# EdgeBOL: A Bayesian Learning Approach for the Joint Orchestration of vRANs and Mobile Edge AI

Jose A. Ayala-Romero*, Andres Garcia-Saavedra*, Xavier Costa-Perez*†, George Iosifidis‡
*NEC Laboratories Europe GmbH, Germany, †i2CAT Foundation and ICREA, Spain
‡Delft University of Technology, Netherlands

***Abstract*—Future mobile networks need to support intelligent services which collect and process data streams at the network edge, so as to offer real-time and accurate inferences to users. However, the widespread deployment of these services is hindered by the unprecedented energy cost they induce to the network, and by the difficulties in optimizing their end-to-end operation. To address these challenges, we propose a Bayesian learning framework for jointly configuring the service and the Radio Access Network (RAN), aiming to minimize the total energy consumption while respecting accuracy and latency service requirements. Using a fully-fledged prototype with a software-defined base station (vBS) and a GPU-enabled edge server, we profile a typical video analytics service and identify new performance trade-offs and optimization opportunities. Accordingly, we tailor the proposed learning framework to account for the (possibly varying) network conditions, user needs, and service metrics, and apply it to a range of experiments with real traces. Our findings suggest that this approach effectively adapts to different hardware platforms and service requirements, and outperforms state-of-the-art benchmarks based on neural networks.**

## I. Introduction

### A. Background and Motivation

There is a growing consensus that the next generation of mobile networks need to support AI and other *intelligent* services at the edge. These services typically require the collection, transfer, and processing of data flows in near-real time, with the aim to provide data operations, e.g., inferences, to end users such as small IoT devices, drones, or smartphones on the go. A representative example of these services are mobile video analytics (MVA), which are used in AR/VR services [1], cognitive assistance applications [2], surveillance systems [3], among other similar AI services. The core task of MVA is that user devices send video frames to the network, which needs to process them and transmit back accurately-detected depicted objects, or extract other important information [4].

While MVA-like services are already considered a utility that each user should be able to enjoy, their wide deployment requires a fundamental shift in the way we manage mobile networks. Namely, in these services, the network's role is not confined to transferring data from one point to another, nor even in processing the data en route. Instead, the network needs to directly optimize the service performance, which involves the criteria of accuracy (confident inferences), end-to-end latency (fast inferences), and task throughput (inferences/sec) in a resource-efficient fashion. This latter requirement is crucial since such services create voluminous data flows, involve heavy computations, and consume large amounts of energy [5]. In fact, energy consumption is not only one of the most prevalent operating expenditures for mobile networks [6][1] but has been also identified as the main blocker for the success of these inherently energy-demanding services. Besides, energy is the common resource consumed by all network operations (e.g., data transmissions, transfer, or computing) and its efficient management is imperative also from a performance point-of-view.

Clearly, in order to deploy these services we need to develop a systematic methodology for energy-aware control of the involved communication and computing network resources. To that end, it is imperative first to understand how these services perform; which system parameters shape their performance and resource requirements; if there are sweat spots in the performance/cost function; and how to optimize the network and the service so as to exploit these opportunities. Our goal in this work is exactly this: to assess experimentally these effects and design a learning-based orchestration framework that manages the resources of base stations and intelligent services following two joint optimization goals: (*i*) minimize the energy toll associated with the service and network; and (*ii*) meet service performance targets.

In particular, we face several challenges when addressing this problem. First, there is a strong coupling among the elements of our system (user device, base station, and edge server) shown in Fig. 1. Both the base station and the server have an individual energy cost for the service provider[2]. Moreover, the configuration of these three elements has a joint impact on service performance. If we focus on one element and minimize the power consumption of the base station, for example, we would need to increase the computing capacity

The work was supported by the European Commission through Grants No. SNS-JU-101097083 (BeGREEN) and 101017109 (DAEMON). Additionally, it has been funded by NextGeneration EU through UNICO I+D with Grants No. 2022/0005395 (CLARION) and 2022/0004810 (SORUS) and CERCA Programme. *(Corresponding author: Jose A. Ayala-Romero)*

J. A. Ayala, and A. Garcia-Saavedra are with NEC Laboratories Europe GmbH, Heidelberg, Germany.

X. Costa-Pérez is with NEC Laboratories Europe GmbH, Heidelberg, Germany, and i2CAT Foundation and ICREA, Barcelona, Spain.

G. Iosifidis is with Delft University of Technology, Delft, Netherlands.

[1]For instance, China Mobile committed to reduce the overall energy consumption per unit of telecom business by no less than 6% in 2021 [7], and Verizon and Vodafone have set targets to reach net zero energy emissions by 2040 [8].

[2]Note that, in contrast to most previous works focused on the energy consumed by the user device (e.g., [9], [10]), we focus on the energy consumption at the network infrastructure (base station and edge server).

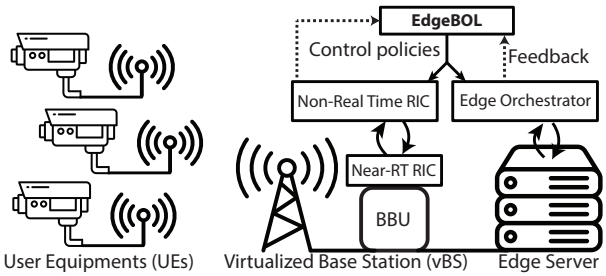


Fig. 1: **Scenario and System Architecture**. Virtualized base stations are configured jointly with the computing infrastructure hosting the service. in order to optimize its performance (accuracy and delay) and reduce the aggregate (vBS and server) energy costs.

of the edge server to compensate for the network delay, which incurs higher overall energy consumption. For that reason, it is very important to consider the *system as a whole*. Note that his particular setting, which is enabled by the new O-RAN architecture, is not standard in the literature.

Second, there is no unique optimal configuration for the system. Instead, the optimal configuration depends on the context (e.g., wireless channel conditions, number of users) and changes over time. For example, poorer wireless links can increase the delay (due to the need for lower modulation levels that render lower data rates) and the system needs to compensate for this by allocating more radio resources (bandwidth or channel time), which increases the computational power in the base station. Another option to compensate for the delay is to increase the computational capacity in the edge server so the processing time is reduced. In consequence, we need to find the mapping between contexts and optimal configurations. This is not straightforward because the relationship between configurations and performance indicators is both highly non-linear and unknown a priori. We show this in our experimental analysis in Section III.

Third, the many parameters involved in our system render a high-dimensional solution space, which also depends on the contextual information. The exploration of this context-dependent solution space may not be efficient enough for most of the previous machine learning solutions (e.g., deep reinforcement learning). For that reason, we rely on Bayesian online learning, which is intrinsically data efficient and can handle high dimensional context and solution spaces. However, Bayesian online learning solutions are relatively unexplored in the context of mobile networks. Moreover, the Bayesian online learning literature considering contextual information and constraints in the objective function is very limited and cannot be directly applied to this problem.

Finally, the considered system needs to handle a variable number of users. A variation in the number of users usually implies a change in the dimensionality of the problem, which is very challenging for ML models. Besides, an increase in the dimensionality of the problem can reduce the convergence time due to the curse of dimensionality. Thus, a novel strategy to handle the number of users needs to be proposed.

### *B. Methodology and Contributions*

We have built a fully-fledged prototype system with a software-defined base station (vBS) (using srsRAN suite [11]) and a GPU-enabled edge server that offers an MVA service to mobile users. We measure the joint impact that resource control policies at the user device (video frame resolution), the vBS (radio configuration), and the server (GPU speed) have on the service accuracy and end-to-end latency (QoS metrics), and on the overall power consumption (cost metric).

Our experiments show that, unlike other services, the service performance is highly volatile and depends on the platform hardware (hosting the vBS and the server), the service configuration parameters, and even the actual user data. This renders previous model-based optimization approaches practically inefficient (details in §II). Furthermore, these services include a wide range of configuration options, e.g., selecting different service models (e.g., neural network architecture), different processing equipment, or adjusting the data sources (e.g., image frame sizes). All these parameters affect in an unknown way the latency and accuracy performance, and hence cannot be optimized in a static and a priori fashion. Finally, our experiments show non-trivial and non-linear trade-offs between the configuration parameters and the performance indicators (details in §III) and are also platform-dependent, making it impractical for model-based solutions.

In order to overcome these challenges, we propose an optimization framework for orchestrating jointly the edge service and the RAN. *Our solution is not bound to any specific application or underlying technology*, i.e., it is agnostic to the RAN technology (e.g., 4G, 5G), the hardware in the edge server, the image recognition model (Faster R-CNN, RetinaNet, YOLOv3, etc.) or the edge service. For that purpose, we propose a non-parametric *Bayesian online learning* algorithm, named `EdgeBOL` (Edge Bayesian Online Learning).

Our solution relies on the use of Gaussian Processes (GPs) as non-parametric function approximators. We use different GPs for the objective and constraint functions and they intrinsically handle noisy observations. Then, we make decisions based not only on the estimated value of the functions but also on the uncertainties of the estimations. This allows us, on the one hand, to efficiently handle the exploration-exploitation trade-off toward finding the optimal configurations and, on the other hand, to provide safety in the decision-making in order to meet the performance targets. This strategy also exhibits extraordinary data efficiency compared to other approaches based on neural networks. To make our solution work in practice, we introduced two main modifications with respect to previous works. First, we propose an acquisition function that finds higher performance configurations and at the same time expands the safe set (collection of configurations meeting the constraints) by exploiting the structure of our problem. In contrast, previous works propose acquisition functions that explicitly expand the safe set with theoretical guarantees but obtain poor performance in practice. Second, we propose a statistical characterization of the channel states of the users as input contexts. Thus, we mitigate several problems such as the dimension explosion of the input as the number of users

increases and the variability of the input size as a function of the number of users. We evaluate empirically this statistical characterization showing near-optimal performance.

The architecture of our solution is shown in Fig. 1. `EdgeBOL` selects the configuration policies to be used at the vBS and the edge server. The objective is to minimize the power consumption of the overall system while satisfying hard service accuracy and latency requirements. Finally, we verify the efficacy of `EdgeBOL` using a prototype, real datasets, and compare `EdgeBOL` with state-of-the-art (SoA) benchmarks [12]. Our solution shows an unprecedented convergence speed and satisfies the service (stochastic) constraints with a very high probability.

To summarize, the key contributions of this paper are:

- We propose the new problem of jointly controlling RAN resources and intelligent edge service parameters, aiming to minimize the overall energy cost while meeting hard performance targets.
- We build a prototype and conduct extensive experiments with a representative MVA application that reveal new trade-offs about the service performance and the system energy consumption.
- We design a Bayesian learning framework that learns in real time the optimal configuration of the system, adapting to different contexts, user needs, and services.
- Finally, we evaluate `EdgeBOL` using our experimental platform and actual datasets, and compare it with SoA neural network-based reinforcement learning algorithms.

This work has been partially published in a preliminary conference version [13].

**Paper Organization**. §II discusses related works; §III presents motivating experimental findings; in §IV we formulate the orchestration problem and in §V introduce the learning algorithm which is evaluated in §VI. We conclude in §VII.

## II. Related Work

**Edge services & MVA.** Some recent works study the deployment of AI services, specifically, MVA at the network edge. Some of them focus on performance optimization by using different strategies such as variable encoding, caching, visual tracking, or adaptive compression [14], [15], [16], [17]; or study the trade-off between accuracy and latency [18], [19], [20]. Other works, in turn, address the resource orchestration in MVA systems. For example, [21] and [22] search greedily in real-time for the most resource-prudent configuration; [23] and [24] allocate computing resources and decide the image compression (or, video quality) and neural network model; Although some works like [25], [9], [10] consider the energy in MVA services, all of them only account for the energy consumption at the user device. In contrast, we address the orchestration of O-RAN systems (vBS and edge server) to minimize their consumed power.

