EdgeBOL: Automating Energy-savings for Mobile Edge AI

Jose A. Ayala-Romero
Huawei Ireland Research Center

Andres Garcia-Saavedra
NEC Laboratories Europe

Xavier Costa-Perez
i2cat, ICREA and NEC Laboratories Europe

George Iosifidis
Delft University of Technology

ABSTRACT

Supporting Edge AI services is one of the most exciting features of future mobile networks. These services involve the collection and processing of voluminous data streams, right at the network edge, so as to offer real-time and accurate inferences to users. However, their widespread deployment is hampered by the energy cost they induce to the network. To overcome this obstacle, 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 desirable accuracy and latency thresholds. Using a fully-fledged prototype with a software-defined base station (BS) and a GPU-enabled edge server, we profile a state-of-the-art video analytics AI service and identify new performance trade-offs. Accordingly, we tailor the optimization framework to account for the network context, the user needs, and the service metrics. The efficacy of our proposal is verified in a series of experiments and comparisons with neural network-based benchmarks.

CCS CONCEPTS

• **Networks → Mobile networks**; **Network management**.

KEYWORDS

Mobile networks, O-RAN, energy efficiency, QoS

ACM Reference Format:
Jose A. Ayala-Romero, Andres Garcia-Saavedra, Xavier Costa-Perez, and George Iosifidis. 2021. EdgeBOL: Automating Energy-savings for Mobile Edge AI. In *The 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '21), December 7–10, 2021, Virtual Event, Germany.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3485983.3494849

1 INTRODUCTION

There is growing consensus that the next generation of mobile networks need to support AI services *at the edge*. This involves the collection, transfer and processing of data flows, with the aim to provide real-time inferences to end users ranging from handheld and small IoT devices to moving nodes (e.g. drones). Representative examples are mobile video analytics (MVA), which are used in AR/VR services [15], cognitive assistance applications [43], surveillance systems [68], and other similar AI services. The core task of MVA services 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 [25].

These services are equally exciting for the users as they are challenging to implement. Namely, in MVA services the network's role is not confined to transferring data from one point to another, nor it suffices to support in-network processing of the data streams. Instead, the network needs to directly optimize the AI 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 AI services create voluminous data flows, involve heavy computations and consume large amounts of energy [59]. In fact, energy consumption is not only one of the most prevalent operating expenditures for mobile networks [45][1] but has been also identified as a risk hampering the penetration of the inherently energy-demanding AI-based mobile services. Besides, energy is the common physical resource that all pertinent network operations, namely computations and transmissions, consume and its prudent management is imperative also from a performance point-of-view.

Clearly, in order to realize Edge AI services we need to devise a systematic methodology for energy-aware control of the communication and computing resources of the network. *Hence, the goal of this paper is to design and evaluate experimentally an orchestration framework that manages the resources of base stations and edge MVA-based AI services following two joint optimization goals: (i) minimize the energy toll associated with the service and the mobile network operation, and (ii) meet service performance indicator targets.*

To shed light on this problem, we have built a fully-fledged prototype system with a software-defined base station (BS) (using srsRAN suite [26]) and a GPU-enabled edge server that offers a MVA service to mobile users. We measure the joint impact that resource control policies at the user device (video frame resolution), the BS (radio configuration) and the server (GPU speed) have on the service accuracy and on end-to-end latency (quality of service performance indicators), and on power consumption (cost).

Our experiments show that, unlike other services, performance is highly volatile and depends on the underlying hardware, the AI service configuration, and even the actual user data. Furthermore, these services include a wide range of configuration options, e.g., selecting different architectures of Neural Networks, different processing equipment, or adjusting the data sources. All these parameters affect in an unknown way the latency and accuracy performance, which in turn renders traditional network resource orchestration techniques ineffective for this problem.

[1]For instance, China Mobile committed to reduce the overall energy consumption per unit of telecom business by no less than 6% in 2021 [12], and Verizon and Vodafone have set targets to reach net zero energy emissions by 2040 [28].

In order to overcome these challenges, we propose an optimization framework for orchestrating jointly these decisions that is oblivious to the underlying hardware and user data streams. This robustness is achieved by using a non-parametric *Bayesian online learning* algorithm, named `EdgeBOL` (Edge Bayesian Online Learning), that deals with different system states and performance constraints. We formulate a problem for minimizing 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 [4]. Our solution shows an unprecedented convergence speed and satisfies the service (stochastic) constraints with very high probability.

To summarize, the key contributions of this paper are:

- We propose the new problem of jointly controlling RAN resources and edge AI 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 AI services.
- Finally, we evaluate `EdgeBOL` using our experimental platform and actual datasets, and compare it with a SoA neural network-based reinforcement learning algorithms.

Paper Organization. §2 discusses related works; §3 presents motivating experimental findings; in §4 we formulate the orchestration problem and in §5 introduce the learning algorithm which is evaluated in §6. We conclude in §7.

2 RELATED WORK

There are recent studies about the deployment of AI services, in particular video analytics, at the network edge. Some of them aim to improve the performance using, e.g., adaptive encoding, caching or visual tracking [31, 39, 73]. Other studies measure the trade-off between accuracy and latency, which is key for real-time AI services [34, 55]. Regarding the resource orchestration in these systems, [32] and [10] search greedily in real-time for the most resource-prudent configuration; [53] and [69] allocate computing resources and decide the image compression (or, video quality) and neural network model; and [36] attempts to minimize the energy consumption and service latency. In turn, Nuberu [23] presents a reliable virtualized base station design. These important experimental studies, however, do not account for the coupling of the mobile network with the edge servers. Our experiments reveal that their joint orchestration is imperative for these demanding AI services.

Orchestrating the resources of mobile networks for traditional systems is a well-studied problem. For instance, [6], [65] select the modulation and coding scheme (MCS) and airtime to maximize throughput, using predetermined models for the operation functions; yet, these models are platform-dependent. On the other hand, model-free approaches [72] are more versatile and have been used, e.g., for slicing [7], throughput forecasting [54], and energy cost reduction [3]; but suffer from the lack of (accurate) training data. Reinforcement learning (RL) is yet another option and has been used for interference coordination [1], network diagnostics [44], or SDN control optimization [74]. These works have been derived for legacy base stations, and without accounting for the performance requirements of AI services such as the accuracy of inferences.

Our approach for controlling the virtualized Base Station (vBS) and the edge server is fundamentally different from the above works as it relies on Bayesian contextual bandit algorithms. Such approaches have been employed to adjust video streaming in mobile networks [5], to optimize BS handovers [13], and to control mmWave networks [29]. Perhaps the most closely related work to ours is [4], 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 [33] that combines bandit exploration with Gaussian Processes (GPs) [66]. Thus, we use the uncertainty in the estimations provided by the GPs to efficiently explore new configurations and speed up convergence. We extend here this approach by including the AI service-related context, and to consider the vBS and edge server power consumption. These ideas have been proven very successful for automated machine learning problems [60], and we show here that they hold great promise for automated resource orchestration in mobile networks.

