SEM-O-RAN: Semantic O-RAN Slicing for Mobile Edge Offloading of Computer Vision Tasks

Corrado Puligheddu, *Member, IEEE,* Jonathan Ashdown, *Senior Member, IEEE,*
Carla Fabiana Chiasserini, *Fellow, IEEE,* and Francesco Restuccia, *Senior Member, IEEE*

Abstract—The next generation of mobile networks (NextG) will require careful resource management to support edge offloading of resource-intensive deep learning (DL) tasks. Current slicing frameworks treat all DL tasks equally without adjusting to their high-level objectives, resulting in sub-optimal performance. To overcome this, we propose SEM-O-RAN, a semantic and flexible slicing framework for computer vision task offloading in NextG Open RANs. Our framework accounts for the semantic nature of object classes as well as the level of data quality to optimally tailor data compression and minimize the usage of networking and computing resources. In fact, we show that different object classes tolerate different levels of image compression while preserving detection accuracy. To address the above issues, we first present the mathematical formulation of the Semantic Flexible Edge Slicing Problem (SF-ESP), which turns out to be NP-hard. We thus define a greedy algorithm to solve it efficiently, which is also able to always select the resource allocation that yields the best resource utilization, whenever multiple allocations satisfy the DL task requirements. We evaluate SEM-O-RAN's performance through extensive numerical analysis and real-world experiments on the Colosseum testbed, considering state-of-the-art computer-vision tasks and DL models. The obtained results demonstrate that SEM-O-RAN allocates up to 169% more tasks and obtains 52% higher revenues than the state of the art.

Index Terms—network slicing, computation offloading, O-RAN, semantics, NextG, edge computing, resource allocation

1 Introduction

Connected mobile devices running computer vision (CV)-based, mission-critical operations, such as autonomous vehicles and drones, have to timely execute demanding processing tasks, which require as input high-resolution images (e.g., video frames) or three-dimensional LIDAR (Light Detection and Ranging) data [2]. Examples of these tasks include multi-object classification and object tracking – operations commonly required for autonomous navigation and typically performed through deep learning (DL) models [3]. Because of the often limited computing power and energy budget of mobile devices, it may be convenient or necessary to offload such tasks through the Radio Access Network (RAN) to the network edge, where they can be timely executed. However, computing resources at the edge may be insufficient to run multiple tasks that are offloaded concurrently, and the RAN itself may get saturated, causing increased delays and, ultimately, compromising the application performance.

To successfully offload CV tasks, mobile devices can benefit from RAN slicing [4]–[9], which allows Virtual Network Operators (VNOs) to use the virtualized and isolated RAN computational and networking resources provided by Mobile Network Operators (MNOs). It is noteworthy that the recent Open RAN (O-RAN) architecture offers full support for RAN slicing. This architecture, which disaggregates the 5G-and-beyond cellular networks (NextG) RAN hardware from its software components, allows for advanced RAN optimizations through fine-grained real-time control of the RAN components [10]–[12].

Existing Issues. Currently, the state of the art defines edge-assisted tasks in a rigid and fixed manner, and ignores additional information that could be used to improve the accuracy of the tasks and the efficiency of their execution environment. We refer to this additional information as task semantics. In this context, we define task semantics as the meaning, context, and purpose of a task that deepen and enrich its definition and that can be derived from the high-level intent of the VNO issuing the task request. Ignoring task semantics prevents further optimization, which leads to sub-optimal performance and revenues. Moreover, many state-of-the-art slicing frameworks are not compliant with the O-RAN specifications; consequently, they are not able to enforce granular RAN control for advanced optimizations.

To tackle these issues, we propose SEM-O-RAN, the first semantic slicing framework for edge-assisted mobile applications based on O-RAN. SEM-O-RAN provides two core innovations, as detailed below.

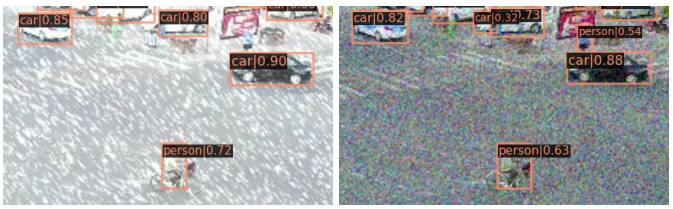
- Existing work considers rigid task definitions, with pre-defined and fixed resource needs. Conversely, SEM-O-RAN semantically considers a richer task definition specified in terms of end-to-end latency and per-class accuracy requirements peculiar to each task, thus allowing for flexibility in the allocation of edge resources. In particular, SEM-O-RAN considers all combinations of resource allocations that satisfy the task requirements, then selects the resource allocation that leads to the best system performance. Through flexible allocation, SEM-O-RAN can allocate

- *C. Puligheddu and C. F. Chiasserini are with the Department of Electronics and Telecommunications, Politecnico di Torino, Italy.*
- *J. Ashdown is with Air Force Research Laboratory, United States.*
- *F. Restuccia is with the Institute for the Wireless Internet of Things, Northeastern University, United States.*

Approved for Public Release; Distribution Unlimited: AFRL-2022-1622.
This is an extended version of our IEEE INFOCOM 2023 paper [1].

(a) Compression 0.87x, 103 KB (b) Compression 0.50x, 59 KB

Fig. 1: Stronger compression rates make some objects undetectable and/or harder to detect by computer vision DL-based models.



(a) Compression 0.77x, snow (b) Compression 0.78x, noise

Fig. 2: External factors such as snow (a) or Gaussian noise (b) can cause unintentional data quality degradation which affects object detection accuracy.

9.1% more CV tasks than the state of the art.

- To further reduce network utilization, SEM-O-RAN applies data compression according to the task semantics, i.e., considering the target object classes, prior data quality degradation, and the per-class accuracy requirements. Our key intuition is that different object classes have different tolerances to data compression, thus providing different optimization margins. For example, in Fig. 1, where a state-of-the-art object detection model is applied on a frame captured by a smart city camera, the car and person objects are correctly identified even when the image is heavily compressed, as opposed to the bicycle, which needs lower data compression. Similarly, in Fig. 2, because of unintentional data quality degradation affecting the picture quality, the same model is not able to detect the bicycle object even with a light compression factor. Thanks to semantic data compression, SEM-O-RAN allocates 57% more tasks than the state of the art.

By combining these two key concepts, SEM-O-RAN delivers a performance improvement of up to 169% compared to [5].

Technical Challenges. The formulation of a mathematical slicing solution that accommodates flexible resource allocation and semantic data compression in an O-RAN framework is particularly challenging, since (i) the relationship between the allocated resources, data compression, per-class accuracy, and end-to-end latency cannot be easily expressed in closed form, because of the high non-linearity of CV DL models and the complex modeling of network communications, and (ii) the flexible resource allocation introduces significant complexity in the optimization problem. *To the best of our knowledge, SEM-O-RAN is the first work to holistically address these two challenges at the same time.*

Novel Contributions. Our main contributions are summarized as follows.