**Mobile network orchestration.** There are some works that study network orchestration in mobile networks. For example, [26] select the modulation and coding scheme (MCS) and airtime to maximize throughput, using predetermined models for the operation functions. Other studies propose model-free approaches, e.g., for slicing [27], throughput forecasting [28], and energy cost reduction [29]; but suffer from the lack of (accurate) training data. Reinforcement learning (RL) is also used for interference coordination [30], network diagnostics [31], or SDN control optimization [32]. However, RL lacks convergence guarantees. Finally, the authors in [33] present a reliable virtualized base station design. However, none of these works account for the coupling of the mobile network with the edge servers. Our experiments reveal that their joint orchestration is imperative.

**Bayesian online learning in networking.** In this paper, we propose a fundamentally different approach for controlling the virtualized Base Station (vBS) and the edge server, as it relies on Bayesian contextual bandit algorithms. Such techniques have been employed to adjust video streaming in mobile networks [34], to minimize the power consumption in vBS [35], to optimize BS handovers [36], and to control mmWave networks [37]. Perhaps the most closely related work to ours is [12], which assigns CPU time to virtualized BSs, but this focuses on data transfers (not accuracy or end-to-end latency). Besides, we follow a different algorithmic approach based on the contextual Bayesian online optimization [38] that combines bandit exploration with Gaussian Processes (GPs) [39]. Thus, we use the uncertainty in the estimations provided by the GPs to efficiently explore new configurations. We extend this approach here by including service-related quality of service constraints and accounting for the vBS and server power consumption.

## III. Experimental Analysis

We have performed an exhaustive set of experiments using the testbed described in detail in §VI-A. In a nutshell, the testbed is comprised of a 3GPP R10-compliant LTE base station (BS), a user equipment (UE) generating service requests via the BS to a well-known object recognition service, and an off-the-shelf server with an NVIDIA GPU running the service. Each request consists of an image with a variable number of objects from the COCO dataset [40]. The images are sent to the service via the uplink channel of the LTE interface, and the service returns to the user a bounding box and a classification label for each identified object in the image. This information is sent via the downlink channel of the LTE interface. Each measurement shown as a dot in the figures of this section is an average of 150 images. The dataset collecting the measurements of this section is available online[3]. Note that, although our testbed implements an MVA application, our framework is flexible enough to be used with any other edge service non-related to image processing. See Appendix C for more details.

Motivated by O-RAN specifications [41], we focus on configuration *policies* set by an orchestrator that operates at a second-level timescale (Non-Real-Time RAN Intelligent Controller in O-RAN). These policies decide rules that must be respected by lower-level controllers that operate at millisecond-level timescale — and which are orthogonal to our study. In the following, we analyze the trade-offs between different *configuration policies* and *performance indicators*: (*i*)

[3]https://github.com/jaayala/energy_edge_AI_dataset

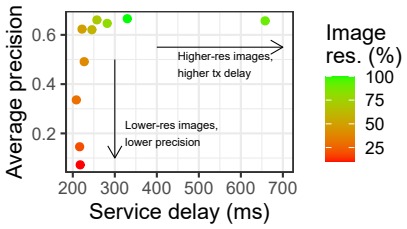


Fig. 2: Mean average precision (mAP) *vs.* service delay for images with different resolutions.

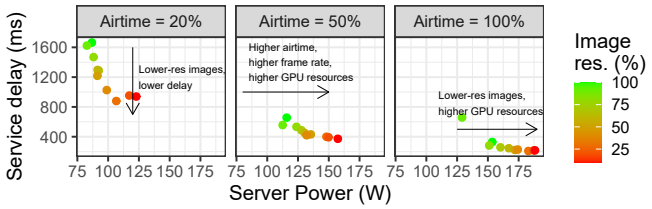


Fig. 3: Service delay *vs.* server's power consumption for images with different resolutions and radio policies.

QoS experienced by users; (*ii*) the energy associated with the service; and (*iii*) the network energy cost.

**Precision, delay and image resolution.** We start analyzing two metrics of interest for QoS: the service performance to recognize objects and the service delay, formally introduced in Performance Indicator 1 and 2, respectively.

*Performance Indicator 1 (Service delay):* End-to-end delay that includes the image pre-processing at the user side, its transmission, the processing at the server (GPU delay), and the return of the bounding boxes and labels.

*Performance Indicator 2 (Mean Average Precision):* The service accuracy is quantified using the Mean Average Precision (mAP) [42]. On the one hand, precision is defined as the ratio of true positives over all positive classifications. On the other hand, the recall measures how well these positives are identified by calculating the ratio between true positives over the sum of true positives and false negatives. The Intersection over Union (IoU) measures the overlap between the calculated bounding box and the ground truth. IoU values above a threshold (set to 0.5 here) trigger a true positive. Then, for a given set of images, the Average Precision (AP) corresponds to the area below the precision-recall curve. Finally, mAP is the mean AP over all object categories, hence ranging from 0 (worst performance) to 1 (best performance).

Based on our measurements, the most relevant feature that affects the mAP is the *image resolution*, see Policy 1.

*Policy 1 (Image resolution):* This policy sets the average encoding of every image (number of pixels) which can be enforced by the service. In our experiments, the maximum (100%) resolution is $640 \times 480$ pixels. Note that, at any given time, the resolution of an image may be actually larger or smaller than the policy, as long as the average across the whole period and users respects the threshold.

We illustrate this in Fig. 2, which shows the trade-off between delay and mAP for the COCO dataset images encoded with different resolutions. The remaining configuration policies (described later) are fixed. The findings reveal interesting and quantifiable trade-offs: (*i*) Higher-resolution images carry more pixels encoded in a larger amount of data; thus, they incur a higher delay due to longer transmission time over the radio interface. (*ii*) Lower-resolution images cause the service to provide lower mAP performance because they carry less information for the object detection engine. Specifically, we measured a 72% improvement in delay at the expense of precision reduction ranging between 10% to 50%.

**Delay, energy consumption, and radio policies.** There also exists a trade-off, which naturally appears in many resource control problems [43], between QoS and overall energy costs. To explore this trade-off, we introduce a policy that governs the allocation of radio resources (Policy 2); and an additional metric that assesses part of the aforementioned cost: the server's power consumption (Performance Indicator 3).

*Policy 2 (Radio Airtime):* This radio policy imposes a constraint on the radio resources (duty cycle) the vBS allocates to the service traffic. The MAC layer radio scheduler, which operates at msec granularity then must allocate radio resources (which may vary across users based on their channels) such that the threshold set by the policy is respected. Due to the nature of this service, we focus on uplink communication.

*Performance Indicator 3 (Server power consumption):* Power cost associated with the computational load of the service's requests, which is dominated by the GPU power consumption.

Fig. 3 depicts the service delay *vs* the server power consumption, for different airtime radio policies and image resolutions. As before, higher-res images increase service delay due to the longer transmission time of requests. We now observe that this occurs irrespective of the radio policy configuration. However, the radio policy has an important impact on delay as well. This is rather expected since lower airtime implies lower usage of radio resources, which further increases the transmission time of the requests at the radio interface. Specifically, our experiments show that an 80% increase in the airtime improves the delay between 65% and 80%. Concerning the server's power consumption, lower-res images and lower radio resource allocations increase this cost. Specifically, there is a 56% increase in power consumption for an 80% increase in radio time resource; a similar increase is attained when there is a 75% increase in image resolution. This is due to the fact that increasing the radio resources allows the user to send a higher rate of requests in a similar way than low-res images do, which ultimately increases the workload assigned to the service's resources (the GPU, in this case). Note that this is a consequence of the variable frame rate considered in our testbed. A fixed frame rate is a particular case that simplifies the complexity and the coupling among the parameters of the system.

**Delay, energy costs, and service policies.** We study the impact of the computing allocation policies on QoS. To this end, we define an additional configuration policy.

*Policy 3 (GPU speed):* The server's policy is a GPU power

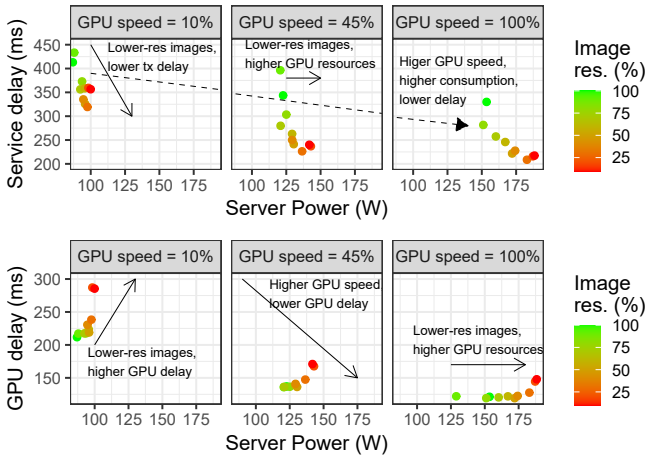


Fig. 4: Delay *vs.* server's power consumption for images with different resolutions and GPU policies.

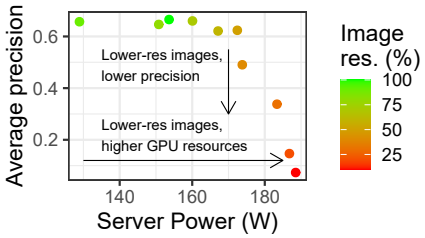


Fig. 5: Mean average precision *vs.* server's power consumption for images with different resolutions. The radio and computing resources are allocated so as to minimize the delay.

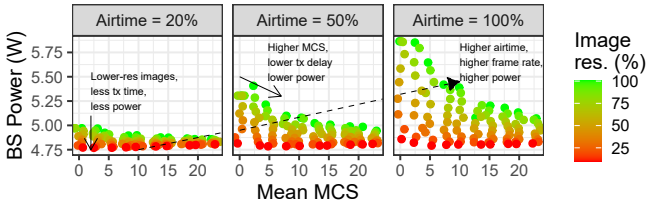


Fig. 6: BS power consumption *vs.* radio policies for images with different resolutions.

limit that adapts the processing speed of a GPU (or a pool of GPUs) in a slice to meet the adopted power constraint. The GPU controller (e.g., NVIDIA driver) may change the GPU speed at any given time (e.g., for different video frames) as long as the GPU power set by this policy is respected.

In our experimental setup, the GPU speed can be set through a configuration parameter available in Nvidia GPU drivers. Fig. 4 (top) depicts the service delay and the server's power consumption for several image resolution configurations. We now fix the airtime to 100% and vary the policy allocating computing resources. A higher amount of computing resources increases the server's power consumption, as we are relaxing the power limit imposed to the GPU. We observe that low-res images contribute to increasing the server's power consumption as the rate of requests also grows. However, it is interesting to note that higher-res images ease the work on the GPU, as evidenced by Fig. 4 (bottom), which shows the delay associated with the GPU tasks only. All in all, despite this improvement in the GPU delay, the corresponding increase in transmission delay when using higher-res images dominates. It is important to observe that, while this is true in our experimental testbed, it may well be different for diverse deployments (e.g. a more energy-efficient GPU, or a higher-bandwidth radio access network). This motivates the need for *learning* algorithms that adapt to the different deployments.

**Precision, energy consumption, and image resolution.** The above trade-off between service delay and the server power consumption, certainly appears for other performance metrics, such as the mAP. To assess this, Fig. 5 shows the mAP achieved by the service as a function of the server's power consumption for various image resolutions. The findings confirm the service cost depends on the mAP. Importantly, however, the relationship with mAP is substantially different from that with the service delay. In this case, higher mAP performance actually requires *less* server power consumption. The reason lies in the fact that higher-res images (which render higher mAP) facilitate object detection and hence require less computation, see Fig. 4 (bottom).

**BS power cost, radio policies, and image resolution.** Finally, the costs associated with the network operator are necessarily driven by the amount of radio resources invested into the service's pipeline. To analyze this, we introduce an additional policy, motivated by [12] in the context of virtualized RANs, which is defined as Policy 4, and Performance Indicator 4, reflecting part of operational costs of the network operator.

*Policy 4 (Radio MCS):* This policy imposes a constraint on the maximum MCS eligible by the vBS to transport the service's data over the air. And we note that the MCS selected by the MAC layer may be lower than this bound for some users depending on their channel state.

