5Growth Data-Driven AI-Based Scaling

5Growth Data-driven AI-based Scaling

Danny De Vleeschauwer*, Jorge Baranda†, Josep Mangues-Bafalluy†, Carla Fabiana Chiasserini‡, Marco Malinverno‡, Corrado Puligheddu‡, Lina Magoula§, Jorge Martín-Pérez¶, Sokratis Barmpounakis§, Koteswararao Kondepu‖, Luca Valcarenghi‖, Xi Li**, Chrysa Papagianni††*, Andres Garcia-Saavedra**
*Nokia Bell Labs, †Centre Tecnològic de Telecomunicacions de Catalunya, ‡Politecnico di Torino,
§National and Kapodistrian University of Athens, ¶Universidad Carlos III de Madrid, ‖Scuola Superiore Sant'Anna,
**NEC Laboratories Europe, ††University of Amsterdam

***Abstract*—This paper presents a data-driven approach leveraging AI/ML models to automate the service scaling operation and, in this way, meet the service requirements while minimizing the consumption of network, computing, and storage resources. This approach is integrated into the 5Growth service management software platform. In particular, a prototype was developed to demonstrate how the novel 5Growth AI/ML platform can be used in a closed-loop automation system to support the automated service scaling operation. Furthermore, a number of additional ML-based approaches are developed in the context of eMBB and V2N scenarios, which can be embedded into the system for handling more complex use cases.**

I. Introduction

5G is set out to enable a broad set of diversified and heterogeneous services with potentially conflicting demands. Towards that end a flexible, adaptable, and programmable architecture based on network slicing, is proposed and gradually adopted, capable of delivering tailored services to distinct verticals and use cases. Network softwarization, using Software-Defined Networking (SDN) and Network Function Virtualization (NFV) are key technologies enabling programmable control and flexible management in 5G networks.

In this emerging framework, resources elasticity [1] and network slice elasticity [2] are deemed essential for making efficient use of the computational and networking resources while guarantying service performance and meeting Service-Level-Agreements (SLAs) [3]. In the 5Growth context, a service can be mapped onto one or multiple network slices, and hence, scaling of a service implies scaling of its composed service instances and resource allocation of the associated network slice(s). However, the automated assurance of service quality under the dynamics of the available infrastructure resources and varying service demands poses additional challenges in closed-loop service orchestration, dynamic resource allocation and automated service scaling. To ensure that slices support the explicit service requirements expressed by specific SLAs, communication service providers need to employ service and network intelligence and slice context awareness.

Towards that end, Artificial Intelligence (AI) and Machine Learning (ML) are evolving into built-in architectural features, enabling autonomous networks driven by AI/ML. Standardization groups, such as the ITU Focus Group on Machine Learning for Future Networks including 5G [4], the ETSI Zero touch network & Service Management (ZSM) [5] and the Experiential Networked Intelligence (ENI) [6], are currently working in this direction, although no complete working solution is yet available. In particular, ETSI ZSM envisions a new end-to-end architecture framework designed for closed-loop automation and optimized for data-driven AI/ML algorithms, while ETSI ENI focuses on the use of AI for the management of network slices as well as of security issues. AI is also one of the pillars of the efforts pursued by the O-RAN alliance[1] for the radio access network in 5G and beyond. O-RAN alliance strives to leverage emerging learning techniques to embed intelligence in every layer of the Radio Access Network (RAN) architecture [7].

In the same direction, the EU 5Growth (5Gr) project [8] extends the service platform developed in the EU 5G-Transformer (5GT) project [9] towards an AI-driven automated service and network management platform for 5G, with the addition of a novel AI/ML platform (5Gr-AIMLP). Its integration in the 5Gr architecture extends the 5Gr platform with a closed-loop automation and zero-touch service and network management system. A description of its first prototype is available in [10].

In the following, we presents the architecture of the 5Gr platform (Sec. II). The main contribution of this paper is the design of the novel AI/ML platform and its integration in the 5Gr software platform for a closed-loop automation system, demonstrated with the scaling of a network service of an Industry 4.0 use case (Sec. III). Furthermore, we propose a number of additional (ML-based) approaches for auto-scaling elastic resources and service instances of network slice(s), in the context of eMBB (Sec. IV) and V2N scenarios (Sec. V).