3 EXPERIMENTAL ANALYSIS

We have performed an exhaustive set of experiments using the testbed described in detail in §6.1. 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 [38]. 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 all the measurements shown in this section is available online[2] to enable reproducibility and to facilitate further research in this area. Motivated by O-RAN specifications [24], we focus on configuration *policies* set by an orchestrator that operates at second-level timescale (Non-Real-Time RAN Intelligent Controller in O-RAN). These policies are rules that must be respected by lower-level controllers that operate at millisecond-level timescale (or faster) — and which are orthogonal (and beyond the scope) of our study. In the following, we analyze the trade-offs between different *configuration policies* and *performance indicators* that are relevant to the system *stakeholders*: (*i*) quality of service experienced by the end-users, (*ii*) the energy associated with the service provider, and (*iii*) the energy cost associated with the mobile network operator.

Precision, delay, and image resolution. We start off by analyzing two metrics of interest for the user's quality of service: the service's performance to recognize objects and the service delay, formally introduced in Performance Indicator 1 and 2, respectively.

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

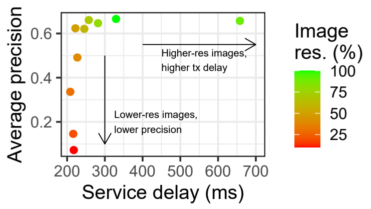

Figure 1: Mean average precision (mAP) *vs.* service delay for images with different resolutions.

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's precision is estimated by the Mean Average Precision (mAP), a popular metric in computer vision for object recognition applications [19]. 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, which in our case is set to 0.5, trigger a true positive. Then, given a set of images, the Average Precision (AP) corresponds to the area below the precision-recall curve. Finally, the mAP is calculated as the mean AP over all object categories, hence ranges from 0 (worst performance) to 1 (best performance).*

According to our measurements, the most relevant feature that affects the mAP is the *image resolution*, defined in Policy 1.

Policy 1 (Image resolution). *This policy sets the average encoding of every image (number of pixels) generated by the users, which can be enforced by the service. In our experiments, the maximum (100%) resolution is 640x480 pixels. Note that, at any give time instance, the resolution of one image, set by the service application, may be larger or smaller than the policy as long as the average across the whole period and users respects the policy threshold.*

We illustrate this in Fig. 1, which shows the trade-off between service delay and mAP for the COCO images dataset encoded with different resolutions. The remaining configuration policies (described later) are fixed so service delay is minimum. The results are rather intuitive: (*i*) Higher-resolution images carry more pixels encoded in a larger amount of data. Therefore, higher-resolution images incur 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 useful information for the object detection engine. Specifically, in our experiments, a 72% improvement in service delay is associated with a reduction of precision that ranges between 10% to 50%.

Delay, energy consumption, and radio policies. There also exists a trade-off, which naturally appears in many resource control problems [16], between the quality of service and the associated energy cost to the provider of such service. To explore this trade-off, we introduce a policy that governs the allocation of radio resources, defined as Policy 2, and an additional metric that assesses part of the aforementioned cost: the server's power consumption, defined as Performance Indicator 3.

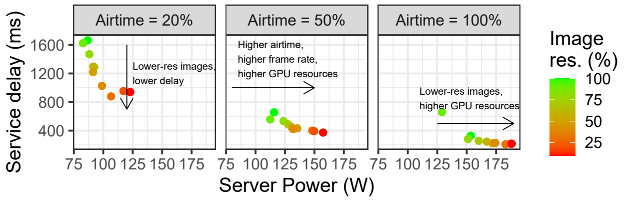

Figure 2: Service delay *vs.* server's power consumption for images with different resolutions and radio policies.

Policy 2 (Radio Airtime). *This radio policy imposes a constraint on the amount of radio resources (duty cycle) the BS allocates to the service's traffic (i.e., for all users). The MAC layer radio scheduler, which operates at millisecond-level granularity then must allocate radio resources (which may be different across users depending on their instantaneous channel quality) such that the threshold set by the policy is respected. Due to the nature of this AI service, we focus on uplink communication.*

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

Fig. 2 depicts the service delay *vs* the server power consumption, for different airtime radio policies and image resolutions. Similarly 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 selected radio policy has an important impact on service delay as well. This is rather expected since lower airtime implies lower usage of radio resources, which further increase the transmission time of the requests at the radio interface. Specifically, our experiments show that an 80% increase of 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 for the service provider. Specifically, there is a 56% increase in power consumption for an 80% increase in radio time resource; a similar increase attained when there is a 75% increase in image resolution. This is due to the fact that increasing the radio resources allow the user to send a higher rate of requests in a similar way than low-res images do, which ultimately increase the workload assigned to the service's resources (the GPU in this case).

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

Policy 3 (GPU speed). *The server's policy is a GPU power 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. 3 (top) depicts the service delay and the server's power consumption

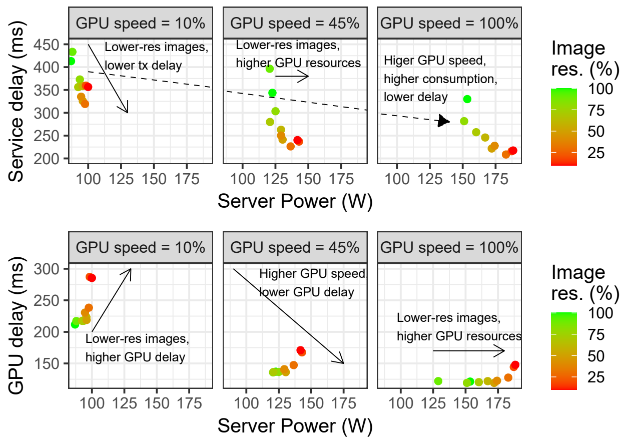

Figure 3: Delay *vs.* server's power consumption for images with different resolutions and GPU policies.

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. 3 (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 (a cost to the service provider), certainly appears for other user-related metrics, such as the mAP introduced earlier. To assess this, Fig. 4 shows the mAP achieved by the service as a function of the server's power consumption for a variable set of image resolutions and the largest amount of radio and computing resources to minimize delay. The figure confirms that the service provider's cost is also dependant on the mAP performance. Importantly, however, the relationship with the mAP is substantially different to that with the service delay. In this case, higher mAP performance actually requires *less* power consumption from the service provider. The reason lies upon the fact that higher-res images (which render higher mAP) facilitate the object detection task and hence require less computing resources as shown in Fig. 3 (bottom).

BS power consumption, radio policies, and image resolution. Finally, the costs associated with the network operator (our third stakeholder) 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 [4] 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.

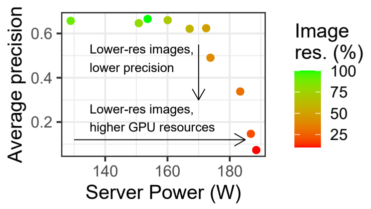

Figure 4: Mean average precision *vs.* server's power consumption for images with different resolutions.