- We present our key ideas of semantics-based data compression and flexible resource allocation and we show that they can be used to optimize edge resource consumption of offloaded CV tasks. Even though no other existing work has considered yet such concepts jointly to provide further network optimization at the edge, our ideas are not exclusive to our work and could be potentially integrated into existing solutions.
- We design SEM-O-RAN, the first *semantic* and *flexible* slicing framework to support edge-assisted DL task offloading in NextG networks. SEM-O-RAN is fully compliant with the O-RAN specifications, which allow for the near-real-time control of slices configuration. To perform the actual slicing, we mathematically formulate the Semantic Flexible Edge Slicing Problem (SF-ESP), which (i) maximizes the revenues for the MNO, (ii) optimizes the number of DL computer vision tasks executed at the RAN edge while (iii) guaranteeing strict requirements satisfaction on the DL task latency/accuracy and (iv) avoiding resource over-provisioning. The SF-ESP is fundamentally different from existing formulations, since (i) it incorporates highly non-linear relation between slicing, compression, end-to-end latency, and classification accuracy; (ii) it adopts flexible resource allocation to balance the consumption of the different types of edge resources and avoid the premature depletion of the most requested ones. We demonstrate that the SF-ESP is NP-hard, and propose a greedy algorithm to solve it efficiently;
- We evaluate SEM-O-RAN through extensive numerical analysis and through a prototype implemented on the Colosseum network emulator [13]. For evaluation purposes, we consider two state-of-the-art CV problems, i.e., multi-object detection and instance segmentation. To address the former, we select the state-of-the-art YOLOX model [14] trained on the Common Objects in Context (COCO) dataset [15]; for the latter, we use the BiSeNet v2 model [16] trained on the Cityscapes dataset [17].
  We compare SEM-O-RAN to 5 baselines, including the state-of-the-art Sl-EDGE framework [5]. Our results show that SEM-O-RAN improves the number of allocated computer vision tasks by up to 169% and by 72% on average with respect to Sl-EDGE. To allow for replicability and benchmarking, we have released our MATLAB algorithm as open-source[1].

Paper organization. The remainder of this paper is organized as follows. Sec. 2 details the fundamental ideas underpinning SEM-O-RAN, namely, the semantic compression and the flexible resource allocation. Sec. 3 presents the design of our SEM-O-RAN framework, based on the O-RAN architecture, which is briefly introduced in Sec. 3.1. We first describe the architecture, functional blocks, and interfaces of SEM-O-RAN (Sec. 3.2), then we provide an overview of

1. https://github.com/corrado113/Semoran

the system workflow (Sec. 3.3). Sec. 4 introduces the system model (Sec. 4.1), the SF-ESP problem formulation (Sec. 4.2), and the greedy heuristics able to solve it efficiently (Sec. 4.3). Sec. 5 explains our testing methodology (Sec. 5.1), and it shows both the numerical analysis of SEM-O-RAN and its alternatives (Sec. 5.2), and the prototype implementation on Colosseum (Sec. 5.3). Relevant related work is discussed in Sec. 6, highlighting the differences between our approach and the existing solutions. Finally, Sec. 7 concludes the paper by wrapping up our contributions and providing directions for future research.

2 Key Concepts in SEM-O-RAN

The first main concept driving the design of SEM-O-RAN is the semantic-based slicing. In Fig. 1, we notice that different target classes have different tolerances to image compression. Furthermore, Fig. 2 highlights that a similar tolerance can be found also in case an image is captured in inclement weather and, more in general, whenever the data quality degrades. Intuitively, some classes are semantically "harder" than others, especially in some circumstances. For example, a person or a car can be more easily identified in a noisy image as opposed to a bicycle or a backpack. In the left side of Fig. 3, we quantitatively evaluate this behavior, showing the mean Average Precision (mAP) values corresponding to multi-object detection, which is one of the mobile sensing applications we focus on in this work. The mAP is a metric used to evaluate object detection models, defined as the mean over all object classes of the area under the Precision-Recall Curve. Fig. 4 shows the mAP that we measured for the same mobile application in case the data quality is affected by common atmospheric agents. Clearly, when the corruption severity is maximum (Fig. 4b), the mAP decreases significantly compared to the case in which the severity is minimum (Fig. 4a). Interestingly, the different object classes have different tolerances to corruption effects, e.g., "bags" really struggles when affected by motion blur compared to other classes.

The takeaway point here is that there is a margin for significant compression on the images sent to the edge for inference, according to the condition in which the images have been captured and the object classes we are interested in, while still obtaining acceptable inference accuracy on average. This compression margin ultimately constitutes the semantic information of a task, encapsulating its meaning, context, and purpose. The semantic knowledge allows for precise compression and significant bandwidth saving, which consequently allows SEM-O-RAN to offload more tasks.

The second concept in the design of SEM-O-RAN is flexibility in task resource allocation. Indeed, a task requires many different kinds of resources, from networking to computation and storage. Therefore, the slicing algorithm can allocate different amounts of resources in each category and still meet performance requirements. To illustrate this point, the right side of Fig. 3 shows experimental end-to-end task latency results of inference on the state-of-the-art YOLOX deep neural network (DNN) model for object detection [14] computed using the Colosseum network emulator [13], as a function of the allocated Resource Block Groups (RBGs) and GPUs. In this plot, 10 images per second were generated from a single User Equipment (UE), without employing image compression.

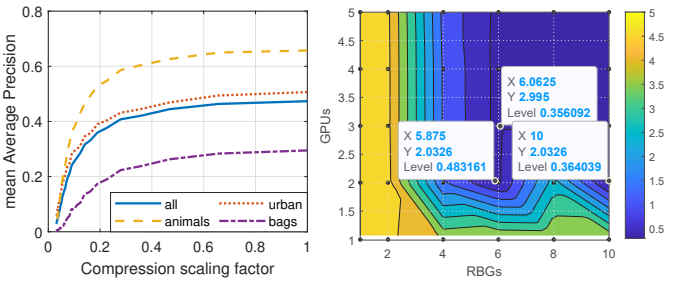


Fig. 3: (Left) Mean Average Precision (mAP) as a function of the compression scaling factor for the application classes defined in Tab. 2; (Right) Experimental latency as a function of allocated Resource Block Groups (RBGs) and GPUs.

The key takeaway is that more than one combination of RBG/GPU allocations can lead to the same latency performance while allowing for more allocated tasks.

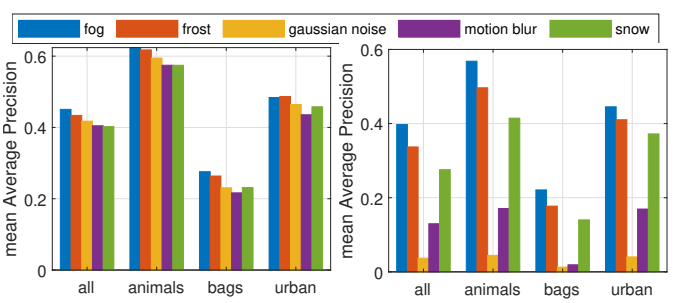
For example, let us assume that 25 RGBs and 4 GPUs are the maximum radio and computational resources available in the RAN, and that two tasks (say, τ_1 and τ_2) requiring 0.4 s of latency need to be allocated. According to Fig. 3, two different RGB/GPU allocations meet the 0.4 s latency requirement, namely, (6, 3) and (10, 2). Let us assume τ_1 is allocated (6, 3), which is the most resource-efficient allocation. In this case, however, τ_2 could not be allocated as there would only be 1 GPU left. Instead, if (10, 2) is allocated to τ_1, τ_2 can be allocated since 2 GPUs and 10 RBGs are still available.

3 The SEM-O-RAN Framework

In this section, we first provide a brief overview of the O-RAN logical architecture, then we introduce the architecture of the proposed SEM-O-RAN framework motivating our design choices; finally, we detail the functional flow of SEM-O-RAN.

3.1 Background Notions on O-RAN

The core philosophy behind O-RAN is the clear separation between the RAN software and hardware [18], by disaggregating the RAN into a Radio Unit (RU), Centralized Unit (CU) and Distributed Unit (DU). The RU implements



(a) Corruption severity 20% (b) Corruption severity 100%

Fig. 4: mAP over different object classes in the COCO dataset [15], with multiple corruption effects.