II. 5Growth Architecture

The 5Gr project aims at developing a modular platform that is fully automated and yet highly flexible. The 5Gr architecture is composed of three core building blocks, namely the Vertical Slicer (5Gr-VS), Service Orchestrator (5Gr-SO) and Resource Layer (5Gr-RL), complemented by the Vertical-oriented Monitoring System (5Gr-VoMS) and the AI/ML Platform (5Gr-AIMLP). These building blocks interact with each other, creating a closed-loop control of the system.

The **5Gr-VS** acts as a single point of entry for verticals requesting 5G network services, through a simplified and vertical-oriented northbound interface. Through this interface,

[1] https://www.o-ran.org/

service requests can be submitted, while the 5Gr-VS maps the requested vertical services to network slices, which are deployed as NFV network services (NFV-NSs) forwarding their request to the 5Gr-SO. The **5Gr-SO** provides network service and resource orchestration capabilities to support end-to-end orchestration of NFV-NSs and their lifecycle management (including scaling). Moreover, the 5Gr-SO offers the 5Gr-VS an integrated view of the end-to-end services. The **5Gr-RL** is a customizable SDN/NFV-based transport and computing platform capable of supporting a diverse range of computing and networking requirements. A single 5Gr-RL can integrate several Virtual Infrastructure Managers (VIM) and Wide-Area Infrastructure Manager (WIM) from different technological domains, exposing a unified abstracted view to the upper layers, which makes the design more scalable. The **5Gr-VoMS** extends the 5GT monitoring platform by integrating vertical-application level monitoring probes and providing enhanced monitoring to support innovative mechanisms related to reliability (via self-healing and auto-scaling), control-loop stability, and analytical features (such as, forecasting and anomaly detection). The **5Gr-AIMLP** will be presented in detail in Sec. III. It is designed to support different 5Growth layers to run AI/ML algorithms for their decision-making processes.

III. Framework for Closed-loop Automation

A. System design

5Growth introduces a new building block in its architecture, the 5Gr-AIML platform, in charge of model lifecycle management (including model uploading, catalog building, and model training). First, its main functionalities are explained, and then, its integration with the rest of building blocks of the 5Growth system.

1) 5Gr-AIML platform: We have designed an AIML as a Service (AIMLaaS) platform, the 5Gr-AIMLP, that allows for the exploitation of AI/ML models for the various decision-making processes necessary in a 5G management and orchestration stack, among which NFV-NS scaling. The models can be uploaded to the 5Gr-AIMLP by any authorized external user. Such models can be already trained and onboarded to then perform inference, or they can be yet-to-be-trained models. In the latter case, the user can provide a suitable data set to be exploited for the training phase. In both cases, the user can specify (i) the scope of the model, i.e., the type of decision-making process to be used for, such as service scaling, and (ii) the type of service the model/dataset should be used for, such as digital twin. When the external user uploads a yet-to-be-trained model, it is the platform that takes care of the training and records the corresponding timestamp and, potentially, a validity time lapse. If no dataset is uploaded along with the yet-to-be-trained model, the 5Gr-AIMLP can exploit the data collected through the 5Gr-VoMS platform about network/computing resource utilization or performance. The configuration of the monitoring platform to gather the monitored data, its aggregation (e.g., through Kafka), and their feeding as input for real-time model execution in the corresponding 5Growth building block also need to be properly set up. Models stored in the 5Gr-AIMLP can be accessed by a 5Growth architecture entity through a Representational State Transfer (REST) interface.