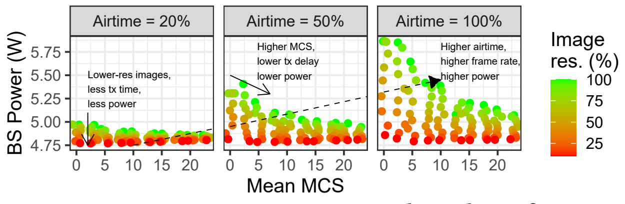

Figure 5: BS power consumption *vs.* radio policies for images with different resolutions.

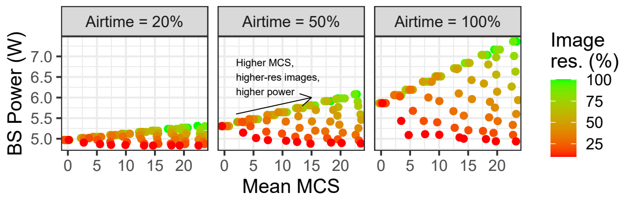

Figure 6: BS power consumption *vs.* radio policies for images with different resolutions and 10x higher load.

Policy 4 (Radio MCS). *This policy imposes a constraint on the maximum modulation and coding scheme (MCS) eligible by the BS to transport the service's data over the air. Like before, the MCS selected by the MAC layer of the BS may be lower than this bound for individual 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. 5 the power consumption measured at the baseband unit of the BS for a variable setting of airtime and MCS radio policies, and different image resolutions. We first observe that lower-resolution images consume less amount of radio resources and therefore have a smaller footprint over the BS' power consumption. Second, a higher investment on radio resources (airtime) actually induces higher power consumption because it allows the user to issue a higher rate of service requests (images). Finally, perhaps surprisingly, higher MCS policies actually cause *lower* BS power consumption. The reason is that the data load at the BS is relatively low compared to the amount of bandwidth available at the BS, 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 power consumption over the long run.

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 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. 6. 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 highly-coupled parameters with non-trivial relationship with the performance and energy cost. As a consequence, we resort to model-free contextual bandit methods, an area of machine learning applied in many control problems, to design a controller that adapts autonomously to context changes and the underlying platform.

4 SYSTEM AND PROBLEM FORMULATION

4.1 System

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 for this service is created including virtualized Base Station (vBS) and the edge server [40, 42]. This is illustrated by the orange boxes in Fig. 7. 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 approprite 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 [51], a carrier-led alliance of operators and manufacturers to build open and intelligent RAN solutions [24].

As shown in Fig. 7, 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 (O-eNBs/O-gNBs):

- An rApp (within O-RAN's non-RT RIC), as defined in [50], interacts with the learning agent and handles O-RAN's A1 interface (specifically, the A1's Policy Management Service) as specified in [48, 49, 52] 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 [51]), and O-eNB in case of 4G (see Section 4.3.7 in [51]).
- The E2 interface, defined in [47], 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 base station, which in our case consists of samples of the BS power consumption, and forwards it to the learning agent.

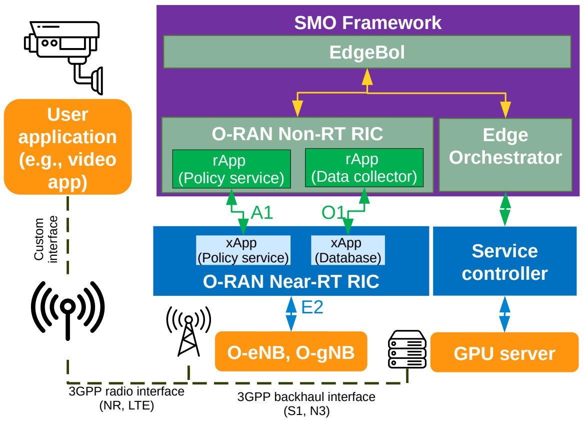

Figure 7: O-RAN compliant system architecture.

We assume that both the O-eNB/O-gNB and the GPU server can implement the configured policies (through radio scheduling at the MAC layer for the former, through a driver such as NVIDIA's for the latter). This framework allows us to use machine learning to solve our problem. Namely, we formulate the problem as a contextual multi-armed bandit or *contextual bandit*.

4.2 Problem Formulation

Let us formalize in the following the context space, action space, and the performance indicators in our system.

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 channel quality indicator (CQI) across all users in the slice during the previous period, and C is the context space.

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; Γ the possible GPU speed configurations; and $\mathcal{M}$ the set of all possible MCS policies 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 [51], and the image resolution is indicated to the user using the application of the service. 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.

Performance indicators. Similarly, our performance indicators were introduced in §3. 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 let $d(c, x) := \max_i D_i(c, x)$ and $\rho(c, x) := \min_i Q_i(c, x)$ denote 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 $p^b(c, x)$. Note that in practice the observations of the performance indicators are noisy (even in static setups) since the system is stochastic in nature. Remarkably, our solution intrinsically deals with noisy observation as we detail in the next section. Henceforth, we denote by $d_t(c_t, x_t)$,

$\rho_t(c_t, x_t)$, $p_t^s(c_t, x_t)$, and $p_t^b(c_t, x_t)$ 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 the LA 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 [22].

4.3 Online Learning 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 [11, 12, 28]. Hence, our goal is to minimize the power consumption of the whole system (vBS and edge server) subject to 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 to 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 δ_1 and δ_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 service-level, going a step beyond other works considering lower-level performance requirements (e.g., [4]) 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 §3. 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 (number of images per second) that the user is going to process, and (*ii*) a minimum mAP denoted by ρ^{min} which indicates a lower bound on how accurate is the service in detecting the objects. We formulate the problem as follows:

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

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 $\rho(c, x) := \min_i Q_i(c, x)$. The nature of this problem calls for a *contextual bandit* formulation, i.e., we need to select a control policy x_t every time period t, given a context c_t. Indeed, the optimal decision at each t only depends on the current context c_t, and the performance indicators are observed at the end of each time period. Our objective is to minimize the cost in the long-term while satisfying the constraint at each time period. We would like to remark that other alternative formulations can be considered. For instance, we could consider power-constrained vBSs or a edge computing power budget by including the power consumption targets as constrains, while minimizing latency and maximizing accuracy. The flexibility of our framework allows us to implement any of these different formulations with minimal changes.

4.4 Practical Considerations & Design decisions

Let us discuss some decisions made in the design of EdgeBOL to make it work in practice. EdgeBOL uses a statistical characterization of the channel state of the users as an input context. That is, the context comprises the number of users and the mean and the variance of the uplink CQI, as defined in §4.2. The straightforward approach is to use the full channel information, i.e., the CQI of each active UEs. However, this has several disadvantages. First, the complexity of the problem grows with the number of users. The higher the number of users, the larger the dimensionality of the context space, and therefore the higher the amount of data needed by the algorithm to converge (curse of the dimensionality). And second, the dimensionality of the context space 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 dimensionality, increasing even more the data requirements for convergence. The approach considered in EdgeBOL to handle multiple users by aggregating statistics overcomes these disadvantages with a negligible impact on performance, as validated empirically in §6.4.