*Performance Indicator 4 (Base Station power consumption):* Power consumption associated with processing the baseband unit in a virtualized RAN environment.

To analyze these, we plot in Fig. 6 the power consumption measured at the baseband unit of the vBS for various airtime and MCS policies and image resolutions. We first observe that lower-resolution images consume less radio resources and hence less vBS power. Second, using larger radio resources (airtime) actually induces higher power costs because it allows the user to transmit images at a higher rate. Finally, and perhaps surprisingly, higher MCS policies cause *lower* BS power consumption. The reason is the data load at the BS is relatively low compared to the bandwidth available at the vBS, e.g., higher-res images with 100% airtime generate up to 2.8 Mb/s, compared to a capacity of around 50Mb/s (SISO LTE @ 20MHz bandwidth). In this scenario, despite the fact that LTE subframes modulated with higher MCS incur higher instantaneous power consumption, they process the load faster, which pays off in terms of long-term power consumption.

From these results and the BS's point of view, there is no reason to use MCS lower than the maximum possible. However, this depends also on the network load, which may

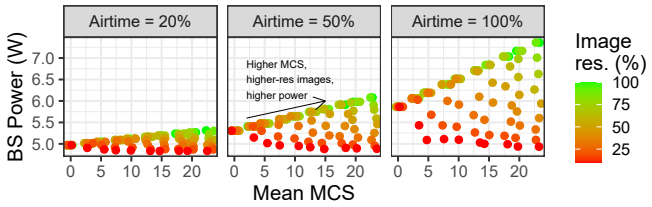


Fig. 7: BS power consumption *vs.* radio policies for images with different resolutions and 10x higher load.

be very different for, e.g., multiple users or other services. To demonstrate this, we emulate a scenario with 10x more load, and present the same plot in Fig. 7. Differently now, we observe that the MCS policy has a negative impact on the BS power consumption for higher-resolution images whereas lower-resolution images cause lower power consumption for higher MCS policies. This motivates the need for learning algorithms that adapt the system to the service requirements.

**Conclusion:** Our system consists of a large number of intertwined parameters with non-trivial effects on the performance and energy consumption. As a consequence, we resort to model-free contextual bandit methods to design a controller that adapts autonomously to context changes and the vBS and server hosting platforms. We have summarized the experimental findings from this section in Table I.

## IV. System and Problem Formulation

### A. System Set Up

We consider a GPU-powered edge server providing an AI service through a radio access network. Specifically, we consider an object recognition service that can be used, for instance, for security surveillance or fault detection in industrial chains. We assume that a slice dedicated to this service is created including virtualized Base Station (vBS) and the edge server [44], [45]. This is illustrated by the orange boxes in Fig. 8. The service operation is as follows: users capture images that are sent to the edge server through the uplink of the radio interface of the vBS. Then, the server's GPU processes the incoming data and generates a response, which is sent back to the users through the vBS downlink.

The workflow of `EdgeBOL` is also simple: `EdgeBOL` periodically observes the context (we provide an appropriate definition later), orchestrates the resources assigned to the wireless access and the GPU-powered service via a set of *control policies*, and uses a *cost* metric aggregating key performance indicators of the system to make better decisions over time. To this end, we follow closely the framework of O-RAN [46], a carrier-led alliance of operators and manufacturers to build open and intelligent RAN solutions [41].

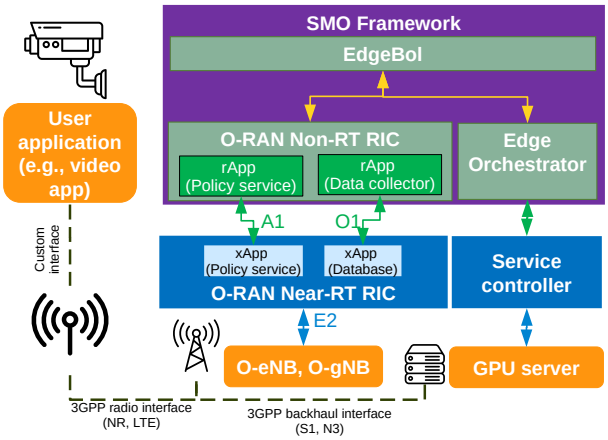


Fig. 8: O-RAN compliant system architecture.

As shown in Fig. 8, `EdgeBOL` interacts with O-RAN's Non-Real-Time RAN Intelligent Controller (RIC) to enforce radio control policies in O-RAN compliant eNBs or gNBs:

- An rApp (within O-RAN's non-RT RIC), as defined in [47], interacts with the learning agent and handles O-RAN's A1 interface (specifically, the A1's Policy Management Service) as specified in [48] to deploy the MCS and radio airtime policies defined above.
- An xApp handles the A1-P service from O-RAN's near-RT RIC side, and uses an E2 interface to forward radio policies to the base station, including O-DU, O-CU, and O-RU in case of 5G (see Sections 4.3.4-4.3.6 in [46]), and O-eNB in case of 4G (see Section 4.3.7 in [46]).
- The E2 interface, defined in [49], is also used to gather vBS KPIs (power consumption, in our case), which is forwarded to the non-RT RIC through the O1 interface. Then, a second xApp manages data KPIs received from the vBS, which in our case consists of samples of its power consumption, and forwards it to the learning agent.

We assume that both the O-eNB/O-gNB and the GPU server can implement the configured policies, namely through radio scheduling at the MAC layer for the former, and a driver such as NVIDIA's for the latter. Note that our objective is to propose a solution that is agnostic to the implementation detail of the underlying system. For example, the RAN technology (4G or 5G), the image recognition model (Faster R-CNN, RetinaNet, YOLOv3, etc.), the fixed/variable frame rate, or even the specific application (security cameras, defect detection in production lines, etc). Next, we formulate the problem as a contextual multi-armed bandit or *contextual bandit*.

### B. Definitions

Let us formally define the elements needed to formulate the online learning problem in the next section. These elements are the contexts, actions, and performance indicators.

| Control Policy | Impact on Performance Indicators |
|---|---|
| Image resolution | Higher img. resolution implies higher delay, better mAP, and lower GPU delay. |
| Airtime | Higher airtime implies lower delay and higher server and BS consumed power. |
| GPU speed | Higher GPU speed implies higher server consumed power and lower delay. |
| MCS | Higher MCS reduces the consumed power at the vBS with low traffic or the opposite with high traffic. |

TABLE I: Summary of experimental findings.

**Contexts.** We define the context at each time period $t$ as $c_t := [n_t, \bar{c}_t, \tilde{c}_t] \in C$, where $n_t$ is the number of users in the slice, and $\bar{c}_t$ and $\tilde{c}_t$ are the mean and the variance of the UL CQI across all users in the slice during the previous period, and $C$ is the context space. Note that we consider a statistical characterization of the channel states of the users, which avoids dimensionality problems for the learning algorithms. This design decision is detailed in Appendix A and experimentally evaluated in §VI-D

**Control policies.** Let $\mathcal{H}$ denote the set of possible image resolutions; $\mathcal{A}$ the set of possible airtime configurations (uplink radio resources) that can be assigned; $\Gamma$ the possible GPU speed configurations; and $\mathcal{M}$ the set of all possible MCS policies (characterizing the data rates) as defined above. Hence, we let:

$$x_t := [\eta_t, a_t, \gamma_t, m_t] \in \mathcal{X} := \mathcal{H} \times \mathcal{A} \times \Gamma \times \mathcal{M}$$

denote the control policy selected at time period $t$. The GPU speed is configured in the same machine where the learning agent runs, the airtime and the MCS policies can be sent to the vBS through the A1-P interface of O-RAN architecture [46], and the image resolution is indicated to the user using the application of the service. We assume that the RAN technology can be configured in terms of time domain and data rate (airtime and MCS in this formulation, respectively). Other radio technologies are out of the scope of this paper. We focus on uplink radio policies because, as our experiments confirm, such AI services have little impact on the downlink as the data surge goes usually upstream with only simple information (bounding boxes, labels) flowing downstream. Note that the policies in $\mathcal{X}$ jointly control parameters from the user device, the vBS, and the edge server. These three elements are highly coupled (as we show in §III) and for that reason should be configured at once.

**Performance indicators.** Similarly, our performance indicators were introduced in §III. The service delay experienced by user $i$ is denoted by $D_i(c, x)$, and the mAP is denoted by $Q_i(c, x)$. We then define:

$$d(c, x) := \max_i D_i(c, x), \quad \text{and} \quad \rho(c, x) := \min_i Q_i(c, x),$$

to be the highest delay and lowest mAP, respectively, across all users. The consumed power at the edge server is denoted by $p^s(c, x)$, and the consumed power at the vBS is denoted by $p^b(c, x)$. Besides, the observation of the performance indicators is noisy in practice due to, for instance, random perturbations of the equipment or transient periods in the network. Importantly, as we detail in the next section, our solution intrinsically handles this noise. Henceforth, we denote by $d_t(c_t, x_t)$, $\rho_t(c_t, x_t)$, $p^s_t(c_t, x_t)$, and $p^b_t(c_t, x_t)$ the noisy observations of our performance indicators at time period $t$. Feedback from the data plane components including all these performance metrics is received by `EdgeBOL` at the end of each time period $t$, as explained above. We assume `EdgeBOL` is working in a pre-production phase where the labels of the images are available for training. Alternatively, we can easily integrate other real-time precision metrics that consider the confidence output of the object recognition algorithms [50].

### C. Online Learning Problem Formulation

Energy consumption is one of the main operational cost components of mobile networks, and its impact is only expected to grow further with the deployment of AI/ML services that raise further this toll. This has been made explicit in a number of reports from vendors, manufacturers, and operators [7], [8]. Hence, our goal is to minimize the power consumption of the whole system (vBS and edge server) subject to the performance constraints of the service. Depending on the form factor of the vBS and the configuration of the server (i.e., GPU model, motherboard, etc.) the consumed energy of each entity may have a different order of magnitude. Moreover, the cost associated with energy consumption may vary depending on the scenario. In regular small-cell based scenarios, such cost may be related to the price of electricity, which may vary between day and night depending on the rates set by the power suppliers in each country. In other scenarios, such as those based on Power over Ethernet (PoE) or a solar-powered vBS, this cost may reflect the scarcity of the energy resource for the RAN. In order to capture these different scenarios, we define the following *cost function*:

$$u(c, x) = \delta_1 p^s(c, x) + \delta_2 p^b(c, x) \tag{1}$$

where $\delta_1$ and $\delta_2$ are the costs of the power at the edge server and the vBS, respectively, in monetary units per watt (mu/W).

On the other hand, we consider performance constraints at the service level, going a step beyond other works considering lower-level performance requirements (e.g., [12]) such as data rate or delay. The mapping between context-action pairs and the service-level performance indicators is very complex and there are no available models, as we detailed in the experimental results of §III. For that reason, we learn them from observations. For our object recognition service, we consider two constraints: (*i*) a maximum service delay denoted by $d^{max}$, which is directly related to the frame rate[4] (number of images per second) that the user is going to process, and (*ii*) a minimum mAP denoted by $\rho^{min}$ which indicates a lower bound on how accurate is the service in detecting the objects. We formulate the problem as follows:

$$\begin{aligned} \min_{\{x_t\}_{t=1}^T \in \mathcal{X}} \quad & \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} u_t(c_t, x_t) & \\ \text{s.t.} \quad & d_t(c_t, x_t) \leq d^{max}, & \forall t \leq T \\ & \rho_t(x_t) \geq \rho^{min}, & \forall t \leq T. \end{aligned} \tag{2}$$

Note that the service constraints are satisfied for the user experiencing the worst service as $d(c, x) := \max_i D_i(c, x)$ and $\rho(c, x) := \min_i Q_i(c, x)$. It is worth mentioning that these functions are unknown beforehand and platform-dependent. That is, the values of performance indicators may change with the software implementing the vBS, the hardware of the server and vBS, and the service running at the edge server. In §III we show that the trade-offs between the control policies and the

[4]We consider a scenario with variable frame rates as some works show that this can provide substantial benefits to the performance of the service [51], [52], [53]. However, we can also consider a fixed frame rate as a particular case of this formulation.

performance indicators for one user are non-trivial and non-linear. With a larger number of users, these relations can be even more complex and exhibit different behavior, making it essential the use of online learning.