Beside a web interface for the interaction with authorized external users, the 5Gr-AIMLP includes the following main components:

- the *Model Registry*, which records the models uploaded to the platform, their metadata, and pointers to the stored models and associated files;
- the *Lifecycle Manager*, which is in charge of the models lifecycle. Upon the uploading of a new model, it adds the corresponding entry to the Model Registry and, if it is a yet-to-be-trained model, it triggers the training process using the appropriate AI/ML framework. After a model is trained, the Lifecycle Manager monitors its status: it can trigger a new training job either periodically, or whenever new data are available from the monitoring platform;
- the *Interface Manager*, which processes the requests for AIML models coming from the architectural stack and forwards them to the proper block inside the computing cluster.
- the *Computing cluster* is based on Apache Hadoop[2], and leverages Yet-Another-Resource-Negotiator (YARN) for the computing resources management, and the Hadoop Distributed File System (HDFS) for the storage of datasets and models. The YARN cluster nodes have access to different AI/ML frameworks, according to the requested model type. Spark[3] is used to train classic supervised and unsupervised models, BigDL[4] is used for deep neural networks, and Ray[5] can be used for reinforcement learning models.

2) Integration in the 5Growth system: The 5Gr-AIMLP has been integrated into the 5Growth system (which includes the 5Growth MANagement and Orchestration (MANO) stack, the monitoring platform and the infrastructure) by defining a workflow aiming at SLA fulfillment of the services under deployment as well as at the correct operation of the underlying infrastructure. In fact, the AIMLaaS offering provided by the 5Gr-AIMLP can be exploited at any layer of the 5Growth stack where decision-making is required (e.g., for slice arbitration at the 5Gr-VS, for automated NFV-NS scaling at the 5Gr-SO, or for automated path restoration at the 5Gr-RL). To do so, the following elements have been developed in the framework.

First, a complete data engineering pipeline must be deployed. For instance, in the automated scaling SLA management case, *data collection* is done at the virtual machines (VMs) to monitor relevant ey Performance Indicators (KPIs) (e.g., CPU load) that are gathered by the 5Gr-VoMS. *Data ingestion* schemes should also be in place (e.g., Kafka) to aggregate in a Kafka topic collected data of interest for the problem to be *consumed* by the model, which will *analyze* the

[2]https://hadoop.apache.org/
[3]https://spark.apache.org/
[4]https://bigdl-project.github.io/
[5]https://ray.io/

data to make an operational decision. A high-level design of this step was first presented in [11].

Second, the data consumption needs are offered through the 5Gr-VoMS. As part of the instantiation process of an NFV-NS, the 5Gr-SO parses its Network Slice Descriptor (NSD), which includes an information element (IE) with the list of metrics to monitor. The 5Gr-SO creates a monitoring job in the 5Gr-VoMS, which will eventually request the exporters in the VMs to start collecting data. Another IE of the NSD lists the metrics that are needed for a given problem (e.g., scaling) to which an AI/ML model is applied. Therefore, the 5Gr-SO creates the relevant Kafka topic that, by interacting with the 5Gr-VoMS, is fed with all relevant metrics for the problem at hand. A preliminary design and implementation thereof was presented in [10].

Finally, the model matching the problem needs for the NFV-NS being deployed is requested to the 5Gr-AIMLP, downloaded (together with auxiliary files to make it run), and installed in the SLA manager of the 5Gr-SO. After that, the SLA manager subscribes to the relevant Kafka topic for this problem and starts feeding the AI/ML model with it towards real-time decision-making (e.g., to modify the instantiation level (IL) of the NFV-NS). In this sense, this paper improves the previous steps and adds the full integration with the 5Gr-AIMLP to realize a full-fledged operational platform with all the components and APIs that enable their interactions.

Once the AI/ML model is running, the SLA manager compares the current IL of the service with that inferred by the AI/ML model (i.e., the one the NFV-NS should be running to deal with a given CPU load) and, in case there is a difference, a scaling operation is triggered (e.g., scale out/in if more/less resources are needed). When this happens, the system takes care automatically of stopping/starting/reconfiguring the data engineering pipeline and monitoring jobs as needed.

We remark that the 5Gr-AIMLP is designed to run multiple AIML models (e.g., supervised or reinforcement learning). Below, we provide an example of the integrated operation of all the building blocks, including the 5Gr-AIMLP, 5Gr MANO stack, and 5Gr-VoMS, acting over the shared infrastructure towards SLA fulfilment through a supervised learning model running at the 5Gr-SO.