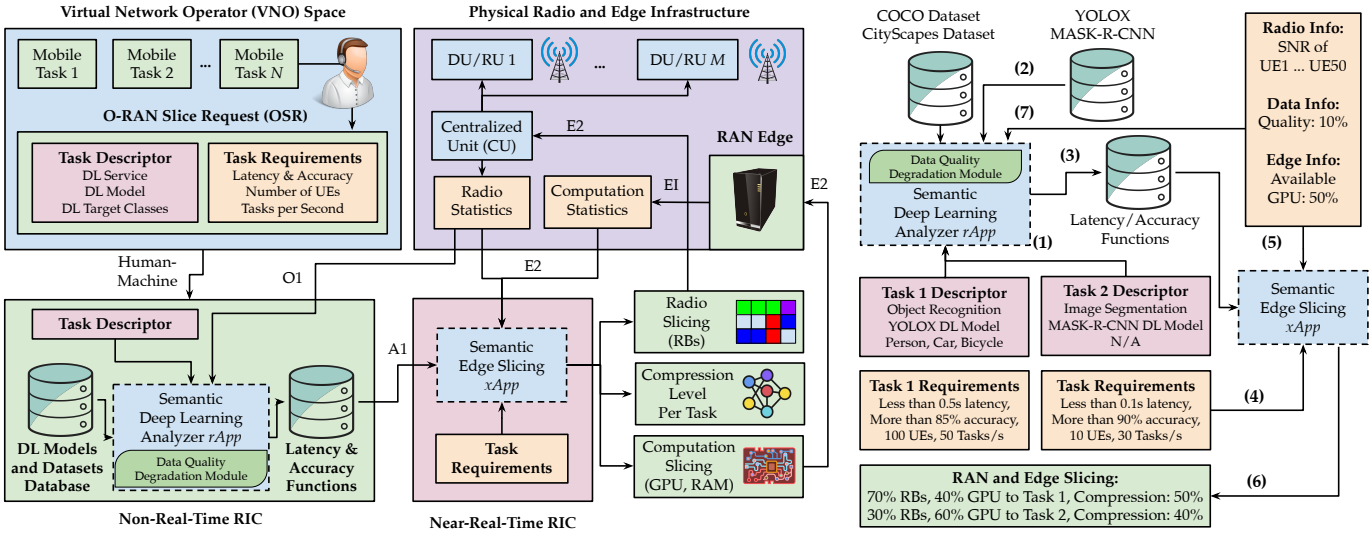


Fig. 5: Functional Blocks and O-RAN Interfaces used by SEM-O-RAN (Left); A Walk-through of SEM-O-RAN (Right).

extremely low-latency operations related to the lower Physical Layer (PHY). The DU, in turn, implements the upper portion of the PHY, as well as the Medium Access Control (MAC) and Radio Link Control (RLC). These are controlled in a softwarized manner by a RAN Intelligent Controller (RIC), which is further divided into a Non-real-time RIC, handling high-level RAN orchestration and management, and a Near-real-time RIC, implementing fine-grained control policies such as RAN slicing, scheduling, and load balancing. Third-party applications called xApps and rApps can be hosted in the Non-real-time RIC and Near-real-time RIC, respectively. The former may implement data-driven control loops or may be used for RAN-specific data collection and analysis. On the other hand, rApps may implement high-level policy guidance as well as application-level interfaces. Please refer to [10] for more information regarding O-RAN.

3.2 SEM-O-RAN: Functional Blocks and Interfaces

The design of the SEM-O-RAN framework is driven by the required ability to dynamically allocate in near-real-time edge resources for the offloaded tasks. The choice of the tasks to be admitted and of the allocated resources depends on the tasks semantics, i.e., the maximum tolerable data compression, the current network state, and the available resources. To achieve this, it is natural to leverage the O-RAN architecture, which enables receiving network state metrics and setting slice resources with applications running in the RIC. Furthermore, its multi-timescale architecture is well suited to accommodate a Near-RT xApp for resource allocation, along with a Non-RT rApp for computing the accuracy and latency functions necessary to calculate the solution of the SF-ESP (introduced in Sec. 4).

Fig. 5 shows the functional blocks of SEM-O-RAN, as well as how the blocks are mapped into the O-RAN modules and interfaces. The core modules of SEM-O-RAN are the Semantic Deep Learning Analyzer (SDLA) and the Semantic Edge Slicing Module (SESM), which respectively reside in the Non-real-time RIC and Near-real-time RIC portions of the O-RAN as an rApp and an xApp.

SEM-O-RAN and the VNO communicate through a human-machine interface [10]. Each VNO requires slices for a given set of mobile tasks. Each mobile task corresponds to an O-RAN Slice Request (OSR), which is composed of a Task Description (TD) field and a Task Requirements (TR) field. The TD is used to define the DL service requested, the DL model to be used and the DL target classes, while the TR specified the latency and accuracy requirements, the number of UEs requested, and the number of jobs (e.g., inference on an image) per second generated by the UEs. As shown in Fig. 5-Right, a TD could be ("Object Recognition", "YOLOX", "{Person, Car, Bicycle}"), with the corresponding TR defined as ("0.5 s max latency", "0.85 min accuracy", "100 UEs", "50 jobs/sec").

The TD is submitted to the SDLA rApp, which is tasked to compute the latency function $l_\tau(\cdot)$ and accuracy function $a_\tau(\cdot)$, which output the latency and accuracy values associated to a given TD, a given level of task compression and amount of edge resources (see Sec. 4.1 for a more formal definition). The accuracy function is computed through representative datasets, considering the data quality deterioration caused by both the intentional data compression and unintentional input quality degradation caused by external interference. The Data Quality Degratation Module (DQDM) takes care of applying artificial data degradation using image corruption libraries that emulate the effects of real-world phenomena. The latency function can be pre-computed through network emulation and then refined using real monitoring data. The latency and accuracy functions are then shared with the SESM xApp running in the Near-real-time RIC. These are ultimately used to solve the SF-ESP.

The output of the SF-ESP xApp is ultimately three-fold: (i) select which tasks to admit; (ii) their compression level; and (iii) the computational resources (GPU/RAM) and the number of Physical Resource Blocks (PRBs) assigned to each admitted task. Real-time information about the available computational resources and the current radio-level statistics are provided to the xApp through the E2 interface. The former is used by the SF-ESP to properly account for the resources that are actually available in the RAN

edge, which are shared through an Enriched Interface (EI) to the RAN. The latter are used to select and update the appropriate latency function from the SDLA according to the radio channel status. The radio slicing and computation slicing are respectively shared with the CU and the RAN edge through the E2 interface. The CU then takes care of propagating the slicing information to the appropriate DUs. The compression level per task is fed back to the VNO, which then communicates this information to the UEs. We acknowledge that this is impractical at scale, however, as of now, the O-RAN specifications do not allow for direct communication between RIC apps and device applications.

3.3 A Walk-through of SEM-O-RAN.

We provide a simplified walk-through of an actual slicing request and enforcement operation in SEM-O-RAN on the right side of Fig. 5. First, TDs are sent to the SDLA rApp (Step 1). If latency/accuracy functions are not already present, they are computed by using the appropriate datasets/models and stored in the Non-real-time RIC. To consider possible data quality degradation, according to the task application class, the dataset images are also artificially degraded by the DQDM to different levels of quality to obtain more robust accuracy functions. (Step 2). In case latency/accuracy functions are ready, they are sent to the SESM xApp (Step 3), which receives the TRs (Step 4) and the current status of the radio channel, data quality, and edge resources (Step 5), which are used used to produce the RAN and edge slicing (Step 6). The data quality can be directly estimated by the mobile device sensors or inferred indirectly by the system, e.g., using smart weather stations. Finally, the current radio/edge status may be shared with the SDLA rApp for refinement of the latency functions (Step 7) to be used for future slicing decisions. If slice requests change, e.g., because a new task is created, a new slicing allocation is computed. Note that new and already running tasks are equally considered, thus it may happen that previously running tasks are no longer admitted and must be terminated.