Furthermore, we have assumed above a pre-configure slice hosting the service. Nevertheless, EdgeBOL may be extended to jointly optimize multiple AI services concurrently with minor changes: (*i*) expand the context and action spaces for all the AI services; (*ii*) add the new KPI constraints for each service; and (*iii*) add additional constraints on the coupled resources (i.e., GPU and radio). Although this solution seems promising from an optimality point of view, the dimension of this extended problem challenges its practicality. Specifically, the dimensionality of the context-action space becomes $4S + 3$, where S is the number of AI services; and the number of constraints raises to $2S + 2$ – assuming each service requires two performance constraints. When the dimensionality of the problem increases, the amount of learning data needed grows exponentially with the problem's dimension. Thus, although it might be a feasible approach in some carefully-crafted scenarios, it becomes intractable in real-life large-scale deployments (i.e., will not converge in a reasonable amount of time). This is a well-known problem in machine learning called the curse of dimensionality [63].

In contrast, EdgeBOL is designed as a practical solution for most systems. For that purpose, we focus on the case where each AI service is hosted by pre-configured network slices [70, 71]. This approach addresses the scalability problem mentioned above (we only have to deal with one service) and is in line with the trend followed by the industry today, which is realizing network slicing to build virtual networks optimized for specific services and has been standardized by 3GPP [27, 41, 64]. Note that network slices can be re-configured in the timescale of hours or minute [14, 30, 46]. For that reason, our solution (that operates in the timescale of seconds) does not modify the slice configuration but jointly optimizes service parameters and resource allocations within the slices, a problem that is faced by the industry today when using network slicing. Hence, we consider our proposed solution a flexible approach to successfully address the efficiency vs. scalability challenges just described.

5 BAYESIAN ONLINE LEARNING

EdgeBOL is an online learning algorithm that solves the contextual bandit problem defined in §4.3. Most of the existing contextual bandit algorithms assume a linear relationship between the contexts-control space and the associated reward [35, 57], or a certain reward function structure [21, 58]. However, as the measurements in §3 revealed, our performance metrics have a non-linear and almost arbitrary structure, though with high correlation with our control policies. That is, a small change in one of the policies (e.g., image resolution) will produce a small change in the service delay and the consumed power. This allows us to get information about unobserved context-control points by observing nearby points, hence reducing the exploration time.

Based on these observations, we propose a Bayesian online learning method that models the cost and constraints 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 [66]. Let $z \in \mathcal{Z} = C \times \mathcal{X}$ 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 $\sim N(0, \zeta_{(i)}^2)$. The posterior distribution of these functions follows a GP distribution with mean $\mu_T^{(i)}(z)$ and covariance $k_T^{(i)}(z, z')$:

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

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

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 function shapes the GP's prior and posterior distributions having an impact on the algorithm's performance. It encodes the correlation of the function values for every pair of context-control points. That is, the kernel characterizes the *smoothness* of the functions [18].

The properties of the kernel function should be thoroughly selected for each specific application and the underlying functions that have to be learned. Thus, we use the experimental data analyzed in Sec. 3 to conclude that the kernel function should satisfy two properties for all the functions: *stationarity* and *anisotropicity*. This means that the kernel $k(z, z')$ is invariant to translations in $\mathcal{Z}$ but not invariant to rotations in $\mathcal{Z}$. The smoothness of the kernel 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 indicates the number of dimensions of $\mathcal{Z}$. The distance between two points based on the length-scale vector is given by

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

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 [66]. Moreover, following standard practice, we particularize it with parameter $\nu = \frac{3}{2}$ (details in [66]), 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 the functions (cost and constraints), their hyperparameters will vary depending on its shape. In fact, the hyperparameters $\mathcal{L}^{(i)}$ 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 shall remain constant. This is because when the hyperparameters are optimized using newly acquired data, it is not guaranteed that the GP's confidence interval will cover the true function within, causing the optimization to fall into poor local optima [9].

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 with 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:

$$S(c) = \{x \in \mathcal{X} \mid d(c, x) \le d^{max} \wedge \rho(x) \ge \rho^{min}\} \tag{7}$$

Nevertheless, the computation of the safe set is very challenging for several reasons. Firstly, the observations of the performance indicators are noisy due to the stochastic nature of the system, as we observed in §3. And secondly, the number of available controls $|\mathcal{X}|$ is usually very large in practice, making it unfeasible to explore all controls for all possible contexts. For that reason, we use the

GPs to compute an estimation of the safe set:

$$S_t = \big\{x \in \mathcal{X} \mid \mu_{t-1}^{(1)}(c_t,x) + \beta\sigma_{t-1}^{(1)}(c_t,x) \le d^{max} \wedge \mu_{t-1}^{(2)}(c_t,x) - \beta\sigma_{t-1}^{(2)}(c_t,x) \ge \rho^{min}\big\} \tag{8}$$

where $\left(\sigma_t^{(i)}(z)\right)^2 = k_t^{(i)}(z,z)$ (eq. (4)) and β is a weighting parameter. Note that at each time period t the point z_t is observed and the vectors Z_t and $y_t^{(i)} \forall i$ are updated consequently. Due to their correlation, the posterior distribution of points near z_t will vary having an impact on the controls that will be included in the safe set in $t+1$.

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 [2, 8, 61, 62], but they do not consider contexts. To the best of our knowledge, SafeOpt [8] is the only work considering contexts. Unfortunately, although SafeOpt does provide theoretical performance guarantees, we found in our experiments that its acquisition function has overly slow convergence. This issue has been reported in other works as well, e.g., [20]. Instead, we use the contextual Lower Confidence Bound (LCB) proposed in [33] as an acquisition function, but *constrained to the safe set*:

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

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 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[3].

Note that EdgeBOL does not expand explicitly the safe set like in other works such as [8, 62]. 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