B. Testbed results for the digital twin use case

We now look at the interaction between the 5Gr-AIMLP and the 5Gr-SO for service scaling, and assess the corresponding latency (a preliminary analysis was presented in [10]). To run the experiments, we use a complete implementation of the 5Growth platform where we deploy a Digital Twin (DT) NFV-NS, as relevant representative of Industry 4.0 services [12]. The DT application is composed of two VNFs. VNF1 creates a rendering of the connected robots based on the geographical coordinates received from the robots themselves, allowing a human user to control the robots' movements through a joystick. VNF2 translates the commands inserted by the human user into instructions for the robots. Between the two, VNF2 is the most critical, as its processing time must be below 60 ms to ensure a timely delivery to the robots of the movement instructions.

To meet such target latency, NFV-NS presents multiple ILs, each corresponding to a different number of VNF2 instances. At deployment-time, the DT NFV-NS consists of one instance per VNF, i.e., IL=1. However, as the number of robots to control increases, so does the CPU load and the processing latency of VNF2, the NFV-NS IL must be properly scaled out to balance the CPU load over multiple instances of VNF2 and keep its processing time below the target values. Likewise, as the CPU load decreases, the NFV-NS IL should be scaled in to save computational resources. A real-time decision on the IL to adopt is therefore required, so as to trigger a scaling (out/in) operation as needed.

The 5Gr-SO does so by using an AI/ML model (a random forest algorithm) that is downloaded from the 5Gr-AIMLP during NFV-NS instantiation and that is fed with the real-time CPU load values provided by the 5Gr-VoMS all the way through the data pipeline. The AI/ML model is trained offline within the 5Gr-AIMLP using Apache MLlib and a training dataset with 318K entries (total size: 20.2 MB). The resulting model is packed in a zip archive of 86.5KB (46.4 KB are devoted to the auxiliary file needed for inference).

TABLE I: 5Gr-AIMLP operation and interaction with 5Gr-SO

5Gr Entity	Operation	Measured Time
5Gr-AIMLP	Dataset upload (offline)	10.3 ± 2.0 s
	Training (offline)	35.6 ± 2.5 s
5Gr-SO	Model/Aux file Downl. (instant.)	281.9 ± 20.4 ms
	AIML-related ops. (instant.)	1367 ± 40.9 ms

Tab. I presents the measured times (average and standard deviation) over 10 repetitions of the experiments In general, the time required by the 5Gr-AIMLP to upload and train the model (offline process) depends on the file size and the server used; under the above settings and using an Intel i7-4790 CPU with a 32 GB-RAM, it is around 46 ± 4.52 s. During NFV-NS instantiation, the 5Gr-SO requires 281.9 ± 20.38 ms to download the files from the 5Gr-AIMLP (online process), which represents a total of 20.6% of the average time devoted by the 5Gr-SO to set up the data engineering pipeline for AI/ML-based scaling operations [10], which in this evaluation is 1367 ± 40.9 ms . However, configuring the pipeline is just a small part of the instantiation process, with VNF deployment being instead the most time-consuming step [9]. In fact, the impact of AIML-related operations on the whole instantiation process is limited to about 3.5% of the total average instantiation time measured at the 5Gr-SO (39.78 ± 3.24 s) (the unit in this case is s and not ms).

TABLE II: AIML-based scaling operation in 5Growth

Operation	Measured Time	AIML impact
Scaling out	26.72 ± 2.75 s	9.55%
Scaling in	24.54 ± 2.58 s	10.32%

Next, we focus on the impact of AIML operations during subsequent scaling operations at the 5Gr-SO. When the real-

time inference process notifies the SLA Manager with a different IL, the scaling workflow is started, as described in [10]. Note that during scaling, there is no need for interaction with the 5Gr-AIMLP. Table II presents the time required to perform a scaling out/in operation, i.e., moving from one to two instances of VNF2 and from two to one instance, respectively, and the percentage of time devoted to AIML operations in the scaling workflow. The impact of AIML operations is now around 10%, i.e., much more than in the case of instantiation. This is mainly due to the time it takes to stop the inference process while a new VNF instance is created/removed and the rest of the NFV-NS (connections and monitoring jobs) is updated accordingly. Stopping such process takes around 2.49 ± 0.17 s and it is performed to avoid overruling of a decision on the IL in the middle of a scaling workflow execution (i.e., in a transient period).