4 Semantic Flexible Edge Slicing (SF-ESP)

In this section, we introduce the system model in Sec. 4.1. Then, we formalize the SF-ESP and prove its NP-hardness in Sec. 4.2. We propose a greedy algorithm in Sec. 4.3.

4.1 System Model

We define an *application class* as a high-level objective that has to be achieved through the execution of one or more *DL tasks* with certain requirements. Every application class specifies the DL service, the classes of objects over which the DL service is supposed to be applied to, and the requirements for maximum delay and minimum expected accuracy that a device running that application must satisfy. For example, a monitoring application class could require the detection and tracking of person and vehicle objects located in the proximity of a road intersection with a minimum expected accuracy of 0.50 mAP and maximum end-to-end delay of 800 ms. Fig 6 shows an example with 3 application classes.

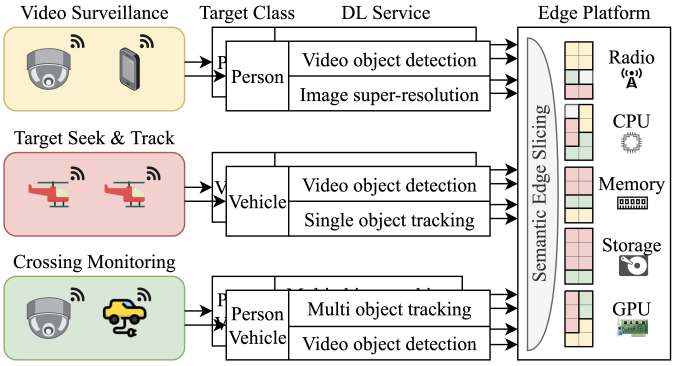


Fig. 6: System model example with $C = 3$ application classes, each of which is run by $|D_c| = 2, \forall c \in \mathbb{C}$ devices. Each device requests $|T_{cd}| = 2, \forall c, d$ tasks to be offloaded to the Edge infrastructure, thus requiring the concurrent allocation of $m = 5$ types of radio and compute resources.

Let $\mathbb{C} = \{1, \dots, C\}$ be the set containing the application classes. The set of devices running an application class $c \in \mathbb{C}$ is D_c. A *device* $d \in D_c$, according to its application class c, submits a set of tasks T_{cd} to be offloaded onto the RAN edge using its wireless link. A task, uniquely identified at the system level by the tuple (c, d, t), is the periodic execution at the edge of a DL service over certain classes of objects, which is applied over a stream of inference data sent by the device, and whose results are then sent back to the requesting device, for a period of time not known a priori. To make the notation clearer, let us define $\tau = (c, d, t) \in \mathbb{T}$ as a generic task. The offer O_τ indicates the economic value associated with the execution of task τ, assuming that it is executed according to the performance requirements defined subsequently. A task that is executed violating the performance requirements does not yield any revenue. Given τ, we define the compression scaling factor as $z_\tau \in (0, 1] = \{x \in \mathbb{R} | 0 < x \leq 1\}$ such that the bitrate of the inference data stream is scaled by that factor, i.e., $b^z_\tau = z_\tau b_\tau$, where b^z_τ is the compressed stream and b_τ is the original stream without any applied compression. A higher scaling factor implies higher inference accuracy. A lower scaling factor sacrifices the data quality to decrease the file size, thus requiring lower network bandwidth and improving latency. In our model, we assume that the inference data original stream size is constant and depends on the application class. Furthermore, we assume the compression latency as constant for different scaling factors.

Given the type of edge resource $k \in \mathbb{K} = \{1, \dots, m\}$, we denote with $s_{\tau k}$ the amount of resource of type k assigned to each task $\tau \in \mathbb{T}$. Resource types can be networking, e.g., Physical Resource Blocks (PRBs), as well as computational, e.g., GPU time and memory needed to run the DL models in the RAN edge. Since edge resources are limited, the total amount of assigned resources of type k cannot exceed the capacity, $S_k, \forall k$. Further, to account for the cost of the resources, for each resource type, we define the resource price p_k, that is the price charged to utilize a unit of resource of type k. The execution of a task τ is economically convenient only if it yields a profit, defined as the difference between the offer (bid) and the resource cost (ask), i.e., if $O_\tau > \sum_k^m p_k s_{\tau k}$. Thus, careful resource allocation is needed

to avoid over-provisioning and make the offloading service economically sustainable.

The performance requirements are imposed by the related application class. Such requirements are defined in terms of (i) minimum expected prediction accuracy A_c on the selected object classes, and (ii) maximum expected end-to-end latency L_c for each of the applications running on the mobile devices belonging to class c. By defining $a_\tau(z_\tau)$ and $l_\tau(z_\tau, s_\tau)$ as, respectively, the accuracy and latency functions of task τ, an allocation solution is acceptable only if $a_\tau \geq A_c$ and $l_\tau \leq L_c$, $\forall \tau = (c, d, t) \in \mathbb{T}$. The knowledge of the dependency of the compression factor and resource allocation on the accuracy and latency functions, and how to tune these parameters to satisfy task requirements, constitute the task semantics, which is ultimately the information that enables the SEM-O-RAN framework. Notice that the accuracy and latency are not trivial functions of the slice allocation and compression factor. Specifically, the accuracy depends on the highly nonlinear output of a DNN, while the latency has a strong dependency on the radio technology and channel conditions between the RU and the UE, even when the slice allocation and the compression factor are given. For this reason, integrating a complex mathematical model to account for all of the great numbers of factors involved (e.g., Signal-to-Noise-Ratio (SNR), Modulation and Coding Scheme (MCS), carrier(s) frequency, to name a few) would be impractical. Instead, we consider a data-driven approach where the accuracy and latency functions can be constructed through a regression model, keeping the explicit dependencies of the accuracy $a_\tau(z) : (0, 1] \rightarrow \mathbb{R}^+$ and latency $l_\tau(z, s) : (0, 1] \times \mathbb{R}^{+^m} \rightarrow \mathbb{R}^+$ functions on the compression scaling factor and resource allocation, and assuming that those are given as part of the problem input. In our performance evaluation, we consider latency and accuracy as piecewise functions defined only for the discrete solution values allowed in our experiments.

TABLE 1: Table of Symbols

Symbol	Description
$\mathbb{C}$	Set of all application classes
c	Application class index
d	Mobile device index running an application
t	Task index requested by a device
(c, d, t)	t-th task requested by device d belonging to class c
τ	Generic task identified by the triplet (c, d, t)
$\mathbb{T}$	Set of all tasks τ of all devices from all classes
$\mathbb{K}$	Set of all Edge resource types
k	Edge resource type index
m	Total number of resource types
p_k	Price of the resource type k
x_τ	Admission of task τ
$s_{\tau k}$	Slice allocation of resource type k for τ
s_τ	Slice allocation vector $(s_{\tau 1}, ..., s_{\tau m})$ for τ
a_τ	Expected inference accuracy for task τ
l_τ	Expected E2E latency for task τ
A_c	Minimum accuracy tolerable for class c tasks
L_c	Maximum latency tolerable for class c tasks
z_τ	Compression scaling factor for task τ
S_k	Total capacity of type k resource

4.2 SF-ESP Problem Formulation

We consider the decision variables to be as follows:

- $\mathbf{x} = [x_\tau]$, defined as the task admission vector where the generic element, x_τ, is a binary variable indicating whether task τ is offloaded to the edge or not;
- $\mathbf{s} = [s_\tau] = [(s_{\tau 1}, ..., s_{\tau m})]$, i.e., the resource allocation matrix;
- $\mathbf{z} = [z_\tau]$ defined as the compression scaling factor vector.