For that reason, we use online learning in order to learn how to configure the system based on the observations (data-driven). Moreover, there is no unique optimal configuration, as it depends on the context. This renders a very challenging problem as the set of contexts is not finite and the observed contexts change over time. In order to learn the mapping between the context and the optimal configuration, we formulate the problem as a contextual bandit with constraints detailed in the next section.

We would like to remark that other alternative formulations can be considered in eq. 2. For instance, we could consider power-constrained vBSs or an edge computing power budget by including the power consumption targets as constraints, while minimizing latency and maximizing accuracy. The flexibility of our framework allows us to implement any of these different formulations with minimal changes.

### D. Contextual Bandit Formulation

The contextual bandit formulation is a particularization of the well-known Reinforcement Learning (RL) formulation with several differences:

First, RL formulation considers that the probability distribution of the state/context at $t+1$ depends on the state/context and the selected action at $t$. Conversely, the contextual bandit formulation considers that the context does not depend on the selected actions. Note that the contexts in our problem (number of users and channel quality) cannot be affected by the configuration of the system.

Second, RL formulation considers that reward can be sparse or delayed, i.e., the selected actions will affect the reward an arbitrary number of time steps later. In some cases, the reward is only revealed at the end of the episode and the algorithm should distribute the credit of this outcome across all the actions in the episode. In contrast, the contextual bandit formulation considers that the reward is instantaneous, i.e., for a given state-action pair at time $t$, we can observe the associated reward instantaneously. Furthermore, the decisions made at time $t$ will not have an impact on future time steps. This property holds in our problem due to the timescale of the decision-making process. Recall that our solution operates in the Non-Real Time RIC of the O-RAN architecture (timescale of seconds). Based on this time scale, the performance observation at each time $t$ is independent of previous time steps as the system becomes stationary.

Finally, it is worth mentioning that the contextual bandit approach does not need to estimate the accumulated rewards until the end of the episode as it is needed in RL (using Temporal-Difference Learning, Monte Carlo Methods, etc.). These simplifications in the formulation make our solution simpler and more effective.

The contextual bandit $\pi_t(c_t; \mathcal{B}_{t-1}) : \mathcal{C} \rightarrow \mathcal{X}$ is an algorithm that maps contexts to actions, where $\mathcal{B}_{t-1} = [c_1, x_1, u_1, d_1, \rho_1, \ldots, c_{t-1}, x_{t-1}, u_{t-1}, d_{t-1}, \rho_{t-1}]$ is the set of historical observations at $t-1$ (i.e., triples of contexts, actions, and performance metrics). At each time period $t$, a context $c_t$ is observed and an action $x_t = \pi_t(c_t; \mathcal{B}_{t-1})$ is computed and applied to the system. At the end of the time period $t$ the performance indicators associated with the pair $(c_t, x_t)$ are measured to update $\mathcal{B}_t$ and consequently $\pi_t$. The contextual bandit $\pi_t$ is sequentially updated with each new measurement from the system towards solving the problem in eq. (2). In §V, we detail the contextual bandit algorithm and its updates.

## V. Bayesian Online Learning in Practice

`EdgeBOL` is an online learning algorithm that solves the problem defined in §IV-C and formulated as a contextual bandit in §IV-D. Most of the existing contextual bandit algorithms assume a linear relationship between the contexts-control space and the associated reward [54]; or assume a certain structure in the reward function [55]. However, as the measurements in §III reveal, our performance metrics have a non-linear and unknown curvature, but we do observe a high correlation with the control policies. That is, a small change in one of the policies (e.g., image resolution) will produce a small change in delay and power. This allows us to get information about unobserved context-control points via nearby points, hence reducing the exploration time.

Based on the above points, we propose a Bayesian online learning method that models the cost and constraint functions as samples of Gaussian Processes (GPs) over the joint context-control space. This non-parametric estimator deals with the aforementioned non-linearities and correlations, and quantifies the function estimation uncertainty, addressing effectively the exploration *vs.* exploitation trade-off.

**Function approximator.** In order to estimate the cost and constraint functions we use GPs, which consist of a collection of random variables that follow joint Gaussian distributions [39]. Let $z \in \mathcal{Z} = \mathcal{C} \times \mathcal{X}$ denote a context-control pair. We model each of the unknown functions as a sample from $GP(\mu(z), k(z, z'))$, where $\mu(z)$ is its mean function and $k(z, z')$ denotes its kernel or covariance function. W.l.o.g., we assume $\mu := 0$ and $k(z, z') < 1$, which we refer to as the *prior distribution*, not conditioned on data. Given the prior distribution and a set of observations, the *posterior distribution* can be computed using closed-form formulas.

The sets of observations of the cost and constraint functions at points $Z_T = [z_1, \ldots, z_T]$ up to time period $T$ are denoted by $y_T^{(0)} = [u_1, \ldots, u_T]$, $y_T^{(1)} = [d_1, \ldots, d_T]$, $y_T^{(2)} = [\rho_1, \ldots, \rho_T]$, respectively, assuming i.i.d. Gaussian noise, $\mathcal{N}(0, \zeta_{(i)}^2)$. The posterior distribution of these functions follows a GP distribution with mean and covariance:

$$\mu_T^{(i)}(z) = k_T^{(i)}(z)^\top \left(K_T^{(i)} + \zeta_{(i)}^2 I_T\right)^{-1} y_T^{(i)} \tag{3}$$

$$k_T^{(i)}(z, z') = k^{(i)}(z, z') - k_T^{(i)}(z)^\top \left(K_T^{(i)} + \zeta_{(i)}^2 I_T\right)^{-1} k_T^{(i)}(z') \tag{4}$$

where $k_T^{(i)}(z) = [k^{(i)}(z_1, z), \ldots, k^{(i)}(z_T, z)]^\top$, $K_T^{(i)}(z)$ is a kernel matrix defined as $[k^{(i)}(z, z')]_{z,z' \in Z_T}$, $I_T$ is the $T$-dimension identity matrix, and $\zeta_{(i)}^2$ the variance of noise in observations. Index $i$ denotes the objective function, with $i = 0$

for the cost function, $i = 1$ for the delay, and $i = 2$ for the mAP. The distribution of unobserved values of $z \in \mathcal{Z}$ for function $i$ is computed from the prior distribution, vector $Z_T$ and the observed values $y_T^{(i)}$ using (3) and (4).

**Kernel selection.** The kernel shapes the GP's prior and posterior distributions, and thus encodes the correlation of the function values for every pair of context-control points. In other words, the kernel characterizes the *smoothness* of the functions [56]. The properties of the kernel should be thoroughly selected for each specific application and the functions to be learned. We observe in our experimental data analyzed in §III that the performance indicator functions exhibit different smoothness for each dimension (control policy). In order to approximate these functions accurately, we select our kernel function satisfying two properties: *stationarity* and *anisotropicity*. This means that $k(z, z')$ is invariant to translations in $\mathcal{Z}$ but not invariant to rotations in $\mathcal{Z}$. The kernel smoothness for each dimension of function $i$ is encoded in the length-scale vector $\mathcal{L}^{(i)} = [l_1^{(i)}, \ldots, l_N^{(i)}]$, where $N$ is the number of dimensions of $\mathcal{Z}$. The distance between two points based on the length-scale vector is:

$$d^{(i)}(z, z') = \sqrt{(z - z')^\top (L^{(i)})^{-2} (z - z')}, \tag{5}$$

where $L^{(i)} = \text{diag}(\mathcal{L}^{(i)})$ is a diagonal matrix of the length-scale vector. In order to satisfy the properties stated above, we select the Matérn kernel on its anisotropic version [39]. Moreover, following standard practice, we particularize it with parameter $\nu = \frac{3}{2}$ (details in [39]), indicating that the function is at least once differentiable. Thus, the expression of the kernel can be particularized as follows:

$$k^{(i)}(z, z') = (1 + \sqrt{3} d^{(i)}(z, z')) \exp(-\sqrt{3} d^{(i)}(z, z')). \tag{6}$$

Note that although we are using the same kernel for all cost and constraint functions, their hyperparameters differ and depend on each function's shape. In fact, $\mathcal{L}^{(i)}$ and noise variance $\zeta_{(i)}^2$ (eq. (3)-(4)) should be optimized for each function $i$ before running the algorithm, by maximizing the likelihood estimation over prior data. During execution, the hyperparameters remain constant, since otherwise (optimized with newly acquired data) it is not guaranteed the GPs' confidence interval will cover the actual function within, causing the optimization to fall into poor local optima [57].

**Safe set.** It is crucial to identify first which controls satisfy the constraints, which, however, depends also on the context. For instance, when the user's channel quality decreases (the context changes), the user uses a lower MCS, which increases the transmission time hence increasing the service delay. Therefore, the controls that are suitable for high channel quality may not meet the delay constraint with low channel quality. We define the *safe set* as the set of policies that satisfy all the constraints for a given context $c$:

$$S(c) = \left\{ x \in \mathcal{X} \mid d(c, x) \leq d^{max} \wedge \rho(x) \geq \rho^{min} \right\}. \tag{7}$$

Nevertheless, the computation of the safe set is challenging. Firstly, the observations of the performance metrics are noisy due to the stochastic nature of the system (e.g., noise in the measurements, random variations in the performance), as we observed in §III. And secondly, the number of controls $|\mathcal{X}|$ is very large in practice, making it unattainable to explore all possible configurations, for all possible contexts. For these reasons, we use GPs to compute an estimation of the safe set:

$$S_t = \Big\{ x \in \mathcal{X} \;\Big|\; \mu_{t-1}^{(1)}(c_t, x) + \beta \sigma_{t-1}^{(1)}(c_t, x) \leq d^{max} \\ \wedge\; \mu_{t-1}^{(2)}(c_t, x) - \beta \sigma_{t-1}^{(2)}(c_t, x) \geq \rho^{min} \Big\} \tag{8}$$

where $\left(\sigma_t^{(i)}(z)\right)^2 = k_t^{(i)}(z, z)$ (eq. (4)) and $\beta$ is a weight parameter. Note that the safe set changes over time for two reasons. First, it is a function of the context and, therefore, when the context changes, the set of control policies meeting the constraints varies. Second, as we get more observations of the constraint functions their estimated values and uncertainties also change according to eq. (3)-(4), allowing us to compute the safe set more precisely. In other words, at each period $t$, point $z_t$ is observed and vectors $Z_t$ and $y_t^{(i)} \forall i$ are updated. Due to their correlation, the posterior distribution of points near $z_t$ will be updated, hence affecting which controls will be included in the safe set.