IV. AI-driven Scaling of eMBB Services

The continuously changing network dynamics of 5G and beyond networks call for continuous monitoring of the network traffic load and scaling accordingly the network services. To this end, the current section proposes a Deep Neural Network (DNN)-based scheme, and specifically a Multi-Layer Perceptron (MLP)-based one, for proactively scaling services, focusing on eMBB, according to the respective network requirements and real-time traffic load. The proposed MLP-based scheme predicts the Instantiation Level (IL), in terms of the number of required VNF instances per VNF type, in order to accommodate the traffic load changes ahead of time and satisfy the respective network service demands.

A. Control Model

The current work investigates how to map traffic load statistics X_s of a service $s \in S$ served via a specific base station, to instantiation levels Y_s of the VNF, in a supervised manner. Let traffic load statistics $x_t \in X_s$, $s \in S$, denote a vector consisting of traffic load measurements, as well as user equipment (UE)-related information, in a specific period of time t. In addition, let $y_t \in Y_s$, $s \in S$, denote the IL required in order to proactively accommodate traffic load demands of the next time period $t+1$. The target of the MLP model is to learn a function $f : X_s \rightarrow Y_s$ so that $f(x)$ accurately predicts the corresponding value of y.

The proposed MLP-based scheme, which is based on [13], considers a set of data features (X_s), related to the traffic load of each eMBB service and how this load evolves over time. The selected features are listed below:

- 5G New Radio gNodeB Base Station ID (denoted gNB ID)
- Average number of associated, active UEs
- Traffic load in packets and bytes (for each gNB ID) for five time periods ($[t-2, t+2]$)
- Change in traffic load in packets and bytes (received by cell) for the five aforementioned time periods.

For each eMBB service s, an MLP model is deployed in order to identify relevant patterns, while mapping features X_s to autoscaling decisions Y_s. The output class of each MLP model refers to the service IL required to meet traffic demands until the next scaling decision (after 5 time periods). In case the host does not have enough capacity to meet the predicted service IL for the VM, then the maximum supported IL is selected.

B. Results

In order to validate the performance of the proposed model, the ns-3 discrete-event network simulator was used. The deployed scenario, with total duration equal to 12000 sec, includes 12 UEs connected to a specific gNB, running two simulated eMBB services (Fast Browsing (FB), UHD Video Streaming (UHDVS)), with different data rates and traffic volumes. The base station is interconnected with a MEC server, hosting one Virtual Machine (VM) with predefined specifications (i.e., number of vCPUs, memory and disk size, CPU speed, etc.). This study applies the proposed scheme to a part of an eMBB service, represented by a pre-configured VNF type (see Table III). Service-related measurement traces (e.g., packet time interval, packet size, packets/bytes received from a cell, etc.) are logged in the form of time-series information, while the logging frequency is set to 1 second. The time period t is set to 5 seconds.

Parameter	**FB**	**UHDVS**
VNF type	vnf1	vnf2
VNF Throughput (packets/sec)	250	150
Packet Interval (ms)	20	10
Packet Size (bytes)	300	1000

TABLE III: Service Template

Since, ns-3 currently does not support the modeling of VNF instances, the correlation between the simulated data and ILs is modeled under a threshold-based logic. More specifically, it is assumed that the traffic load (in packets) is equally distributed among the VNF instances running on a VM host. Each VNF instance is able to handle specific throughput (packets/sec). Every 1 second, given the traffic load in packets (produced by ns-3), the CPU load of each VNF instance is defined by the division of the traffic load by the VNF throughput. In addition, every second, the described algorithm verifies whether the average $cpu\ load$ of all running VNF instances is greater than a predefined upscale threshold thr_{up} (set to 0.8) or less than/equal to a predefined downscale threshold thr_{dw} (set to 0.5) of the VNFs' throughput in order to upscale or downscale to the appropriate IL (by dividing the CPU load of each VNF instance by $thr_{up/dw}$). The resulted ILs comprise the set Y_s that is provided as a new extracted label set to the respective MLP model.