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: $S_t = S_0 \cup \{x \in \mathcal{X} | \mu_{t-1}^{(1)}(c_t,x) + \beta\sigma_{t-1}^{(1)}(c_t,x) \le d^{max} \wedge \mu_{t-1}^{(2)}(c_t,x) - \beta\sigma_{t-1}^{(2)}(c_t,x) \ge \rho^{min}\}$
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
```

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 §6.

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 on 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.

6 EXPERIMENTAL EVALUATION

6.1 Experimental Setup

Our prototype consists of a vBS, a user equipment (UE), a digital power meter, and an edge server, Fig. 8. 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 [26] (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

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

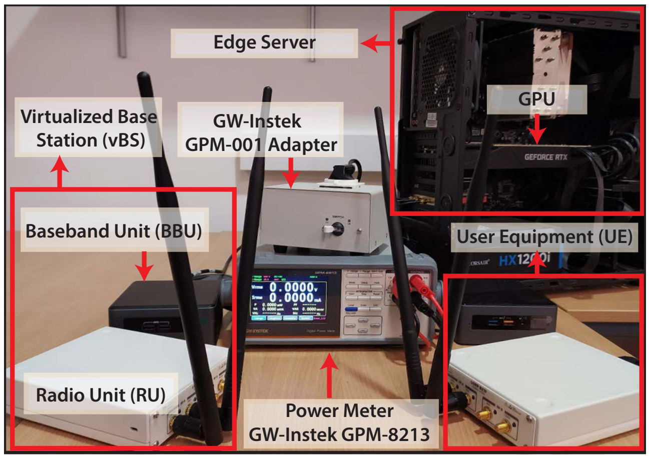
Edge Server
GPU
Virtualized Base Station (vBS)
GW-Instek GPM-001 Adapter
Baseband Unit (BBU)
User Equipment (UE)
Radio Unit (RU)
Power Meter GW-Instek GPM-8213

Figure 8: Experimental testbed.

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 [67], developed by Facebook, which performs object recognition. Specifically, Detectron2 is configured with a region-based convolutional neural network (Faster R-CNN) [56] 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 §3 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 §3. Secondly, we integrated the O-RAN E2 interface as defined in [47] 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 §4 and as defined in [48–50, 52]. 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 §6.4), we adopt simple controllers (e.g. MAC layer scheduling) that are detailed where relevant. In line with previous works [8, 20], we select $\beta^{1/2} = 2.5$, which shows good performance in our evaluations. Finally, unless otherwise stated, we will plot our results with lines and shadowed areas representing, respectively, the median value and the 10th and 90th percentiles, across 10 independent repetitions.

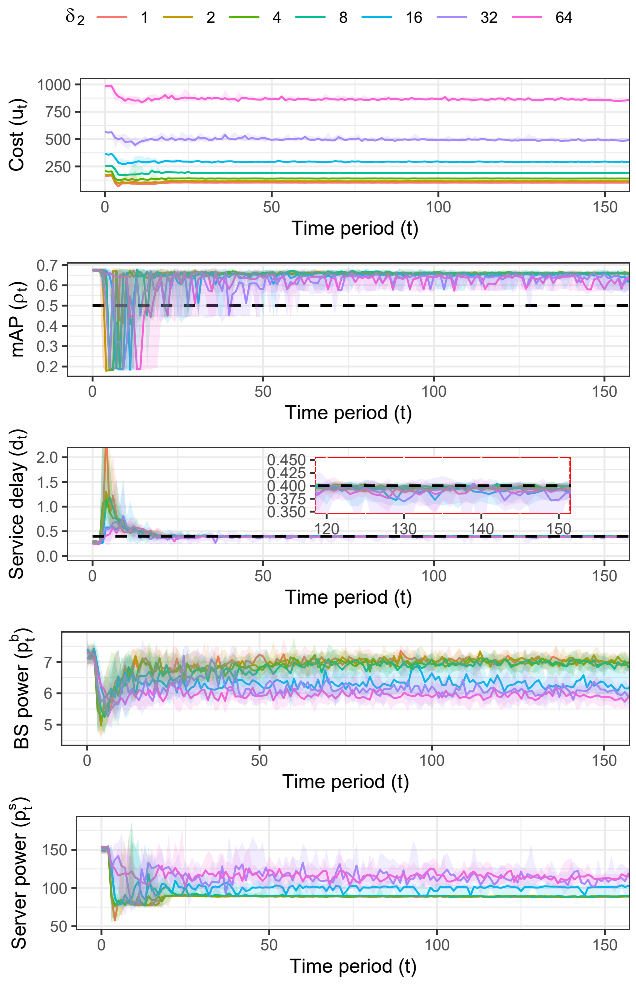

Figure 9: Convergence evaluation. A scenario with steady channel conditions (no context changes), $\delta_1 = 1$ mu/W, $\rho^{min} = 0.5$, and $d^{max} 0.4$ s.

6.2 Convergence

To evaluate the convergence of EdgeBOL, we consider a single context and a certain constraint set with ρ^{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. 9 plots the evolution over time of the cost (u_t), mAP performance (ρ_t), delay (d_t), and server and BS power consumption (p_t^s and p_t^b) as a function of δ_2.

The first observation is that the cost u_t (top plot) converges within roughly 25 time periods across all $\delta_2 = \{1, 2, 4, 8, 32\}$. Higher δ_2 values induce higher cost as the *price* associated to each watt consumed by the BS grows. Remarkably, both the mAP performance and delay fall within the selected system constraints upon convergence *with high probability*. In fact, we have observed consistent results (converge speed, satisfaction of system constraints) irrespectively of the context and system constraints.

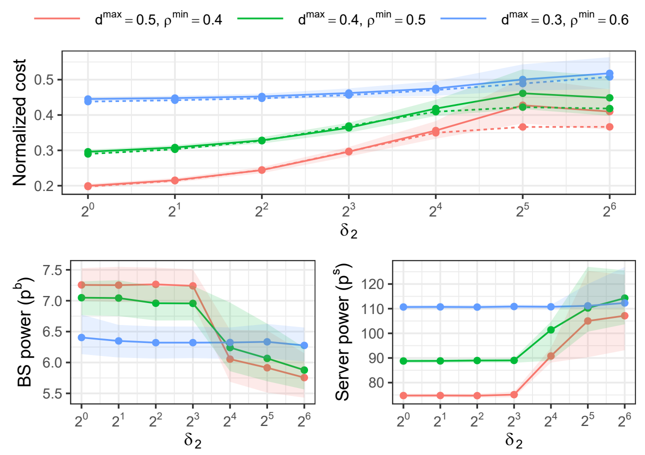

Figure 10: Power consumption and normalized cost for a single context as a function of δ_2, with $\delta_1 = 1$ mu/W. Dashed lines represent our exhaustive search approach.

The system power consumption presents interesting trade-offs with δ_2. In particular, small δ_2 (e.g., $\delta_2 = 1$) induce high power consumption at the BS but low at the server. This is because the maximum net power consumption of *our* virtualized BS (around 7.25 W) is much smaller to that of the server (between 85 to 180 W). Therefore, if the associated cost (mu/W) is similar for both BS and server, EdgeBOL will minimize the power consumption of the server at the expense of a small energy toll for the BS. However, when δ_2 is relatively high (e.g., $\delta_2 = 64$), the actual cost associated with the energy footprint of the BS becomes comparable, or even higher, than that of the server. Hence, EdgeBOL drives the system to configurations that minimize BS power consumption at the expense of the server energy. This latter case is relevant for situations when a small cell has a stringent power budget (e.g., solar-powered) or has cooling restrictions. Indeed, different types of BS have different energy footprint, which motivates the need for approaches that *learn* the relationship between power consumption patterns, performance metrics and configuration policies.

6.3 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 δ_2. Fig. 10 shows the power consumption and *normalized* cost once EdgeBOL has converged for $\delta_1 = 1$ and $\delta_2 = \{1, 2, 4, 8, 16, 32, 64\}$ mu/W. We compute the normalized cost independently for each δ_2 so we can compare across different δ_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 constraints settings (different colors in the plots), we can make two observations. First, we can confirm our earlier observation that higher values of δ_2 (compared to δ_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.

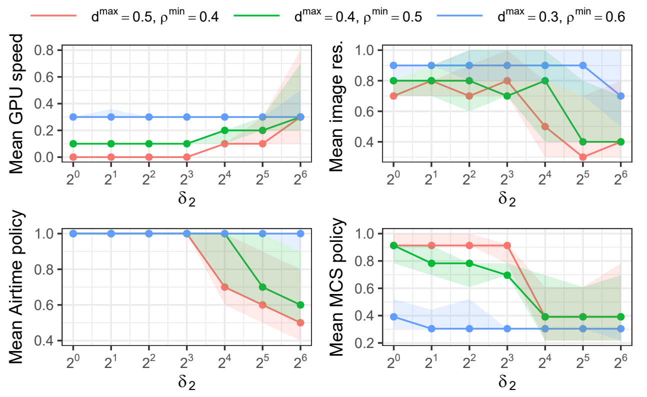