**Acquisition function.** It indicates, at each time period $t$, which control $x_t$ shall be used in the system given context $c_t$. This task is crucial for the convergence of the algorithm and needs to interleave an exploration process in order to expand the safe set while seeking a safe control with high performance. Many previous works have proposed acquisition functions for constrained Bayesian optimization [58], [59], [60], but they do not consider contexts. To the best of our knowledge, SafeOpt [59] is the only work using contexts. However, while SafeOpt provides theoretical performance guarantees, we found in our experiments that its acquisition function has overly slow convergence; an issue that has been reported in other works as well, e.g., [61]. Therefore, we expand this approach by using the contextual Lower Confidence Bound (LCB) proposed in [38] as an acquisition function, but *constrained to safe set*, i.e.:

$$x_t = \underset{x \in S_t}{\text{argmin}} \left\{ \mu_{t-1}^{(0)}(c_t, x) - \sqrt{\beta} \sigma_{t-1}^{(0)}(c_t, x) \right\}. \tag{9}$$

Algorithm 1 summarizes the whole workflow `EdgeBOL`. At the beginning of the time period $t$, the context $c_t$ is observed (line 4). Based on the observed context $c_t$ and the vectors $Z_{t-1}$ and $y_{t-1}^{(i)} \forall i$ from the previous time period, the posterior distribution of all the functions is computed using eq. (3)-(4) (line 5). Note that when we do not have observations ($Z_0 = \emptyset$, $y_0^{(i)} = \emptyset, \forall i$) the posterior distribution is equal to the prior distribution. Using the expectation and uncertainty of the constraint functions and eq. (8), the safe set $S_t$ is built (line 6). The control $x_t$ is selected from the safe set $S_t$ based on the posterior distribution of the cost function and the acquisition function (line 7). At the end of the time period $t$, all the performance indicators are observed. Then, the cost function is computed using eq. (1). Finally, the new context-control pair $z_t$, the value of the cost function $u_t(c_t, x_t)$ and the value of the constraint functions ($d_t(c_t, x_t)$ and $p_t(c_t, x_t)$) are added to

**Algorithm 1:** EdgeBOL

1: **Inputs:** Control Space $\mathcal{X}$, kernel $k$, $S_0$, $\beta$, $\delta_1$, $\delta_2$, $\rho^{min}$, $d^{max}$
2: **Initialize:** $Z_0 = \emptyset$, $y_0^{(i)} = \emptyset, \forall i$.
3: **for** $t = 1, 2, \ldots$ **do**
4: Observe the context $c_t$
5: Compute $\mu_{t-1}^{(i)}$, $\sigma_{t-1}^{(i)}$ $\forall i$ using eq. (3)-(4)
6: Estimate the safe set:
$S_t = S_0 \cup \{x \in \mathcal{X} | \mu_{t-1}^{(1)}(c_t, x) + \beta\sigma_{t-1}^{(1)}(c_t, x) \leq d^{max} \wedge \mu_{t-1}^{(2)}(c_t, x) - \beta\sigma_{t-1}^{(2)}(c_t, x) \geq \rho^{min}\}$
7: $x_t = \text{argmin}_{x \in S_t}\ \mu_{t-1}^{(0)}(c_t, x) - \sqrt{\beta_t}\sigma_{t-1}^{(0)}(c_t, x)$
8: Observe $d_t(c_t, x_t)$, $\rho_t(c_t, x_t)$, $p_t^s(c_t, x_t)$, and $p_t^b(c_t, x_t)$ at the end of the time period $t$
9: Compute the cost $u_t(c_t, x_t) = \delta_1 p_t^s(c, x) + \delta_2 p_t^b(c, x)$
10: Update $Z_t \leftarrow Z_{t-1} \cup [c_t, x_t]$
11: Update $y_t^{(0)} \leftarrow y_{t-1}^{(0)} \cup u_t(c_t, x_t)$
12: Update $y_t^{(1)} \leftarrow y_{t-1}^{(1)} \cup d_t(c_t, x_t)$
13: Update $y_t^{(2)} \leftarrow y_{t-1}^{(2)} \cup \rho_t(c_t, x_t)$
14: **end for**

their respective vectors to generate $Z_t$ and $y_t^{(i)} \forall i$ (lines 10-13). The source code of EdgeBOL is publicly available online[5].

Note that EdgeBOL does not expand explicitly the safe set like in other works such as [59], [58]. These works propose an explicit expansion of the safe set by intentionally exploring controls in the boundary. The objective is to converge to the true safe set and therefore to reach the optimal safe control. However, we found that our acquisition function can both minimize the cost function and expand the safe set. The reason is that control policies with lower values of power consumption are usually in the boundary of the constraint (e.g., they are associated with higher service delay). Hence, when the acquisition function explores lower power controls it is indirectly exploring the boundaries of the constraint, reducing its uncertainty and thus expanding the safe set. In other words, the acquisition function exploits the problem structure to efficiently expand the safe set, see §VI.

**Practical Issues**. It is interesting to note that, if the performance bounds (constraints) are very tight and the problem is infeasible, the safe set will converge to the initial safe set, that is, $\lim_{t \to \infty} S_t = S_0$ (since $S_0$ is always included in $S_t$, Algorithm 1 line 5). This might happen only for certain contexts, e.g., for very low channel quality. In any case, EdgeBOL will select control policies from the initial safe set $S_0$, which are intentionally selected to be the ones with the lowest delay, the highest mAP and, therefore, the highest consumed power. On top of that, EdgeBOL is robust to changes in the constraint settings, and hence can adapt if, for example, the operator decides to relax them during the system runtime in order to avoid such infeasibilities. We demonstrate this in the next section. Finally, it is worth mentioning that the computation of the posterior distribution in eq. (3)-(4) is $O(N^3)$. However, we found in our experiments that this does not introduce any delay since we have a wide enough time window to update the control policy, according to O-RAN specifications.

[5]https://github.com/jaayala/constrained_bayes_opt.

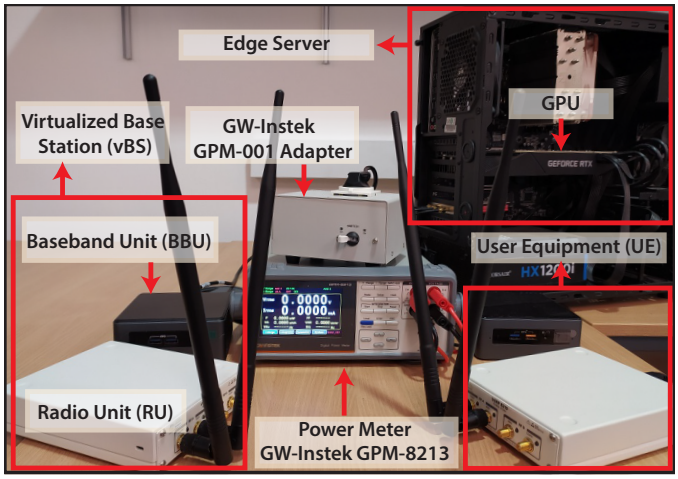


Fig. 9: Experimental testbed.

## VI. Experimental Evaluation

### A. Experimental Setup

Our prototype consists of a vBS, a user equipment (UE), a digital power meter, and an edge server, Fig. 9. The vBS and UE include an NI USRP B210 as radio unit (RU) and a general-purpose computer (Intel NUCs with CPU i7-8559U@2.70GHz) deploying the near-RT RIC (for the vBS) and the baseband unit (BBU), implemented with the srsRAN suite [11] (which emulates an O-eNB for experimentation). The vBS and UE are connected through SMA cables with 20 dB attenuators, and we adjust the transmission gain of the RU's RF chains to attain different uplink SNR values. Without loss of generality, we set 20 MHz bandwidth for the LTE interface.

Our edge server is equipped with a CPU Intel i7-8700K @ 3.70GHz and a GPU Nvidia GeForce RTX 2080 Ti. The vBS and server are connected using a switch with Gigabit Ethernet technology. To measure the power consumption at the BBU and the server, we use the digital power meter GW-Instek GPM-8213 with the GW-Instek Measuring adapter GPM-001. The evaluated AI service is implemented using Detectron2 [62], developed by Facebook, which performs object recognition. Specifically, Detectron2 is configured with a region-based convolutional neural network (Faster R-CNN) [63] comprising a ResNet backbone with conv4 layers and a conv5 head with a total of 101 layers. The UE sends to server images from the COCO data set we used in §III through the LTE uplink. The images are resized at the user side using the OpenCV library in Python. The bounding boxes and object classes are computed by Detectron2 and sent back to the UEs (LTE downlink).

We introduced two key srseNB modifications. First, we modified the radio MAC scheduler to implement the two radio policies of §III. Secondly, we integrated the O-RAN E2 interface as defined in [49] to enforce the radio control policies (MCS and airtime) on the fly and send consumed power consumption samples to the corresponding xApp. For the latter, we have added code into srsRAN to collect this information from the power meter. We have also implemented a proof-of-concept Near-RT RIC and Non-RT RIC with the

interfaces mentioned in §IV and as defined in [47], [48]. We configure the GPU speed by using the Nvidia driver that allows us to set the maximum power management limit, ranging between 100 and 280W. This runtime configuration does not affect the GPU operation. Note that the actual GPU consumed power depends on its duty cycle.

We consider $|\mathcal{H}| = |\mathcal{A}| = |\Gamma| = |\mathcal{M}| = 11$; hence there is a large number of $|\mathcal{X}| = 11^4 \simeq 14.6 \cdot 10^3$ control policies, which, in combination with the effect of the possible contexts, highlights the need for a data-efficient learning mechanism. Given the complexity of running experiments with multiple users, we rely on a single user in most of our experiments (which render trivial low-layer controllers). However, whenever needed (we test out multiple heterogeneous users in §VI-D), we adopt simple controllers (e.g. MAC layer scheduling) that are detailed where relevant. In line with previous works [59], [61], we select $\beta^{1/2} = 2.5$, which shows good performance in our evaluations. We configure the duration of the time periods to 2 seconds. Finally, unless otherwise stated, we will plot our results with lines and shadowed areas representing, respectively, the median value and the $10^{\text{th}}$ and $90^{\text{th}}$ percentiles, across 10 independent repetitions.

### B. Convergence

To evaluate the convergence of `EdgeBOL`, we consider a single context and a certain constraint set with $\rho^{min}$ (minimum mAP performance) and $d^{max}$ (maximum service delay). Dynamic context changes and different constraints are evaluated later. Namely, we set the mean SNR to 35 dB (good wireless conditions), $\delta_1 = 1$ mu/W, $\rho^{min} = 0.5$, and $d^{max} = 0.4$ s. Fig. 10 plots the evolution over time of the cost ($u_t$), mAP performance ($\rho_t$), delay ($d_t$), and server and BS power consumption ($p_t^s$ and $p_t^b$) as a function of $\delta_2$. The first observation is that the cost $u_t$ (top plot) converges within roughly 25 periods across all $\delta_2 = \{1, 2, 4, 8, 32\}$. Higher $\delta_2$ values induce higher cost as the Watt *price* for the BS, grows. Remarkably, both the mAP performance and delay fall within the selected system constraints upon convergence *with high probability*. In fact, we observed consistent results (converge speed, satisfaction of system constraints) irrespectively of the context and system constraints.

The system power consumption presents interesting trade-offs with $\delta_2$. Small $\delta_2$ (e.g., 1) induces high power consumption at the BS but low at the server. This is because the maximum net power consumption of our vBS (around 7.25 W) is much smaller than that of the server (between 85 to 180 W). Therefore, if the associated cost (mu/W) is similar for both the vBS and server, `EdgeBOL` will minimize the power consumption of the server at the expense of a small vBS energy toll. However, when $\delta_2$ is relatively high (e.g., 64), the actual cost associated with the vBS energy footprint becomes comparable, or even higher, to the server. Hence, `EdgeBOL` drives the system to configurations that minimize vBS power consumption at the expense of more server energy. This latter case is relevant for situations when a small cell has a stringent power budget or cooling restrictions. Indeed, different types of vBS have different energy footprints, which motivates the need for approaches that *learn* the relationship between power consumption, performance, and configuration policies.