Note that the data quality is maximum when $z_\tau = 1$ and decreases for lower values of z_τ. Consequently, the expected inference accuracy $a_\tau(z)$ is directly derived from z_τ, as it has no dependency on the resource allocation, while the expected latency $l_\tau(z, s)$ is a result of the choice of both z_τ and $\{s_{\tau k}\}_{\forall k}$. The problem formalization according to the system constraints and definitions is given by:

Semantic Flexible Edge Slicing Problem (SF-ESP)

$$\max_{\mathbf{x},\,\mathbf{s},\,\mathbf{z}} \quad \sum_{\tau \in \mathbb{T}} (O_\tau - \sum_{k}^{m} p_k s_{\tau k}) x_\tau \tag{1a}$$

$$\text{s.t.} \quad \sum_{\tau \in \mathbb{T}} s_{\tau k} x_\tau \leq S_k, \quad k = 1, \ldots, m, \tag{1b}$$

$$z_\tau \in (0, 1], \forall \tau \in \mathbb{T}, \tag{1c}$$

$$a_\tau(z_\tau) \geq A_c x_\tau, \forall \tau \in \mathbb{T}, c \in \mathbb{C}, \tag{1d}$$

$$l_\tau(z_\tau, s_\tau) x_\tau \leq L_c, \quad \forall \tau \in \mathbb{T}, c \in \mathbb{C}, \tag{1e}$$

$$x_\tau \in \{0, 1\}, \forall \tau \in \mathbb{T}. \tag{1f}$$

The objective function (1a) maximizes the profit associated with allocated tasks x_τ, by considering task offer O_τ, which constitutes the revenue for the task, and the cost of the task determined by summing the product of the price and the amount of allocated resources for the task, $p_k s_{\tau k}$, for each resource type.

Notice that the SF-ESP includes both integer and continuous variables, thus it belongs to the class of mixed integer nonlinear problems (MINLP). Proposition 1 below proves that the problem is NP-hard.

Proposition 1. *The SF-ESP is NP-hard.*

Proof. We prove the result by showing that the binary multidimensional Knapsack problem (0/1 d-KP), which is NP-hard [19], can be reduced to an instance of the SF-ESP in polynomial time. Let us assume that the compression factor is fixed to $z_\tau = 1, \forall \tau$, and the slice allocation $s_{\tau k}$ is given for every task and resource type. Then let us ignore the constraints on performance by making them always satisfied, i.e., by setting $A_1 = 0$ and $L_1 = \inf$. The problem now has only $\mathbf{x}$ as the decision variable and the value and weight of each task are known and constant. The problem thus is an instance of the 0/1 d-KP, whose statement is the following: given a set of items (tasks), each with a multidimensional weight (resource allocation) and a value (unused resources by their price), determine which items to include in a collection so that the total weight is less than or equal to a given limit (total resources) and the total value is maximized. We observe that the SF-ESP is a reduction of 0/1 d-KP that can be built in polynomial time. ■

The above proof also suggests that SF-ESP is a harder problem than 0/1 d-KP, as it is a combination of the 0/1 d-KP, and a variant of the strongly correlated knapsack with

variable weights and non-linear constraints. Even though an algorithm with $(1-\epsilon)$-approximation ratio exists for the 0/1 d-KP [20], for the strongly correlated knapsack with variable weights an algorithm with an acceptable approximation ratio is available only for the simpler case where constraints are linear [21, KLC2]. Thus, we provide a greedy heuristic algorithm for which, however, the existing results do not permit to obtain a non-trivial approximation ratio.

4.3 Greedy Algorithm for the SF-ESP

Given the NP-hardness of the SF-ESP, we propose a greedy heuristic to find a sub-optimal solution with low computational complexity, which is based on the primal effective gradient method of [22] for solving the 0/1 d-KP. The key idea of this method is to admit the tasks that have the highest effective gradient, a measure of the task's relative value according to a penalty vector that prioritizes the allocation of unused resources. By defining the occupied resources of type k as $o_k = \sum_{\tau \in T} s_{\tau k} x_\tau$, the effective gradient (EG) can be calculated as follows:

$$EG(s_\tau) = \begin{cases} \dfrac{(O_\tau - \sum_k^m p_k s_{\tau k})\sqrt{m}}{(\sum_k^m s_{\tau k}/S_k)}, & \text{if} \sum_k^m o_k = 0 \\ \dfrac{(O_\tau - \sum_k^m p_k s_{\tau k})\sqrt{\sum_k^m o_k^2}}{(\sum_k^m s_{\tau k} o_k/S_k)}, & \text{otherwise}. \end{cases} \tag{2}$$

Nevertheless, to calculate the gradient of a task and apply this method, we need to first fix the resource allocation of the tasks $s_{\tau k}$. To this end, we consider that the latency function $l_\tau(z_\tau, s_\tau)$ is monotonically increasing over the compression factor z_τ, and we derive the optimal task compression factor z_τ^* as the minimum that satisfies the accuracy requirement A_c from (1d):

$$z_\tau^* = \min_{z_\tau} z_\tau \ s.t. \ a_\tau(z_\tau) \geq A_c \tag{3}$$

Then, given z_τ^*, s_τ could be found by applying the same idea to (1e), i.e., by deriving the resource allocation that minimizes the resource cost for all tasks τ:

$$s_\tau^* = \arg\min_{s_\tau} \sum_k^m p_k s_{\tau k} \ s.t. \ l_\tau(z_\tau^*, s_\tau) \leq L_c \tag{4}$$

However, using Eq. 4 and admitting tasks in ascending order of their resource allocation may yield a non-optimal resource allocation, thus preventing the admission of a larger number of tasks. Indeed, a task may satisfy latency and accuracy constraints through several combinations of resource allocation, with the best being not the minimum, but the one that best balances the consumption of different types of resources, according to their current availability. As an example, if radio resources are scarce, we allocate fewer of them and compensate for the increased network latency by lowering the processing delay through increased compute resources.

Thus, SEM-O-RAN identifies the optimal resource allocation as the one that best balances the utilization of the different types of resources according to their availability, by maximizing, for every task τ, the EG function in [22] over the possible values of s_τ:

$$\begin{aligned} &s_\tau^* = \arg\max_{s_\tau} EG(s_\tau) \\ &s.t. \ l_\tau(z_\tau^*, s_\tau) \leq L_c, \ s_{\tau k} \leq S_k - \left(\sum_{\tau \in \mathbb{T}} s_{\tau k} x_\tau \right), \forall k \end{aligned} \tag{5}$$

To efficiently find a solution to (3) and (5), which depend on the definition of the accuracy and latency functions, it would be necessary to know the properties of such functions (e.g., monotonicity, convexity). Since we assume that accuracy and latency are generic functions, we solve the above equations through the enumeration of the resource allocation solution space. Once s_τ^* has been determined for any task τ, we can compute $EG(s_\tau^*)$, use this value to order the tasks, and select the task associated with the highest value of the effective gradient.