Figure 11: Policies for a single context as a function of δ_2, with $\delta_1 = 1$ mu/W.

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 δ_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 δ_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 to that of the server (between 50 and 200 W). As a result, when we increase δ_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. 11 shows the corresponding control policies for the same scenarios shown in Fig. 10. Let us first take a look to the lax settings ($d^{max} = 0.5$ s, $\rho^{min} = 0.4$, depicted with red lines in the figure). When δ_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 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 at the expense of higher BS power consumption. Conversely, when δ_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 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 δ_2 values (with mild differences for the highest settings).

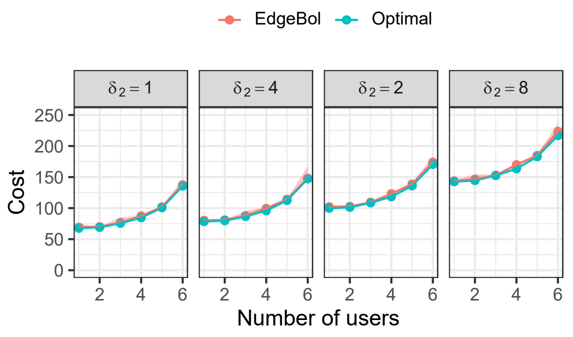

Figure 12: Empirical optimality gap in scenarios with multiple heterogeneous users. Each scenario has N users with different SNR conditions: user 1 has the best channel conditions (SNR = 30 dB in average) and every additional user has 20% lower SNR. We evaluate different values of δ_2. $\delta_1 = 1$.

6.4 Heterogeneous users

In an effort to reduce the problem of the dimensionality, we aggregate statistics of individual users (mean SNR, variance SNR, etc.) when describing the context. See §4.4 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 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 [26] (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 in average) and every additional user has 20% lower SNR.[4] 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. 12 depicts the cost of the system (as defined in eq. (1)) for scenarios with different values of N. We do this for different weights δ_2 in the trade-off between the service's power consumption and that of the vBS ($\delta_1 = 1$ in all scenarios).

We compare the performance of EdgeBOL with that of 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%. Though it is not shown in the plot due to space constraints, EdgeBOL satisfies the service constraints with probability 0.98. This validates that aggregated statistics across users suffice to provide good performance yet 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

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

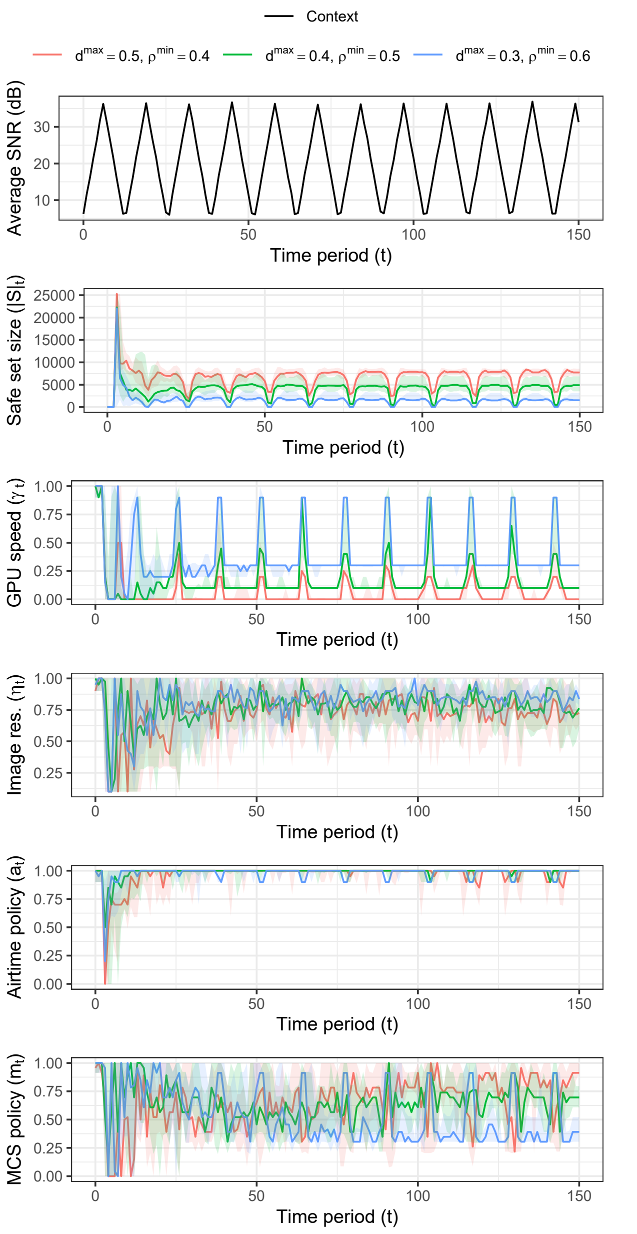

Figure 13: Evolution of policies for dynamic contexts ($\delta = 8$).

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 this degradation of mean wireless conditions.

6.5 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 plot in Fig. 13, and set $\delta_1 = 1$ and $\delta_2 = 8$. The top right plot depicts the size of the safe control set over time. As expected, the safe set quickly reduces within

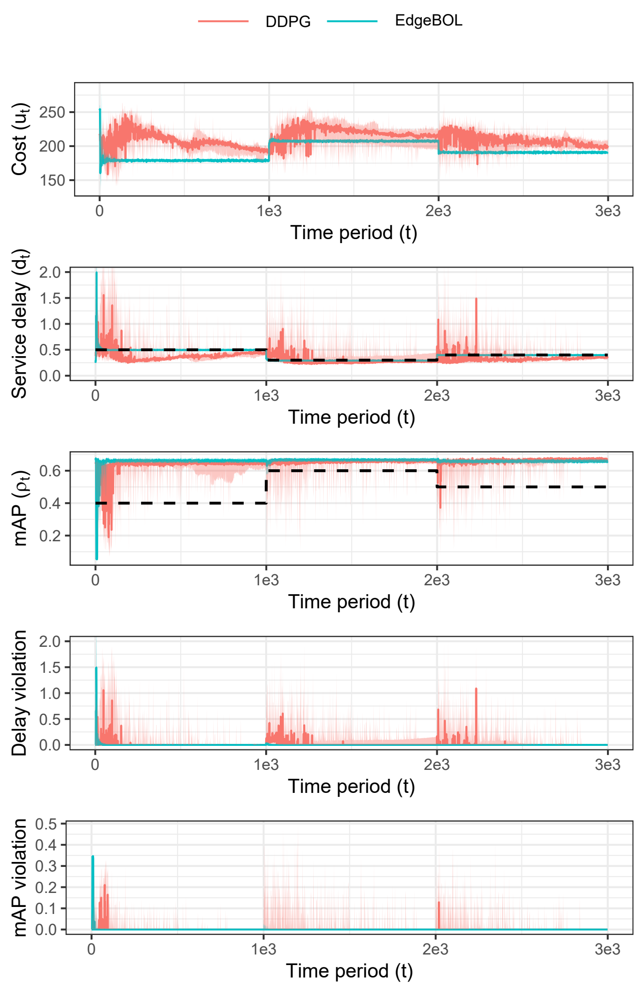

Figure 14: Evolution of delay and mAP upon changes on the constraint settings for EdgeBOL and a DDPG approach implemented with neural networks ($\delta = 8$).

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 δ 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 this policy dynamics is substantially different for diverse values of δ (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) [37]. This benchmark is inspired by [4], 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. 4). The DDPG algorithm uses an actor-critic neural network architecture, but the critic, instead of approximating the Q function (full-RL problem), it 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 [17]. We mildly modified the architecture presented in [4] 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. 14 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.

7 CONCLUSIONS

The energy-aware implementation of AI services at the network edge is increasingly important for performance, economic and environmental reasons [45], [59]. 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.

ACKNOWLEDGEMENTS

This work has been supported by the European Commission through Grant No. 101017109 (DAEMON project), and the CERCA Programme/Generalitat de Catalunya.

REFERENCES

[1] Juan J Alcaraz, Jose A Ayala-Romero, Javier Vales-Alonso, and Fernando Losilla-López. 2020. Online reinforcement learning for adaptive interference coordination. *Transactions on Emerging Telecommunications Technologies* 31, 10 (2020), e4087.

[2] Sanae Amani, Mahnoosh Alizadeh, and Christos Thrampoulidis. 2020. Regret Bounds for Safe Gaussian Process Bandit Optimization. *arXiv preprint arXiv:2005.01936* (2020).

[3] Jose A Ayala-Romero, Juan J Alcaraz, Andrea Zanella, and Michele Zorzi. 2019. Online learning for energy saving and interference coordination in hetnets. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1374–1388.

[4] Jose A Ayala-Romero, Andres Garcia-Saavedra, Marco Gramaglia, Xavier Costa-Perez, Albert Banchs, and Juan J Alcaraz. 2020. vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs. *IEEE Transactions on Mobile Computing* (2020).