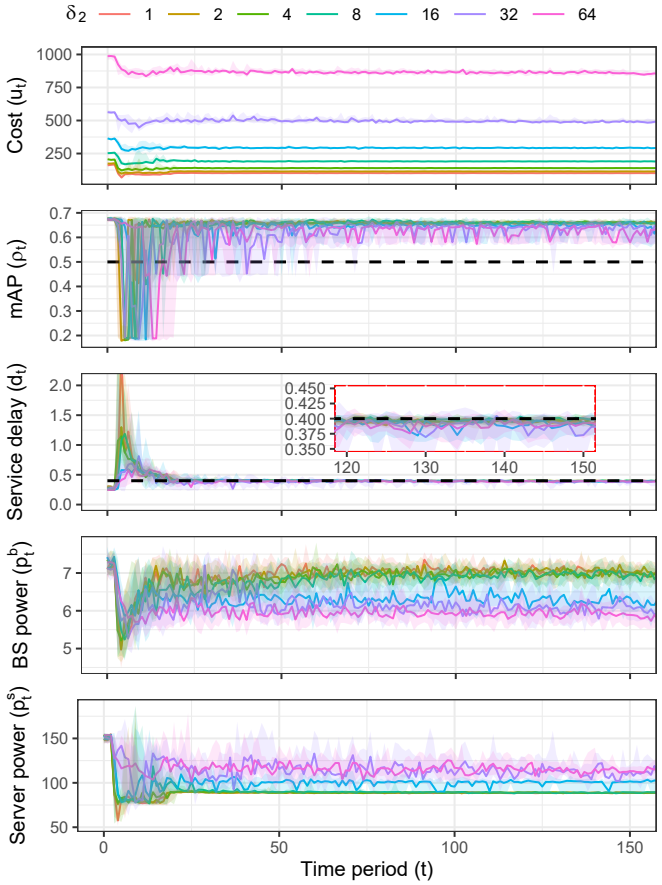


Fig. 10: Convergence evaluation. A scenario with steady channel conditions (no context changes), $\delta_1 = 1$ mu/W, $\rho^{min} = 0.5$, and $d^{max} = 0.4$ s. The dotted black line indicates the service constraint.

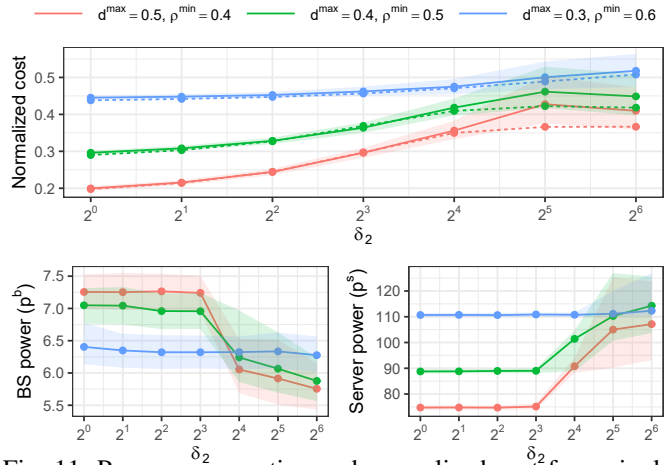


Fig. 11: Power consumption and normalized cost for a single context as a function of $\delta_2$, with $\delta_1 = 1$ mu/W. Dashed lines represent our exhaustive search approach.

### C. Static Scenarios

Let us now take a closer look at the power consumption and the respective `EdgeBOL`'s policies for different constraints and values of $\delta_2$. Fig. 11 shows the power consumption and *normalized* cost once `EdgeBOL` has converged for $\delta_1 = 1$ and

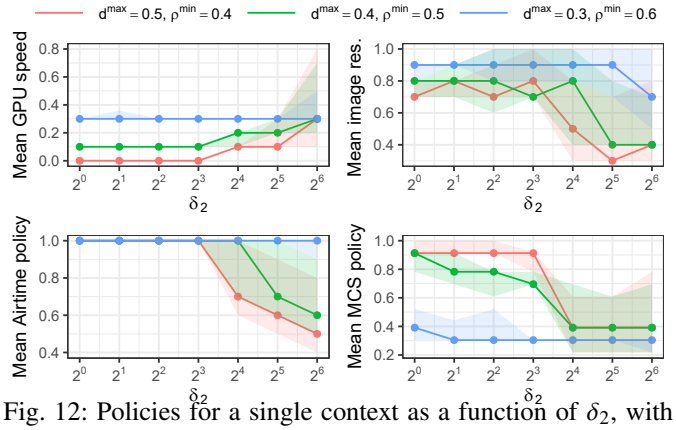


Fig. 12: Policies for a single context as a function of $\delta_2$, with $\delta_1 = 1$ mu/W.

$\delta_2 = \{1, 2, 4, 8, 16, 32, 64\}$ mu/W. We compute the normalized cost independently for each $\delta_2$ so we can compare across different $\delta_2$ values. We now test different constraint settings: (*i*) $d^{max} = 0.5$ s, $\rho^{min} = 0.4$ (lax constraints), (*ii*) $d^{max} = 0.4$ s, $\rho^{min} = 0.5$ (medium constraints), and (*iii*) $d^{max} = 0.3$ s, $\rho^{min} = 0.6$ (stringent constraints), represented in red, green, and blue in the figure. In addition, we represent with dashed lines the cost attainable by an offline oracle, which we obtained using a time-consuming exhaustive search procedure over the whole control space. Though this approach is unfeasible in practice, it is a good benchmark to empirically assess the optimality of EdgeBOL.

Ignoring, for now, the differences across different constraint settings (different colors in the plots), we can make two observations. First, we can confirm our earlier observation that higher values of $\delta_2$ (compared to $\delta_1$) steer EdgeBOL to shift power consumption from the server to the BS (and *vice versa*). Second, EdgeBOL is able to drive the system to near-optimal points of operation, when comparing the cost of EdgeBOL with that obtained by our oracle.

In more detail, the figure renders very different behavior across different constraint settings (colors in the plot). In the case of $d^{max} = 0.5$ s, $\rho^{min} = 0.4$ (lax constraints, in red in the plot), there is a drastic change in the selected policies and resulting power consumption as we increase $\delta_2$. Because these settings are rather lax, EdgeBOL has more leeway to explore (and then select a policy from) a larger space of *feasible* policies. This is made evident when we compare its normalized cost with that of the most stringent settings ($d^{max} = 0.3$ s, $\rho^{min} = 0.6$, blue line in the figure): for $\delta_2 = 1$, the minimum cost attained by EdgeBOL is 25% larger for the latter, and 10% for $\delta_2 = 64$. Moreover, the normalized cost consistently grows, though with a shrinking gap in cost across constraint settings, as we increase $\delta_2$. This occurs because, in our testbed, the range of power values that the BS can consume (across all policies) goes between 4 and 8 W, which is substantially smaller than that of the server (between 50 and 200 W). As a result, when we increase $\delta_2$, i.e., when we increase the importance given to reducing BS power, the cost variance across policies reduces. Needless to say, this may be different with different types of BS such as macro cells.

Finally, Fig. 12 shows the corresponding control policies for

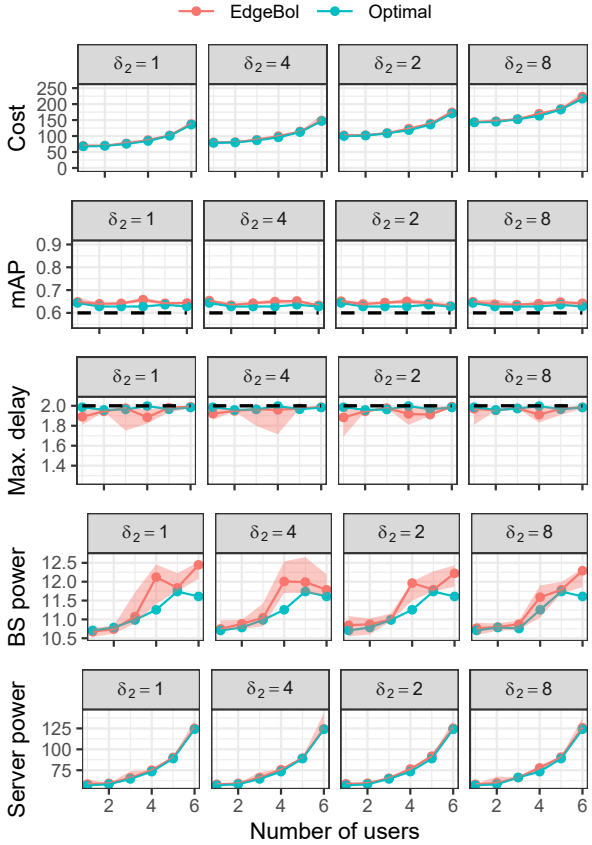


Fig. 13: Empirical optimality gap in scenarios with multiple heterogeneous users. Each scenario has $N$ users with different SNR conditions: user 1 has the best channel conditions (SNR = 30 dB on average) and every additional user has 20% lower SNR. We evaluate different values of $\delta_2$. $\delta_1 = 1$. The dotted black line indicates the service constraint.

the same scenarios shown in Fig. 11. Let us first take a look at the lax settings ($d^{max} = 0.5$ s, $\rho^{min} = 0.4$, depicted with red lines in the figure). When $\delta_2$ is small, EdgeBOL imposes low-consuming server-side policies, i.e., low GPU speed policies. This certainly helps to reduce the server consumption, and the overall cost as a consequence. However, to meet the performance constraints, EdgeBOL has to compensate for low GPU speed policies with higher image resolutions and higher radio policies that ease the job of the service while minimizing delay, which comes at the expense of higher BS power consumption. Conversely, when $\delta_2$ increases, EdgeBOL selects low-consuming radio policies and, to compensate, lower image resolutions and higher GPU speed policies that help reduce service delay. On the other side, for the scenario with the most stringent constraints ($d^{max} = 0.3$ s, $\rho^{min} = 0.6$, blue lines), EdgeBOL is forced to deal with a smaller space of feasible policies. Therefore, all policies are roughly consistent across different $\delta_2$ values (with mild differences for the highest settings).

### D. Heterogeneous Users

In an effort to reduce the problem of the dimensionality, we aggregate statistics of individual users (mean SNR, variance

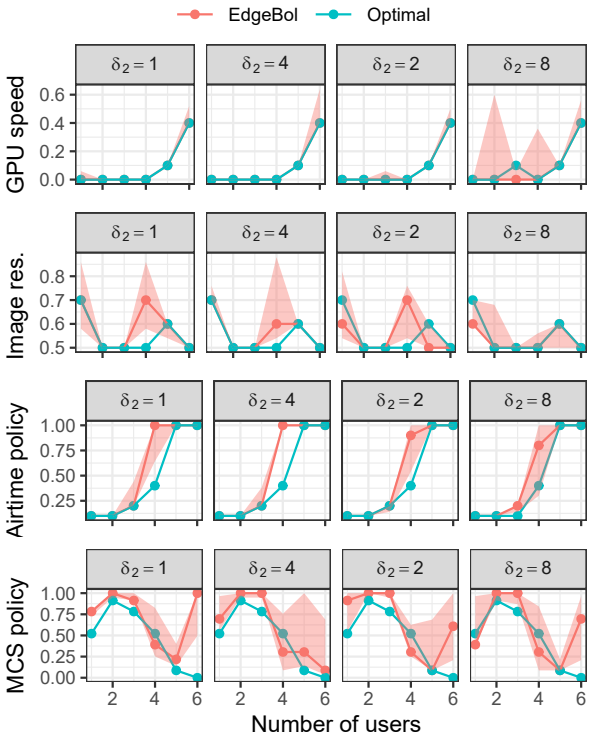


Fig. 14: Optimal and EdgeBOL policies with multiple heterogeneous users. Each scenario has $N$ users with different SNR conditions: user 1 has the best channel conditions (SNR = 30 dB in average) and every additional user has 20% lower SNR. We evaluate different values of $\delta_2$. $\delta_1 = 1$.

SNR, etc.) when describing the context. See Appendix A for a discussion on this issue. To validate that this design choice does not compromise optimality, we have performed a series of experiments with multiple heterogeneous users. Without loss of generality, we adopt a simple low-level control mechanism to enforce the selected policies when allocated resources to individual users: (*i*) a round-robin radio scheduling approach at the MAC layer of the BS, (*ii*) equal image resolution across users, (*iii*) MCS selection approach legacy of srsRAN [11] (upper bounded by the policy), and (*iv*) highest GPU speed to handle individual video frames allowed by the policy.

We train the algorithm with a variable number of *heterogeneous* users with changing channel quality. Once trained, we evaluate the performance of EdgeBOL in scenarios with a fixed number $N$ of heterogeneous users. The first user has the best channel conditions (SNR = 30 dB on average) and every additional user has 20% lower SNR.[6] We trivially choose $d^{max} = 2$ and $\rho^{min} = 0.6$ so the system has a feasible solution in the worst case (with 6 users). Fig. 13 depicts the mean cost of the system (as defined in eq. (1)) and all performance indicators for scenarios with different values of $N$. We do this for different weights $\delta_2$ in the trade-off between the server's power consumption and that of the vBS ($\delta_1 = 1$ in all scenarios). In turn, Fig. 14 shows the selected policies selected in average.