Algorithm 1 Greedy Algorithm for the SF-ESP

1: $T_c \leftarrow \mathbb{T}$ ▷ consider all tasks candidate for admission
2: **for all** $\tau \in \mathbb{T}$ **do**
3: $G_\tau \leftarrow 0, x_\tau \leftarrow 0, s_\tau \leftarrow (0, ..., 0), z_\tau \leftarrow 1$
4: **if** $\exists\, z_\tau^*$ **then** ▷ if minimum accuracy can be met
5: $z_\tau \leftarrow z_\tau^*$ ▷ save the optimal compression factor
6: **else**
7: $\mathbb{T}_c \leftarrow \mathbb{T}_c \setminus \tau$
8: **repeat**
9: **for** $k \leftarrow 1, m$ **do**
10: $o_k \leftarrow \sum_{\tau \in T} s_{\tau k} x_\tau$ ▷ occupied resources
11: **for all** $\tau \in \mathbb{T}_c$ **do**
12: **if** $\exists\, G_k \leftarrow \max_{s_{\tau k}} EG(s_\tau) \ s.t. \ s_{\tau k} \leq S_k - o_k, \forall k$ **then**
13: $s_\tau \leftarrow \arg\max_{s_\tau} EG(s_\tau) \ s.t. \ s_{\tau k} \leq S_k - o_k$
14: **else**
15: $\mathbb{T}_c \leftarrow \mathbb{T}_c \setminus \tau$
16: $\tau \leftarrow \tau \mid G_\tau = \max\{G_\tau\}_{\forall \tau}$
17: $x_\tau \leftarrow 1$ ▷ admit task whose gradient is maximum
18: $\mathbb{T}_c \leftarrow \mathbb{T}_c \setminus \tau$
19: **until** $T_c = \emptyset$
20: **return** $(x_\tau, s_\tau, z_\tau)_{\forall \tau \in \mathbb{T}}$
21: **function** EG(s_τ) ▷ calculate effective gradient
22: **if** $o_k = 0, \forall k$ **then** ▷ penalize resource usage equally
23: **return** $(O_\tau - \sum_k^m p_k s_{\tau k}) m^{\frac{1}{2}} / (\frac{\sum_k^m s_{\tau k}}{S_k})$
24: **else** ▷ penalize resource usage as per availability
25: **return** $(O_\tau - \sum_k^m p_k s_{\tau k})(\sum_k^m o_k^2)^{\frac{1}{2}} / (\frac{\sum_k^m s_{\tau k} o_k}{S_k})$

The preliminary step of the greedy algorithm (Alg. 1) is to (i) include all submitted tasks in the candidate task set (ln. 1), which contains the tasks that are considered feasible and worth of admission, and (ii) initialize the solution by setting the task admission vector and resource allocation matrix to zero, and the compression scaling factor to the unitary vector (ln. 3). Then, for each task, the optimal compression factor z^* is calculated according to its target accuracy (ln. 5), as per Eq. 3. An initial pruning of the candidate task set is performed by removing tasks whose target accuracy cannot be met for any compression factor (ln. 7).

The main loop of the algorithm (lines 8-19) examines the tasks in the candidate task set to find the most convenient

one to admit, based on the current resource occupation and until the set empties. First, the current resource occupation vector is updated (ln. 10). After that, the maximum effective gradient of each task in the candidate task set is calculated by exploring the feasible resource allocations (line 12), following Eq. 5. The effective gradient is calculated according to the function, defined in lines 21-25, in which the return value is computed differently whether resources are currently free (ln. 23) or not (25). If the maximum gradient is found, then the corresponding resource allocation for the examined task is saved (ln. 13), otherwise the task is discarded (line 15). Then, the task with the maximum value of maximum effective gradient is found (ln. 16), admitted by setting to one its corresponding value of the task admission vector (ln. 17) and therefore removed from the candidate task set (ln. 18). Finally, after the loop ends, the task admission vector, the resource allocation matrix, and the scaling factor vector are returned as the solution of the SF-ESP (line 20).

Defining $T = |\mathbb{T}|$ as the input size of the problem, we derive that the time complexity of Alg. 1 is $O(T^2 r^m)$, where the parameters r and m are the resource resolution, i.e., the number of discrete values allowed for the resource allocation, and the number of resource types, respectively. Instead, a brute-force search has exponential time complexity equal to $O(2^T r^m)$. The greedy algorithm, which has been coded in MATLAB and open-sourced to the research community, for a problem instance with $T = 50, r = 8, m = 2$, on average runs in 0.70 s on our test machine (see Sec. 5.1 for the technical specifications), using a single MATLAB worker.

5 Performance Evaluation

In this section, after describing the setup of the experiments (Sec. 5.1), we evaluate the performance of `SEM-O-RAN` through extensive numerical analysis (Sec. 5.2) and practical experiments on the Colosseum network emulator (Sec. 5.3).

5.1 Experiments Setup

Applications and datasets. As far as the DL services are concerned, we consider object detection and instance segmentation, which are state-of-the-art problems in computer vision. For the former, we consider (i) the widely-known COCO as dataset, which is a large-scale image database containing more than 200K labeled examples across 80 object classes [15]; (ii) the YOLOX classifier, which is based on the Modified CSP v5 as the backbone and has 54.2M parameters [14]. For the latter, we selected (i) the Cityscapes dataset, which contains pixel-level annotated video sequences of street scenes recorded in 50 different cities [17]; (ii) the BiSeNet v2 real-time classifier, which is based on a bilateral segmentation backbone network and has 14.8M parameters [16]. For performance evaluation purposes, we define a set of 10 computer vision tasks in Tab. 2.

Data degradation. In this work, we consider intentional data degradation, namely image compression, applied to save network bandwidth, and unintentional prior data degradation, such as the one caused by poor weather or illumination conditions. To apply compression, we use the *Pillow* python imaging library, which allows for the compression of an image by decreasing its resolution and saving it in JPEG format (JPEG quality: 75). To emulate the image quality degradation, we use the *imagecorruptions* python package, which provides a set of corruption effects at 5 different severity levels that can be applied to test the robustness of computer vision applications to unseen perturbations [23]. Of the several corruption effects available, for our experiments, we selected those in Tab. 3, for which an example is provided in Fig. 7.

TABLE 2: Multi-object detection (COCO) and instance segmentation (Cityscapes) applications

Application	Target Classes
COCO All	Entire set of classes (80) of COCO
COCO Urban	Bicycle, car, motorcycle, bus, truck, traffic light, stop sign, person
COCO Bags	Handbag, backpack, suitcase
COCO Animals	Bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe
COCO Person	Person
Cityscapes All	All evaluation classes (19) of Cityscapes
Cityscapes Vehicles	Car, truck, bus, train, motorcycle, bicycle
Cityscapes Objects	Pole, traffic light, traffic sign
Cityscapes Flat	Road, sidewalk
Cityscapes Person	Person

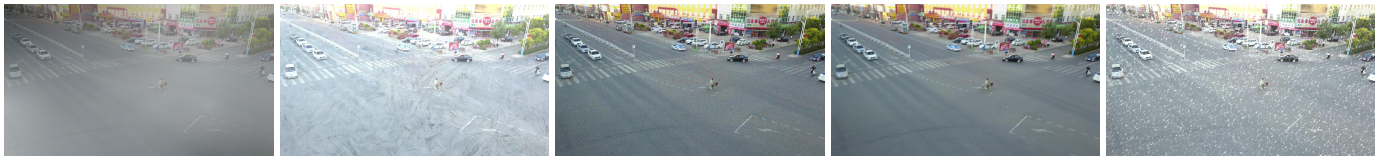
TABLE 3: Selected corruption effects and the motivation for which they have been chosen

Corruption	Motivation
Fog	Common weather during cooler months
Frost	Occasional winter weather
Gaussian noise	Digital images acquired with poor illumination
Motion blur	Movement/vibration during image acquisition
Snow	Common winter weather

To calculate the accuracy functions to be provided to the SF-ESP solver and the baselines, for each original dataset, we created a new dataset where all combinations of compression and corruptions are applied and tested. Starting from the original data size of 64 GB (52 GB COCO, 12 GB Cityscapes), a total of 332 GB has been used to store the augmented datasets.

Baselines. For comparison purposes, we consider the following baselines.