In order to increase the performance of the MLP models, the Stochastic Gradient Descent (SGD) with momentum was selected, with learning rate equal to $\eta = 0.001$. All models used L2 (weight decay) regularisers with regularization factor of 10^{-4}. The split ratio for the dataset is set to 0.8. Figure 1 illustrates the two timelines of the predicted and the real (ground truth) labels of the test set for both VNF types. As it

can be inferred, the predicted and actual timelines present high similarity for both VNF types. The latter is validated by Table IV, which depicts the performance of the two MLP models using accuracy and f1-score as metrics. More precisely, the accuracy is equal to 88.98% and 90.38% respectively, while the f1-score amounts to 88.89% and 90.14% for VNF1 and VNF2 respectively.

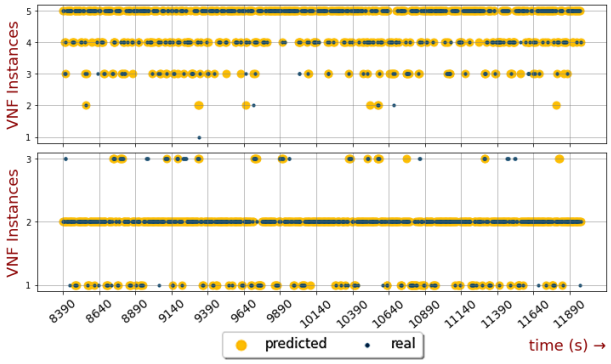


Fig. 1: Prediction - Groundtruth Timeline for vnf types: **VNF1** (up) and **VNF2** (down).

VNF type	Accuracy	F1-Score
vnf1	88.98%	88.89%
vnf2	90.38%	90.14%

TABLE IV: Model Performance

V. AI-driven Scaling of V2N Services

In this section we consider a C-V2N (cellular vehicle-to-network) environment where cars exchange information with the cloud via Points of Presence (PoPs) that are located throughout a city. Whenever a car enters into the neighborhood served by a PoP, its workload needs to be supported by the computational resources of that PoP. These resources come in discrete chunks expressed as the number N of CPUs (central processing unit) that have been activated at a certain moment in time. It is the job of the 5Gr-RL to determine that number of CPUs, as cars travel through the city, and hence, as the workload that is presented to that PoP fluctuates over time.

In the simulation that we discuss in this section, the number of cars that enter the neighborhood of a PoP is driven by a trace that was taken in Torino (Italy) from January 2020 to October 2020, counting the number of cars that passed certain measuring points over consecutive intervals of 5 minutes. The trace contains over 100 measuring points, but in this paper we only use one. The complete trace was divided in a training trace spanning the first 80% of the trace and a test trace spanning the remaining 20%. The training trace is used to determine the variables of the controllers, while the test trace is used to assess its performance.

We assume that each car brings the same amount of work and that the total workload W brought by all cars (during a 5 minute interval) is distributed over the N currently active CPUs with weights that randomly deviate a little from $\frac{1}{N}$. If more work is brought than the N CPUs can handle (in this 5 minute interval) a backlog is created that impairs the performance of the application. Consequently, if no backlog is built up, the load on each CPU is on average $\frac{W}{N}$. This performance is expressed in a reward function that depends on the CPU loads and the backlog. The reward function has the property that it increases as the CPU loads increase, but decreases as the backlogs increase.

The aim of the 5Gr-RL is to maximize the long term reward by setting the number N (i.e., the IL) of CPUs adequately. To make that decision the controller is provided with information from the monitoring system associated to the PoP as input. In particular, the algorithms discussed below use as input a subset of the CPU loads, the backlogs, and the work that was presented to the PoP in the previous 5 minute slot.