[5] Alt Bastian et al. 2019. CBA: Contextual Quality Adaptation for Adaptive Bitrate Video Streaming. In *Proc. of IEEE INFOCOM*.

[6] Dario Bega et al. 2018. CARES: Computation-aware Scheduling in Virtualized Radio Access Networks. *IEEE Trans. on Wireless Communications* 17, 12 (2018), 7993–8006.

[7] D. Bega et al. 2020. DeepCog: Optimizing Resource Provisioning in Network Slicing With AI-Based Capacity Forecasting. *IEEE J. Sel. Areas Commun.* 38, 2 (2020), 361–376.

[8] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. 2016. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *arXiv preprint arXiv:1602.04450* (2016).

[9] Adam D Bull. 2011. Convergence Rates of Efficient Global Optimization Algorithms. *Journal of Machine Learn. Res.* 12 (2011), 2879–2904.

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

[11] Ayon Chakraborty, Eugene Chai, Karthikeyan Sundaresan, Amir Khojastepour, and Sampath Rangarajan. 2018. SkyRAN: a self-organizing LTE RAN in the sky. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. 280–292.

[12] China Mobile Limited. 2020. 2020 Sustainability Report. white Paper.

[13] Jie Chuai, Zhitang Chen, Guochen Liu, Xueying Guo, Xiaoxiao Wang, Xin Liu, Chongming Zhu, and Feiyi Shen. 2019. A Collaborative Learning Based Approach for Parameter Configuration of Cellular Networks. In *Proceedings of IEEE INFOCOM*.

[14] Cisco. 2020. 5G network slicing: cross-domain orchestration and management will drive commercialization. Technical Specification. https://www.cisco.com/c/dam/en/us/products/collateral/cloud-systems-management/network-services-orchestrator/white-paper-sp-5g-network-slicing.pdf

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

[16] L. Diez et al. 2020. LaSR: A Supple Multi-Connectivity Scheduler for Multi-RAT OFDMA Systems. *IEEE Transactions on Mobile Computing* 19, 3 (2020), 624–639. https://doi.org/10.1109/TMC.2018.2876847

[17] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2016. Deep Reinforcement Learning in Large Discrete Action Spaces. arXiv:1512.07679 [cs.AI]

[18] David Duvenaud. 2014. *Automatic model construction with Gaussian processes.* Ph.D. Dissertation. University of Cambridge.

[19] Mark Everingham et al. 2015. The pascal visual object classes challenge: A retrospective. *International journal of computer vision* 111, 1 (2015), 98–136.

[20] Marcello Fiducioso, Sebastian Curi, Benedikt Schumacher, Markus Gwerder, and Andreas Krause. 2019. Safe contextual Bayesian optimization for sustainable room temperature PID control tuning. *arXiv preprint arXiv:1906.12086* (2019).

[21] Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. 2010. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*. 586–594.

[22] Apostolos Galanopoulos, Jose A. Ayala-Romero, George Iosifidis, and Douglas Leith. 2020. Bayesian Online Learning for MEC Object Recognition Systems. In *2020 IEEE Global Communications Conference (Globecom)*. IEEE.

[23] Gines Garcia-Aviles et al. 2021. Nuberu: Reliable RAN Virtualization in Shared Platforms. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking* (New Orleans, Louisiana) *(MobiCom '21)*. Association for Computing Machinery, New York, NY, USA, 749–761. https://doi.org/10.1145/3447993.3483266

[24] Andres Garcia-Saavedra and Xavier Costa-Perez. 2021. O-RAN: Disrupting the Virtualized RAN Ecosystem. *IEEE Communications Standards Magazine* (2021). https://doi.org/10.1109/MCOMSTD.101.2000014

[25] Andres Garcia-Saavedra, George Iosifidis, Xavier Costa-Perez, and Douglas J Leith. 2018. Joint optimization of edge computing architectures and radio access networks. *IEEE J. Sel. Areas Commun.* 36, 11 (2018), 2433–2443.

[26] Ismael Gomez-Miguelez et al. 2016. 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*. 25–32.

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

[28] GSMA Associatio. 2020. 5G energy efficiencies: green is the new black. white Paper.

[29] Morteza Hashemi, Ashutosh Sabharwal, C. Emre Koksal, and Ness B Shroff. 2018. Efficient Beam Alignment in Millimeter Wave Systems Using Contextual Bandits. In *Proc. of IEEE INFOCOM*.

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

[31] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2015. OverLay: Practical Mobile Augmented Reality. In *Proc. of ACM MobiSys*.

[32] Junchen Jiang et al. 2018. Chameleon: Scalable Adaptation of Video Analytics. In *Proc. of ACM SIGCOMM*.

[33] Andreas Krause and Cheng S Ong. 2011. Contextual gaussian process bandit optimization. In *Advances in neural information processing systems*. 2447–2455.

[34] Hongshan Li et al. 2018. JALAD: Joint Accuracy- and Latency-Aware Deep Structure Decoupling for Edge-Cloud Execution. In *Proc. of IEEE ICPADS*.

[35] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. 661–670.

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

[37] Timothy P. Lillicrap et al. 2016. Continuous control with deep reinforcement learning. In *Proc. of ICLR 2016*. San Juan, Puerto Rico.

[38] Tsung-Yi Lin et al. 2015. Microsoft COCO: Common Objects in Context. arXiv:1405.0312 [cs.CV]

[39] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge Assisted Real-time Object Detection for Mobile Augmented Reality. In *Proc. of ACM MobiCom*.