We compare the performance of EdgeBOL with that of

[6]Our experimental setup is constrained to scenarios with $N < 7$.

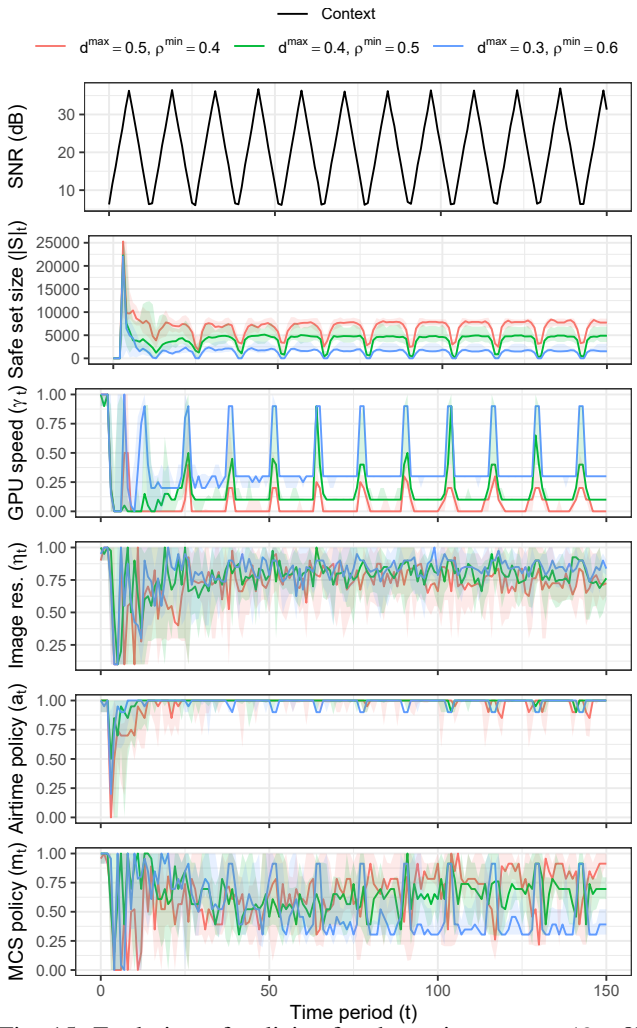


Fig. 15: Evolution of policies for dynamic contexts ($\delta = 8$).

an optimal oracle that finds the best possible combination of policies offline after an exhaustive search where all the system dynamics are known. Hence, though it is unfeasible to use in practice, it provides a lower bound cost that helps us assess the optimality gap of EdgeBOL empirically. The results show that the performance attained by EdgeBOL is remarkably close to that of the oracle, well within 2%. Moreover, EdgeBOL satisfies the service constraints (represented in Fig. 13 with dashed dark lines) with probability 0.98. This validates that aggregated statistics across users suffice to provide good performance while keeping the problem's complexity tractable. We can also observe that the overall cost increases with the number of users. The reason is that, as each additional user has lower SNR, its transmission time is higher. As a consequence, EdgeBOL is forced to invest more resources (i.e., airtime, GPU speed) in the system to compensate for this degradation of mean wireless conditions.

### *E. Dynamic Scenarios*

We now test the performance of EdgeBOL in the presence of fast context dynamics and sudden constraint changes. Let us start with the former. To this end, we deploy an untrained EdgeBOL in an environment where the wireless conditions quickly vary between 5 and 38 dB, as depicted by the first

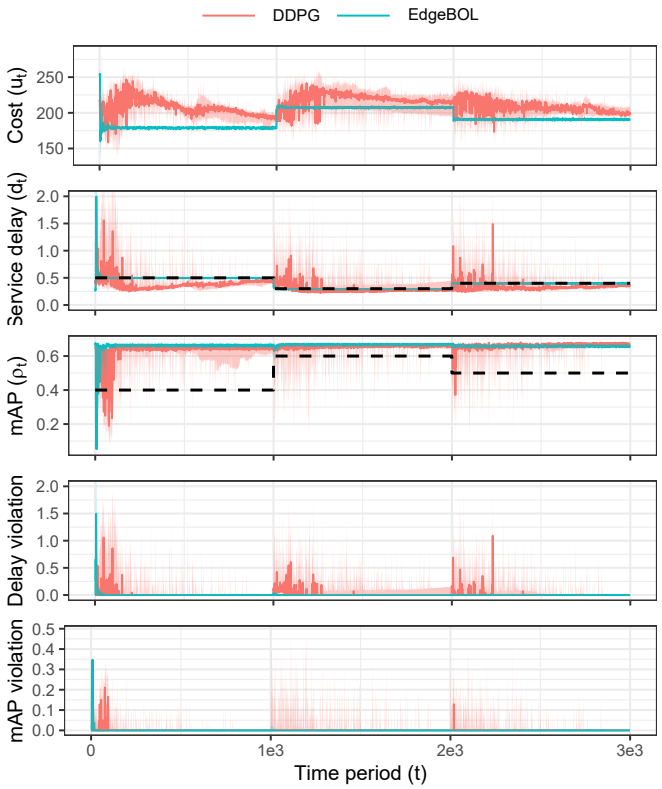


Fig. 16: Evolution of delay and mAP upon changes on the constraint settings for EdgeBOL and a DDPG approach implemented with neural networks ($\delta = 8$).

plot in Fig. 15, and set $\delta_1 = 1$ and $\delta_2 = 8$. The top part of the plot depicts the size of the safe control set over time. As expected, the safe set quickly reduces within roughly 25 time periods and then adapts to the eventual contextual changes, with fluctuations matching the context changes. Remarkably, EdgeBOL convergences upon only 3 cycles across all contexts under evaluation. This is possible because the knowledge acquired by EdgeBOL for one context *is actually transferred across similar contexts*. That is, EdgeBOL is able to select judicious policies, shown in the remaining plots of the figure, *even for unseen contexts*. Specifically, for this choice of $\delta$ parameters, the GPU speed policy and the MCS policy highly vary upon context dynamics, whereas the image resolution and the airtime policy remain consistently high. It is worth mentioning that these policy dynamics are substantially different for diverse values of $\delta$ (not shown due to space constraints).

The above illustrates one of the major advantages of our approach, which makes appropriate decisions—even for contexts unseen before—by inferring correlations between the cost function and the context-control space. Conventional neural network-based approaches are substantially less efficient in doing so, which renders EdgeBOL a particularly *data-efficient solution*.

To assess this, we implement a customized version of the deep deterministic policy gradient (DDPG) [64]. This benchmark is inspired by [12], which is the most related work to ours. Since the DDPG is designed to address the full-RL problem, we need to adapt it to address a contextual bandit problem, according to our formulation (Sec. IV). The DDPG algorithm uses an actor-critic neural network architecture, but the critic, instead of approximating the Q function (full-RL problem), learns a new cost function referred to as *DDPG cost*. The *DDPG cost* takes the value of (1) when all the constraints in (2) are satisfied, and the maximum cost value otherwise. Note that the DDPG does not handle constraints naturally and by using the *DDPG cost* function we allow the algorithm to do it. DDPG is particularly appealing for this type of problem because it operates with continued-valued control spaces. Conversely, value-based approaches (such as Deep Q-Networks) are impractical for large control spaces, such as ours [65]. We mildly modified the architecture presented in [12] with a sigmoid function for the actor's output and optimized all the hyper-parameters (such as the decay) to minimize convergence time and performance.

We then test EdgeBOL and DDPG in a dynamic scenario where the constraint settings change over time: (*i*) $d^{max} = 0.5$ s, $\rho^{min} = 0.4$ from $t = 0$ through $t = 1000$; (*ii*) $d^{max} = 0.4$ s, $\rho^{min} = 0.6$ from $t = 1000$ through $t = 2000$; and (*iii*) $d^{max} = 0.5$ s, $\rho^{min} = 0.5$ from $t = 2000$ on. Fig. 16 depicts the evolution over time of the service delay and the mAP performance for both approaches. Not surprisingly, EdgeBOL rapidly converges to policies that respect the performance constraints, even when they suddenly change. The non-parametric nature of our approach and the fact that we can compute safe control sets for any constrained setting based on prior data, allows EdgeBOL to drive the system to the new optimal points of operations almost instantaneously. In marked contrast, the neural network-based benchmark takes a substantially higher number of time periods to find the new optimal—it is actually unable to converge prior to the constraint changes—and fails to adapt graciously upon constraint changes because neural networks are parametric models that need to re-learn upon such changes.

## VII. Conclusions

The energy-aware implementation of AI services at the network edge is increasingly important for performance, economic and environmental reasons [6], [5]. Our measurements showed non-trivial trade-offs between the delay and accuracy of such services, and revealed how these metrics are shaped by the base station and edge server control policies. Using a Bayesian learning algorithm we automated the identification of a policy that minimizes the aggregate energy costs while adhering to predetermined performance criteria. Remarkably, this framework comes with minimal assumptions and is proved particularly effective in exploring the huge system configuration space. The proposed resource control mechanism is fully compliant with O-RAN and particularly promising for enabling edge AI services, as we verify experimentally using a prototype.

## Appendix

### A. Statistical Characterization of Contexts

Let us discuss the statistical characterization of the channel states of users as input context, defined in Sec. IV-B. This design decision is made in order to make EdgeBOL work better in practice. The context comprises the number of users

and the mean and variance of uplink CQIs. The straightforward approach is to use the full context, i.e., the CQI of each active user. However, this has several disadvantages. First, the problem dimension grows with the number of users. The higher the number of users, the larger the dimension of context space, and therefore the more data are needed by the algorithm to converge. And secondly, the context space dimension changes with the number of users (e.g., 4 dimensions for 3 users, 7 dimensions for 6 users, etc). To solve this, we would need to run a different instance of `EdgeBOL` for each context space dimension, increasing even more the data requirements for convergence. Our approach, instead, is to handle multiple users by aggregating statistics, and this overcomes these disadvantages with a negligible impact on performance, as we empirically validate in Sec. VI-D.

### *B. Multiple Edge Services*

In this work, we assume a pre-configured slice hosting the edge service. This allows us to decouple the impairments and couplings across different services. Nevertheless, `EdgeBOL` may be extended to jointly optimize multiple services concurrently with a few changes: (*i*) expand the context and action space for all services; (*ii*) add the KPI constraints for each service; and (*iii*) add constraints on the coupled resources (i.e., GPU and radio). Although this modeling approach seems promising performance-wise, the dimension of the extended problem endangers its practical performance. Specifically, the dimension of the context-action space becomes $4S+3$, where $S$ is the number of services; and the number of constraints raises to $2S+2$ – assuming each service requires two performance constraints. This dimension expansion increases exponentially the required amount of observation data – hence introducing convergence delays – and therefore must be employed with caution and in small-scale settings only.

In contrast, `EdgeBOL` is aimed to be a general-purpose practical solution. To that end, we focus on the case where each service is hosted by pre-configured network slices [66], [67]. This approach addresses the scalability issue mentioned above (we only have to deal with one service) and is in line with recent standardization activities of 3GPP [68]. Note that network slices can be re-configured in the timescale of hours or minutes [69]. For that reason, our solution (that operates in the timescale of seconds) does not modify the slice configuration, but instead jointly optimizes the service parameters and resource allocation within the slices, a problem that is faced by the industry today when using network slicing. Hence, we consider this solution to be a flexible approach that addresses the efficiency vs. scalability challenges at hand.

