The MIT Press, 2018.
- [7] J. Wang, et al, "Thirty Years of Machine Learning: The Road to Pareto-Optimal Wireless Networks", IEEE Comms. Surveys & Tutorials, Vol. 22, No. 3, 3rd Quarter, 2020.
- [8] I. Vilà, et al. "Impact Analysis of Training in Deep Reinforcement Learning-based Radio Access Network Slicing", IEEE CCNC, 2022.
- [9] H. X. Nguyen, et al., "Digital Twin for 5G and Beyond", *IEEE Communications Magazine*, Volume 59, no. 2, pp. 10-15, 2021.
- [10] C. Zhou et al. "Digital Twin Network: Concepts and Reference Architecture", *Internet Engineering Task Force*, Internet Draft draft-irtf-nmrg-network-digital-twin-arch-03, April, 2023.
- [11] I. Vilà, O. Sallent, J. Pérez-Romero, "On the Design of a Network Digital Twin for the Radio Access Network in 5G and Beyond", *Sensors*, MDPI, Vol. 23, 1197, Jan. 2023.
- [12] I. Vilà, J. Pérez-Romero, O. Sallent, A. Umbert, "A Multi-Agent Reinforcement Learning Approach for Capacity Sharing in Multi-tenant Scenarios", *IEEE Trans. on Veh. Tech.*, July 2021.