[40] Cristina Marquez et al. 2018. 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*. 191–206.

[41] Cristina Marquez, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2019. Resource sharing efficiency in network slicing. *IEEE Transactions on Network and Service Management* 16, 3 (2019), 909–923.

[42] Jose Mendes et al. 2019. Cellular access multi-tenancy through small-cell virtualization and common RF front-end sharing. *Computer Communications* 133 (2019), 59–66.

[43] Microsoft. [n.d.]. Seeing AI. https://www.microsoft.com/en-us/ai/seeing-ai.

[44] Faris Mismar, Jinseok Choi, and Brian L. Evans. 2019. A Framework for Automated Cellular Network Tuning With Reinforcement Learning. *IEEE Transactions on Communications* 67, 10 (2019), 7152–7167.

[45] GSMA Future Networks. 2019. Energy Efficiency: An Overview. https://www.gsma.com/futurenetworks/wiki/energy-efficiency-2

[46] Nokia. 2020. Network Slicing explained. Technical Specification. https://www.nokia.com/about-us/newsroom/articles/network-slicing-explained/

[47] O-RAN Alliance. 2020. O-RAN Near-Real-time RAN Intelligent Controller Architecture E2 General Aspects and Principles 1.01 (O-RAN.WG3.E2GAP-v01.01). Technical Specification.

[48] O-RAN Alliance. 2021. O-RAN A1 interface: Application Protocol 3.01 (O-RAN.WG2.A1AP-v03.01). Technical Specification.

[49] O-RAN Alliance. 2021. O-RAN A1 interface: Transport Protocol 1.01 (ORAN-WG2.A1.TP-v01.01). Technical Specification.

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

[51] O-RAN Alliance. 2021. O-RAN-WG1-O-RAN Architecture Description (O-RAN.WG1.O-RAN-Architecture-Description-v04.00). Technical Specification.

[52] O-RAN Alliance. 2021. 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.

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

[54] D. Raca et al. 2020. On Leveraging Machine and Deep Learning for Throughput Prediction in Cellular Networks: Design, Performance, and Challenges. *IEEE Communications Magazine* 58, 3 (2020), 11–17.

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

[56] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence* 39, 6 (2016), 1137–1149.

[57] Paat Rusmevichientong and John N Tsitsiklis. 2010. Linearly parameterized bandits. *Mathematics of Operations Research* 35, 2 (2010), 395–411.

[58] Yoan Russac, Olivier Cappé, and Aurélien Garivier. 2020. Algorithms for Non-Stationary Generalized Linear Bandits. *arXiv preprint arXiv:2003.10113* (2020).

[59] New Scientist. 2019. Creating an AI can be five times worse for the planet than a car. https://www.newscientist.com/article/2205779-creating-an-ai-can-be-five-times-worse-for-the-planet-than-a-car/

[60] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (2016), 148–175.

[61] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. 2015. Safe exploration for optimization with Gaussian processes. In *International Conference on Machine Learning*. PMLR, 997–1005.

[62] Yanan Sui, Vincent Zhuang, Joel W Burdick, and Yisong Yue. 2018. Stagewise safe bayesian optimization with gaussian processes. *arXiv preprint arXiv:1806.07555* (2018).

[63] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[64] Jianhua Tang, Byonghyo Shim, and Tony QS Quek. 2019. Service multiplexing and revenue maximization in sliced C-RAN incorporated with URLLC and multicast eMBB. *IEEE Journal on Selected Areas in Communications* 37, 4 (2019), 881–895.

[65] K. Wang et al. 2019. Computing Aware Scheduling in Mobile Edge Computing System. *Springer Wireless Networks* (2019), 1–17.

[66] Christopher KI Williams and Carl Edward Rasmussen. 2006. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA.

[67] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. https://github.com/facebookresearch/detectron2.

[68] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf. 2014. Deepface: Closing the Gap to Human-level Performance in Face Verification. In *Proc. of IEEE CPVR*.

[69] P. Yang et al. 2020. Edge Coordinated Query Configuration for Low-Latency and Accurate Video Analytics. *IEEE Trans. on Industrial Informatics* 16, 7 (2020), 4855–4864.

[70] Lanfranco Zanzi et al. 2021. LACO: A Latency-Driven Network Slicing Orchestration in Beyond-5G Networks. *IEEE Transactions on Wireless Communications* 20, 1 (2021), 667–682. https://doi.org/10.1109/TWC.2020.3027963

[71] Lanfranco Zanzi, Vincenzo Sciancalepore, Andres Garcia-Saavedra, and Xavier Costa-Pérez. 2018. OVNES: Demonstrating 5G network slicing overbooking on real deployments. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 1–2.

[72] C. Zhang, P. Patras, and H. Haddadi. 2019. Deep Learning in Mobile and Wireless Networking: A Survey. *IEEE Commun. Surv. Tutor.* 21, 3 (2019), 2224–2287.

[73] Wenxiao Zhang, Bo Han, and Pan Hui. 2018. Jaguar: Low Latency Mobile Augmented Reality with Flexible Tracking. In *Proc. of ACM Conference on Multimedia*.

[74] Ziyao Zhang, Liang Ma, Konstantinos Poularakis, Kin K. Leung, and Wu Lingfei. 2019. DQ Scheduler: Deep Reinforcement Learning Based Controller Synchronization in Distributed SDN. In *Proceedings of IEEE ICC*.