A. Scaling Approaches

We consider three classes of scaling algorithms:

- Classical algorithms, which observe (via the monitoring system) the load of the heaviest loaded CPU and try to keep this load around a given threshold by setting N appropriately.
- Prediction algorithms, which observe the evolution of the amount of work over time, make a prediction for the amount of work in the next slot and use this prediction to set N.
- Reinforcement learning algorithms, which learn the long-term reward that actions (which determine N) yield in a certain state and gradually choose the actions that yield the largest long-term reward.

B. Results

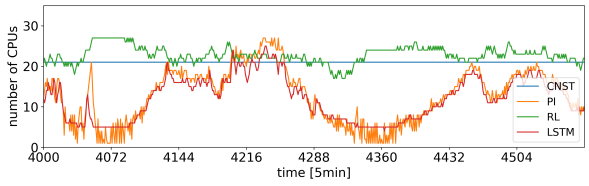
As benchmark we use a simulation (labelled with 'CNST') in which the number N of CPUs is kept constant over time. Simulations over the training trace with various values of N revealed that $N = 21$ yielded the best average reward.

As classical algorithm we use a Proportional Integral (PI) controller [14]. It aims to keep the load of the heaviest loaded CPU ρ to a threshold ρ_{tgt}. Therefore, it first calculates $\delta(t) = \alpha(\rho(t) - \rho_{tgt}) + \beta(\rho(t) - \rho(t-1))$ and then adjusts the number of CPUs for the next slot to $N(t+1) = N(t) + \delta(t)$, suitably bounded below by a minimum number of CPUs and above by a maximum number of CPUs. Via simulations on the training trace we concluded that values $\rho_{tgt} = 0.6$, $\alpha = \beta = 5.0$ bring the highest average reward. We used these values on the test trace.

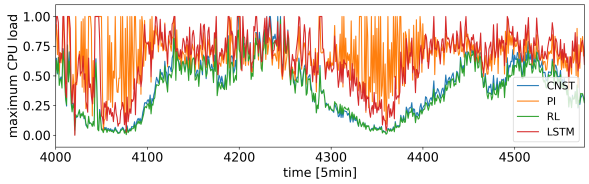
As prediction algorithm we use a LSTM (long short-term memory) predictor [15] with 2 layers with 4 cells each and we use a look back of 3 slots (i.e., a prediction is made based on the 3 previous values). The LSTM predictor takes as input the amount of work $W(t)$ brought in a 5 minute slot t. The LSTM predictor is trained based on the evolution of that amount of work that it sees during the training trace. Subsequently, for the test trace, it predicts the amount of work $\tilde{W}(t+1)$ expected in the next slot and sets the number of CPUs for the next slot as $N(t+1) = \frac{\tilde{W}(t+1)}{\rho_{tgt}}$. Via simulation on the training set we concluded that $\rho_{tgt} = 0.5$ is a good value.

The reinforcement learning (RL) algorithm [16] also uses the amount of work $W(t)$ brought in the current slot, but truncates this to the nearest integer. This truncated value together with the number $N(t)$ of CPUs during the current interval forms the "state" of the RL algorithm. We define three "actions": increase $N(t)$ by 1, decrease $N(t)$ by 1 and keep $N(t)$ the same. By taking an action in a certain state the RL algorithm collects a reward. It adds this reward to the long-term discounted reward it received up to now (with a discounting factor γ) to obtain the new value of the long-term reward. By running through the training trace (as many times as is needed for convergence) the RL algorithm maximizes this long-term reward. The discount factor is set as $\gamma = 0.9$.

Figure 2 illustrates how the four algorithms perform over a period of two days (or equivalently, 576 5-minute intervals). The RL controller is the most conservative: it is reluctant to decrease the number of processors, but it hardly ever overloads any CPU. The PI controller is the most aggressive: it activates a sufficient number of CPUs and while frequently a CPU gets overloaded, this is mostly of short duration. Thus the backlog incurred does not usually lead to a low reward. The LSTM controller has a behaviour between these both extremes. Figure 3 compares the average reward the controllers get to the average number of CPUs they activate over the test trace. Based on these metrics the LSTM controller performs best.