1) Sl-EDGE [5], the state-of-the-art algorithm for RAN edge slicing. We implement the Sl-EDGE algorithm as a multidimensional binary knapsack solver and we provide as input each task set to the optimal compression factor considering all the object classes, instead of just the classes specified by the application class, and the resource requirements as the minimum that allow for the satisfaction of the performance requirements (as for Eq. 4).
2) MinRes-SEM, an algorithm that considers the task semantics to set the optimal task compression factor but, instead of flexibly allocating resources, it allocates the minimum resources for each task, similarly to Sl-EDGE.
3) FlexRes-N-SEM, which implements flexible resource allocation according to Eq. 5 but does not



(a) Fog (b) Frost (c) Gaussian noise (d) Motion blur (e) Snow

Fig. 7: The selected corruption effects are applied on the same test image using the minimum severity (0).

account for the semantics, thus setting the compression factor considering all the object classes.

4) HighComp, which sets the compression factor of each task to 10%, so as to reach mAP of about 0.25 in the COCO dataset. This is a baseline that tries to compress tasks aggressively to minimize resource allocation.
5) HighRes, which statically allocates tasks 20% of the total resource capacity. This is a baseline that attempts to maximize the probability that admitted tasks will meet application constraints.

Numerical experiments configuration. To investigate the impact of our approach, we consider (i) different numbers (2 and 4) of edge/network resources (e.g., CPUs, GPUs, PRBs, etc.); (ii) different thresholds of accuracy ("low", "medium" and "high") and latency ("low", "high"). We define the accuracy thresholds A_c as 0.20, 0.35, and 0.55 mAP for object detection tasks, and 0.35, 0.50, and 0.70 mean Intersection over Union (mIoU) for instance segmentation tasks, while for latency threshold L_c we choose 0.2 s and 0.7 s. Tasks are equally distributed across the applications defined in Tab. 2. We empirically formulate a latency function l_τ that expresses the computational and network latency as a function of compression factor, resource allocation, and task generation rate. All numerical results are derived by repeating the experiments 64 times to obtain statistically meaningful results. Unless otherwise specified, all tasks have the same offer value, $O_\tau = \sum_k^m S_k$, and all resources have the same price, $p_k = 1/S_k$. To derive the numerical results, we used a 2x 32-cores AMD EPYC 7601 machine with 256 GB of DDR4 memory. The MATLAB SF-ESP heuristics has been executed in MATLAB R2022a leveraging the Parallel Toolbox with 64 parallel workers.

Prototype on Colosseum. We designed and developed a proof of concept of SEM-O-RAN on the Colosseum wireless network emulator [13], which emulates radio scenarios with up to 128 Standard Radio Nodes (SRNs). The channel between the SRNs is emulated through the Massive Channel Emulator (MCHEM), which processes radio signals through a series of Finite Impulse Response (FIR) filters. We used the open-source SCOPE framework [24], based on srsRAN [25], as a prototyping platform for NextG systems. Since SCOPE did not support slicing of uplink resources, we extended SCOPE to implement uplink slicing as well [2].

Fig. 8 shows a high-level overview of the SEM-O-RAN prototype. We reserve a set of 20 SRNs to implement the O-RAN network, with 1 SRN used to process received jobs of admitted tasks and to implement the base station using

2. https://github.com/corrado113/colosseum-scope/tree/ul_slicing

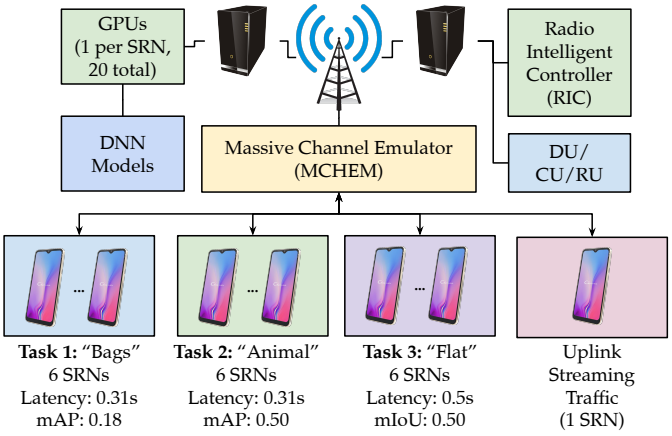


Fig. 8: Experimental setup on Colosseum.

srsRAN DU/CU/RU and the RIC, where we run the slice admission system and the solvers of the SF-ESP, implemented in MATLAB. Out of the remaining 19 SRNs, to emulate traffic separated from the mobile applications requiring RAN slices, we use one SRN to generate uplink streaming traffic with the *iperf* tool. The other 18 SRNs run the mobile terminals that run srsRAN User Equipment (srsUE) application to connect to the RAN and continuously send CV tasks according to the application classes in Fig. 8. The UEs belong to one of three VNOs, each with its own dedicated slice. As for the PHY, we utilize the standard SCOPE parameters, i.e., 10 MHz of bandwidth corresponding to 50 PRBs in total grouped in 17 RBGs. We assign the uplink streaming traffic 2 RBGs, thus, 15 RBGs are available for slicing as radio resources. Regarding computing resources, each of the 20 reserved SRNs is provided with 2x 12-cores Intel Xeon E5-2650 v4, 128 GB of DDR4 memory, and a Tesla K40m GPU. Thus, up to 20 GPUs can be utilized to accelerate the DNNs execution through the CUDA-compatible version of the Torch Python package. To simplify the scenario complexity, only GPUs and RBGs are considered limited and allocable by SEM-O-RAN. Conversely, the CPU time, memory, and disk space are assumed to be unlimited resources; it follows that we have m=2. To run the DL model inference on user images, we configure each UE to send CV tasks to the base station as periodic HTTP POST requests using the cURL client. The input images, fetched from the Colosseum Network Attached Storage (NAS) accessible by all the SRNs, are selected according to the compression factors decided by SEM-O-RAN and sent as multipart/form-data. To use all of the allocated GPUs in the slice, each SRN, besides running srsUE and sending tasks to the base station, executes an instance of the YOLOX and the BiSeNet v2 classifiers

embedded in a Flask HTTP server. To access the classifiers pool, the requests are first sent by the UEs to an Nginx server running at the base station through the RAN. Nginx acts as a frontend round-robin load balancer, so it forwards the requests to the backend classifiers to the other SRNs using the 10 GE Colosseum collaboration network. After an inference is completed, the classification result is returned traversing back the Nginx proxy through the collaboration network and to the requesting UE through the RAN. In this way, a task can effectively run on multiple GPUs by distributing inference frames according to the slicing decision.

5.2 Numerical Results

Task allocation results. Fig. 9 shows the number of allocated tasks by SEM-O-RAN and the baseline algorithms, as a function of the number of requested tasks when 2 and 4 types of edge/network resources are available. Fig. 9(a) shows that, in general, the performance of SEM-O-RAN is similar to that given by MinRes-SEM. Even when the requirements are medium accuracy and high latency, SEM-O-RAN allocates 20% more tasks than Sl-EDGE and FleRes-N-SEM, and 402% more tasks than HighRes, when 50 tasks are generated. On the other hand, when the accuracy requirements deviate from medium, we notice that SEM-O-RAN delivers significantly better performance than Sl-EDGE. Specifically, we notice that when high mAP/mIoU is required, only SEM-O-RAN and MinRes-SEM are able to allocate tasks that meet the requirements. Sl-EDGE does not allocate tasks since Sl-EDGE considers all the tasks as belonging to the "All" application, which can never reach the required mAP/mIoU of 0.55/0.70 (see the left side of Fig. 3). While HighComp and HighRes do allocate tasks, they will not meet the requirements. The reason is that HighComp and HighRes allocate tasks while being agnostic of the target latency and accuracy. The effect of joint semantic slicing and flexible resource allocation is even more evident in Fig. 9(b), where more types of edge/network resources are considered. In this case, SEM-O-RAN outperforms all the other schemes in all of the considered scenarios, especially when the number of tasks increases and the requirements become more stringent. Here, FlexRes-N-SEM, as opposed to Fig. 9(a), outperforms Sl-EDGE, proving the benefit of flexible resource allocation with more resource types.