### *C. Heterogeneous Edge Services*

In this paper, we selected the object recognition service because it is among the most challenging ones to optimize (i.e., there are many involved parameters, it is very resource-intensive, etc). Moreover, it is very representative of modern AI services that rely on computer vision applications (e.g., vehicle navigation, surveillance systems, mobile health apps [70], etc.). It is important to highlight that Policy 1 (img. resolution), defined in Sec. III seems to be very specific and bound to a computer vision application. However, we can define it in a more general way as Policy 1 indicates the amount of data sent from the user to the service. This general definition can be transferable to other domains and applications. To illustrate that, let us consider another service of different nature.

Let us consider that our solution is used for predictive maintenance. In this setting, there are many vibration and audio sensors deployed in a factory that collect and send data to an anomaly detection model. When the model predicts an anomaly, contingency measures can be taken in real-time avoiding future breakdowns [71], [72]. In this example, the sensors collect continuous values that should be pre-processed (sampled, quantified, and compressed) before sending them to the AI service. We can observe the same trade-off as in our application example. Higher compression reduces the transmission delay but can worsen the accuracy of the model, and vice versa. Although we cannot directly transfer the measurement findings in Sec. III to very different applications, the fundamental trade-offs and the need for joint, data-driven, optimization of performance and energy still hold.

## REFERENCES


[1] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile Augmented Reality Survey: From Where We Are to Where We Go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.

[2] Microsoft, "Seeing AI," https://www.microsoft.com/en-us/ai/seeing-ai.

[3] Y. Taigman *et al.*, "Deepface: Closing the Gap to Human-level Performance in Face Verification," in *Proc. of IEEE CPVR*, 2014.

[4] A. Garcia-Saavedra *et al.*, "Joint optimization of edge computing architectures and radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2433–2443, 2018.

[5] N. Scientist, "Creating an ai can be five times worse for the planet than a car," 2019. [Online]. Available: https://www.newscientist.com/article/2205779-creating-an-ai-can-be-five-times-worse-for-the-planet-than-a-car/

[6] GSMA - Future Networks, "Energy efficiency: An overview," 2019. [Online]. Available: https://www.gsma.com/futurenetworks/wiki/energy-efficiency-2

[7] China Mobile Limited, "2020 Sustainability Report," white Paper, 2020.

[8] GSMA Association, "5G energy efficiencies: green is the new black," white Paper, 2020.

[9] S. Alyamkin *et al.*, "Low-power computer vision: Status, challenges, and opportunities," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 411–421, 2019.

[10] A. Goel *et al.*, "A survey of methods for low-power deep learning and computer vision," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. IEEE, 2020, pp. 1–6.

[11] I. Gomez-Miguelez *et al.*, "srslte: an open-source platform for lte evolution and experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, 2016, pp. 25–32.

[12] J. A. Ayala-Romero *et al.*, "vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs," *IEEE Transactions on Mobile Computing*, 2020.

[13] ——, "EdgeBOL: automating energy-savings for mobile edge AI," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 397–410.

[14] W. Zhang *et al.*, "Jaguar: Low Latency Mobile Augmented Reality with Flexible Tracking," in *Proc. of ACM Conference on Multimedia*, 2018.

[15] P. Jain, J. Manweiler, and R. Roy Choudhury, "OverLay: Practical Mobile Augmented Reality," in *Proc. of ACM MobiSys*, 2015.

[16] Z. He, H. Li, Z. Wang, S. Xia, and W. Zhu, "Adaptive compression for online computer vision: an edge reinforcement learning approach," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 4, pp. 1–23, 2021.

[17] G. Lu, R. Yang, S. Wang, S. Liu, and R. Timofte, "Deep learning for visual data compression," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 5683–5685.

[18] A. Galanopoulos *et al.*, "Measurement-driven analysis of an edge-assisted object recognition system," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[19] H. Li *et al.*, "JALAD: Joint Accuracy- and Latency-Aware Deep Structure Decoupling for Edge-Cloud Execution," in *Proc. of IEEE ICPADS*, 2018.

[20] X. Ran *et al.*, "DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics," in *Proc. of IEEE INFOCOM*, 2018.

[21] J. Jiang *et al.*, "Chameleon: Scalable Adaptation of Video Analytics," in *Proc. of ACM SIGCOMM*, 2018.

[22] C.-C.Hung,et al., "VideoEdge:Processing Camera Streams Using Hierarchical Clusters," in *Proc. of IEEE/ACM SEC*, 2018.

[23] Q. Liu, and T. Han, "DARE: Dynamic Adaptive Mobile Augmented Reality with Edge Computing," in *Proc. of IEEE ICNP*, 2018.

[24] P. Yang *et al.*, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Trans. on Industrial Informatics*, vol. 16, no. 7, pp. 4855–4864, 2020.

[25] Y. Li, Y. Chen, T. Lan, and G. Venkataramani, "MobiQoR: Pushing the Envelope of Mobile Edge Computing Via Quality-of-Result Optimization," in *Proc. of IEEE ICDCS*, 2017.

[26] D. Bega *et al.*, "CARES: Computation-aware Scheduling in Virtualized Radio Access Networks," *IEEE Trans. on Wireless Communications*, vol. 17, no. 12, pp. 7993–8006, 2018.

[27] ——, "DeepCog: Optimizing Resource Provisioning in Network Slicing With AI-Based Capacity Forecasting," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 361–376, 2020.

[28] D. Raca *et al.*, "On Leveraging Machine and Deep Learning for Throughput Prediction in Cellular Networks: Design, Performance, and Challenges," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 11–17, 2020.

[29] J. A. Ayala-Romero *et al.*, "Online learning for energy saving and interference coordination in hetnets," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1374–1388, 2019.

[30] J. J. Alcaraz *et al.*, "Online reinforcement learning for adaptive interference coordination," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 10, p. e4087, 2020.

[31] F. Mismar, J. Choi, and B. L. Evans, "A Framework for Automated Cellular Network Tuning With Reinforcement Learning," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7152–7167, 2019.

[32] Z. Zhang, L. Ma, K. Poularakis, K. K. Leung, and W. Lingfei, "DQ Scheduler: Deep Reinforcement Learning Based Controller Synchronization in Distributed SDN," in *Proceedings of IEEE ICC*, 2019.

[33] G. Garcia-Aviles *et al.*, "Nuberu: Reliable ran virtualization in shared platforms," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 749–761.

[34] A. Bastian *et al.*, "CBA: Contextual Quality Adaptation for Adaptive Bitrate Video Streaming," in *Proc. of IEEE INFOCOM*, 2019.

[35] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Orchestrating Energy-Efficient vRANs: Bayesian Learning and Experimental Results," *IEEE Transactions on Mobile Computing*, vol. 22, no. 5, pp. 2910–2924, 2023.

[36] J. Chuai, Z. Chen, G. Liu, X. Guo, X. Wang, X. Liu, C. Zhu, and F. Shen, "A collaborative learning based approach for parameter configuration of cellular networks," in *Proceedings of IEEE INFOCOM*, 2019.

[37] M. Hashemi *et al.*, "Efficient Beam Alignment in Millimeter Wave Systems Using Contextual Bandits," in *Proc. of IEEE INFOCOM*, 2018.

[38] A. Krause and C. S. Ong, "Contextual gaussian process bandit optimization," in *Advances in neural information processing systems*, 2011, pp. 2447–2455.

[39] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[40] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," 2015.

[41] A. Garcia-Saavedra and X. Costa-Perez, "O-ran: Disrupting the virtualized ran ecosystem," *IEEE Communications Standards Magazine*, 2021.

[42] M. Everingham *et al.*, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[43] L. Diez *et al.*, "Lasr: A supple multi-connectivity scheduler for multi-rat ofdma systems," *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 624–639, 2020.

[44] C. Marquez *et al.*, "How should i slice my network? a multi-service empirical evaluation of resource sharing efficiency," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 191–206.

[45] J. Mendes *et al.*, "Cellular access multi-tenancy through small-cell virtualization and common rf front-end sharing," *Computer Communications*, vol. 133, pp. 59–66, 2019.

[46] O-RAN Alliance, "O-RAN-WG1-O-RAN Architecture Description (O-RAN.WG1.O-RAN-Architecture-Description-v04.00)," Technical Specification, March 2021.

[47] ——, "O-RAN Non-RT RIC: Functional Architecture 1.01 - March 2021 (O-RAN.WG2.Non-RT-RIC-ARCH-TR-v01.01)," Technical Specification, March 2021.

[48] ——, "O-RAN Working Group 2 (Non-RT RIC and A1 interface WG) A1 interface: General Aspects and Principles (O-RAN.WG2.A1GAP-v02.02 )," Technical Specification, March 2021.

[49] ——, "O-RAN Near-Real-time RAN Intelligent Controller Architecture & E2 General Aspects and Principles 1.01 (O-RAN.WG3.E2GAP-v01.01)," Technical Specification, July 2020.

[50] A. Galanopoulos, J. A. Ayala-Romero, G. Iosifidis, and D. Leith, "Bayesian online learning for mec object recognition systems," in *2020 IEEE Global Communications Conference (Globecom)*. IEEE, 2020.

[51] Y. Inoue *et al.*, "Real-time frame-rate control for energy-efficient on-line object tracking," *IEICE Trans. on Fundamentals of Electronics, Comms. and Computer Sciences*, vol. 101, no. 12, pp. 2297–2307, 2018.

[52] Y. Zhao *et al.*, "What factors influence the mobile health service adoption? a meta-analysis and the moderating role of age," *International Journal of Information Management*, vol. 43, pp. 342–350, 2018.

[53] A. H. Sodhro *et al.*, "Mobile edge computing based qos optimization in medical healthcare applications," *International Journal of Information Management*, vol. 45, pp. 308–318, 2019.

[54] L. Li *et al.*, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 661–670.

[55] S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári, "Parametric bandits: The generalized linear case," in *Advances in Neural Information Processing Systems*, 2010, pp. 586–594.

[56] D. Duvenaud, "Automatic model construction with gaussian processes," Ph.D. dissertation, University of Cambridge, 2014.

[57] A. D. Bull, "Convergence Rates of Efficient Global Optimization Algorithms," *Journal of Machine Learn. Res.*, vol. 12, pp. 2879–2904, 2011.

[58] Y. Sui *et al.*, "Stagewise safe bayesian optimization with gaussian processes," *arXiv preprint arXiv:1806.07555*, 2018.

[59] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics," *Machine Learning*, pp. 1–35, 2021.

[60] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with gaussian processes," in *International Conference on Machine Learning*. PMLR, 2015, pp. 997–1005.

[61] M. Fiducioso, S. Curi, B. Schumacher, M. Gwerder, and A. Krause, "Safe Contextual Bayesian Optimization for Sustainable Room Temperature PID Control Tuning," in *IJCAI-19*, 7 2019, pp. 5850–5856.

[62] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[63] S. Ren *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[64] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. of ICLR 2016*, San Juan, Puerto Rico, May 2016.

[65] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," 2016.

[66] L. Zanzi *et al.*, "Laco: A latency-driven network slicing orchestration in beyond-5g networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 667–682, 2021.

[67] ——, "Ovnes: Demonstrating 5g network slicing overbooking on real deployments," in *IEEE INFOCOM 2018 Workshops*. IEEE, 2018.

[68] GSMA, "Network Slicing: Use case requirerments," Technical Specification, April 2018. [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2020/01/2.0_Network-Slicing-Use-Case-Requirements-1.pdf

[69] IBM, "IBM Cloud Pak for Network Automation. Zero-touch network operations with AI-powered automation," Technical Specification, 2021. [Online]. Available: https://www.ibm.com/downloads/cas/PALQDWZ4

[70] J. D. Akkara *et al.*, "Smartphone apps for visually impaired persons," *Kerala Journal of Ophthalmology*, vol. 31, no. 3, p. 242, 2019.

[71] K. Wang and Y. Wang, "How ai affects the future predictive maintenance: a primer of deep learning," in *International Workshop of Advanced Manufacturing and Automation*. Springer, 2017, pp. 1–9.

[72] W. Shin, J. Han, and W. Rhee, "Ai-assistance for predictive maintenance of renewable energy systems," *Energy*, vol. 221, p. 119775, 2021.