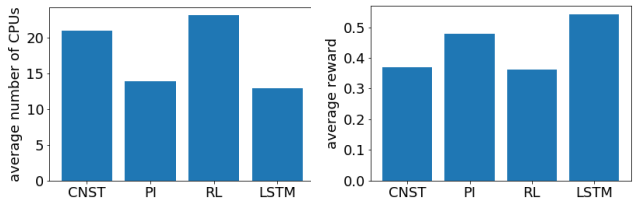


(a) Number of CPUs.



(b) CPU load.

Fig. 2: Evolution of performance metrics over time.



(a) Number of CPUs. (b) Reward.

Fig. 3: Average performance metrics.

VI. Conclusion

This paper presents a novel design of the AI/ML platform and its integration into the 5Growth service management software platform for a closed-loop service automation system. As a proof-of-concept, we develop a data-driven AI-based service scaling prototype to automate the service scaling operation to meet the service requirements while minimizing the consumption of resources. The developed prototype was demonstrated with a network service of an Industry 4.0 use case. The results show the impact of the AIML-related operations on the total service operation time is below 10%. Furthermore, a number of additional ML-based approaches are designed for auto-scaling elastic resources and service instances of network slice(s), which can be embedded to the system for handling more complex use cases such as eMBB and V2N scenarios.

Acknowledgments

This work has been partially supported by EC H2020 5GPPP 5Growth project (Grant 856709).

References

[1] D. M. Gutierrez-Estevez *et al.*, "Artificial intelligence for elastic management and orchestration of 5g networks," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 134–141, 2019.

[2] D. Tsolkas and S.-A. Charismiadis, "Managing computational elasticity for 5g networks," *Wiley 5G Ref: The Essential 5G Reference Online*, pp. 1–18, 2019.

[3] L. Zanzi *et al.*, "Ovnes: Demonstrating 5g network slicing overbooking on real deployments," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops*. IEEE, 2018, pp. 1–2.

[4] International Telecommunication Union (ITU) - Focus Group on Machine Learning for Future Networks including 5G. https://www.itu.int/en/ITU-T/focusgroups/ml5g/Pages/default.aspx/. Accessed on: 15 January. 2021.

[5] ETSI Zero touch network and Service Management (ZSM). https://www.etsi.org/technologies/zero-touch-network-service-management/. Accessed on: 15 January. 2021.

[6] ETSI Experiential Network Intelligence (ENI). https://www.etsi.org/technologies/experiential-networked-intelligence/. Accessed on: 15 January. 2021.

[7] J. A. Ayala-Romero *et al.*, "vrAIn: A Deep Learning Approach Tailoring Computing and Radio Resources in Virtualized RANs," in *Proc. of ACM MobiCom 2019*.

[8] EU 5G-PPP 5Growth Project: 5G-enabled Growth in Vertical Industries. https://5growth.eu/.

[9] X. Li, T. Deiss *et al.*, "Automating vertical services deployments over the 5gt platform," *IEEE Communications Magazine*, vol. 58, no. 7, pp. 44–50, 2020.

[10] J. Baranda *et al.*, "On the integration of ai/ml-based scaling operations in the 5growth platform," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2020, pp. 105–109.

[11] C. Papagianni *et al.*, "5Growth: AI-driven 5G for Automation in Vertical Industries," in *2020 European Conference on Networks and Communications (EuCNC)*, 15-18 June 2020, Dubrovnik, Croatia, Jun. 2020. [Online]. Available: https://doi.org/10.1109/EuCNC48522.2020.9200919

[12] L. Girletti, M. Groshev and C. Guimarães and Carlos J. Bernardos and Antonio de la Oliva, "An intelligent edge-based digital twin for robotics," in *2020 IEEE Globecom Workshop on Advanced Technology for 5G Plus (GC 2020 Workshop - AT5Gp)*, Taipei, Taiwan, Dec. 2020.

[13] T. Subramanya and R. Riggio, "Machine learning-driven scaling and placement of virtual network functions at the network edges," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 414–422.

[14] Kiam Heong Ang, G. Chong, and Yun Li, "PID control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.