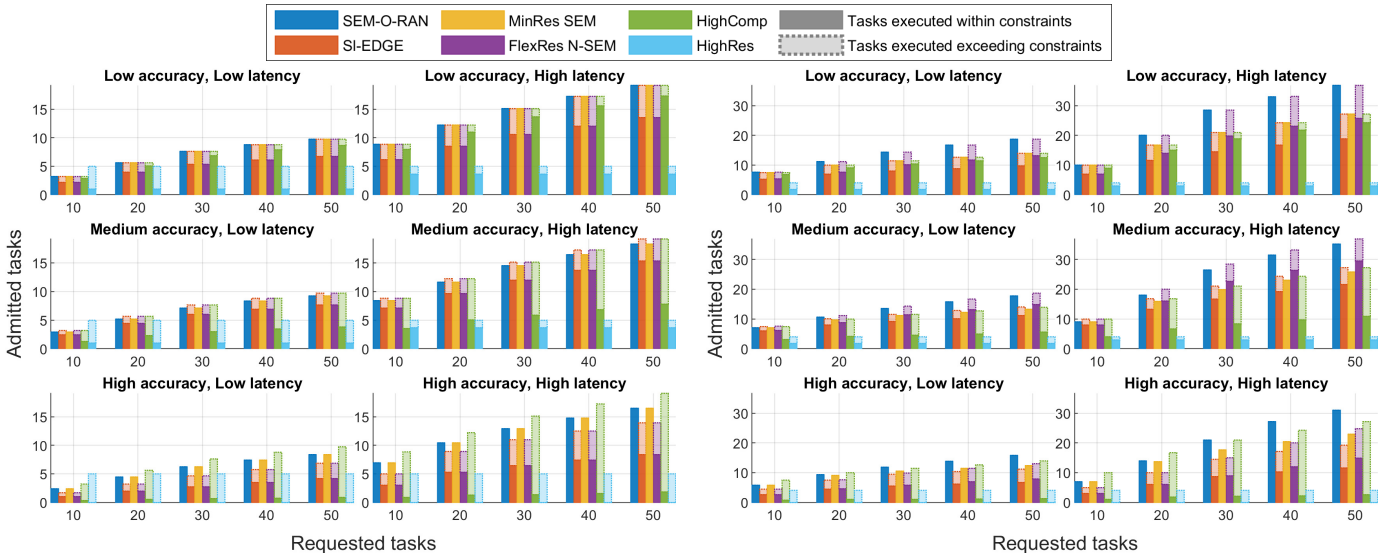
Overall, SEM-O-RAN allocates a maximum of 169% and an average of 72.0% more tasks than the existing state-of-the-art Sl-EDGE algorithm. Compared to Sl-EDGE, MinRes-SEM and FlexRes-N-SEM allocate an average of 57.2% and 9.1% more tasks respectively, thus suggesting that semantic data compression provides better performance gains than flexible resource allocation, particularly with few resource types, and that jointly applying the two SEM-O-RAN's key innovations provides better performance gains (+5,7%) than the sum of the individual gains contributions.

Data quality robustness. To make SEM-O-RAN robust to perturbation in the image quality, SEM-O-RAN's DQDM artificially corrupts datasets' images to learn the tolerable compression according to the application class. Here, we evaluate the importance of anticipating perturbations in the image quality, by testing SEM-O-RAN performance when tasks input data is degraded by artificial image corruption effects. Tab. 4 shows the comparison between SEM-O-RAN and Sl-EDGE, with and without the DQDM, which adds robustness to perturbations in the image quality in the presence of image degradation at different severity levels. The reported values are calculated by considering 50 requested tasks, which are affected by data degradation caused by an effect randomly selected from those in Tab. 3. Then, the results are averaged over the values collected from the experiments conducted using the parameters described in Sec. 5.1

SEM-O-RAN is always able to successfully execute all the allocated tasks, whose number decreases with the increase of the severity. Of the 19.43 average tasks successfully executed when no degradation is applied, only 8.60 are accepted and successfully executed when the degradation is maximum. If the DQDM is deactivated, SEM-O-RAN is no longer able to guarantee the successful execution of all the admitted tasks. Furthermore, the selected compression is often too aggressive, which causes a minimum of 0.27 successful tasks when the maximum degradation is applied. Sl-EDGE, when integrated with the DQDM, can accept a fair number of tasks. However, as seen in the task allocation results, it delivers worse results than SEM-O-RAN, since it does not consider the individual object classes: only 3.95 tasks are successfully executed at 100% severity. For the same reason, when the DQDM is disabled, Sl-EDGE is always able to successfully execute more tasks than SEM-O-RAN for all non-zero severity levels. To conclude, as SEM-O-RAN's capability of successfully meeting tasks' accuracy requirements is strongly affected by the fidelity of the accuracy function when working with real data, the DQDM is fundamental in a real-world scenario when tasks' input data may be affected by disturbances. On the contrary, the state-of-the-art Sl-EDGE, unaware of the semantics of the task, even with the DQDM, is not able to execute successfully the admitted tasks, thus leading to poor QoS and wasted edge resources.

Profits analysis. We now evaluate the impact of tasks' offers on the number of accepted tasks and the resulting profit. The revenue for the MNO is calculated as the difference between the profit obtained when tasks are executed within constraints and costs of the resources needed to accommodate the tasks, even when the tasks are not executed successfully. To do so, we compare SEM-O-RAN and Sl-EDGE in a for-profit configuration, where tasks' offer is not constant, and in a non-profit configuration, as they have been considered up to this moment. In the for-profit configuration, tasks' offers are chosen by randomly sampling from a discrete uniform distribution in the interval $[0, 2\sum_k^m S_k]$ such that tasks submitted for offloading have the same average offer in both configurations. The results are generated using the same configuration used for task allocation results to allow for a direct comparison. Then the resulting values are further averaged over the different numbers of edge resources, the thresholds of accuracy and latency, and the numbers of requested tasks, thus averaging over a total of 3,840 values.

We show in Tab. 5 that SEM-O-RAN for-profit, although accepting 12% fewer tasks, can increase profits by 13% over SEM-O-RAN non-profit, considering tasks with the same average offer. Sl-EDGE is worse than SEM-O-RAN in both of its configurations. Sl-EDGE for-profit only achieves 66%



(a) Numerical results with 2 types of edge/network resources

(b) Numerical results with 4 types of edge/network resources

Fig. 9: Numerical results: comparison between SEM-O-RAN and baselines.

TABLE 4: Data quality impact on admitted and successful tasks according to varying degradation severity levels

Tasks	Admitted				Successful			
Severity / Solution	0%	20%	60%	100%	0%	20%	60%	100%
SEM-O-RAN	19.43	16.02	11.54	8.60	19.43	16.02	11.53	8.60
SEM-O-RAN w/o DQDM	19.43	19.45	19.47	19.44	19.43	4.18	0.71	0.27
Sl-EDGE w/ DQDM	15.64	12.63	8.52	5.69	11.17	9.21	6.11	3.95
Sl-EDGE w/o DQDM	15.64	15.74	15.70	15.66	11.17	8.36	4.72	2.92

TABLE 5: Successful tasks and profits of SEM-O-RAN and Sl-EDGE, in their for- and non-profit configurations

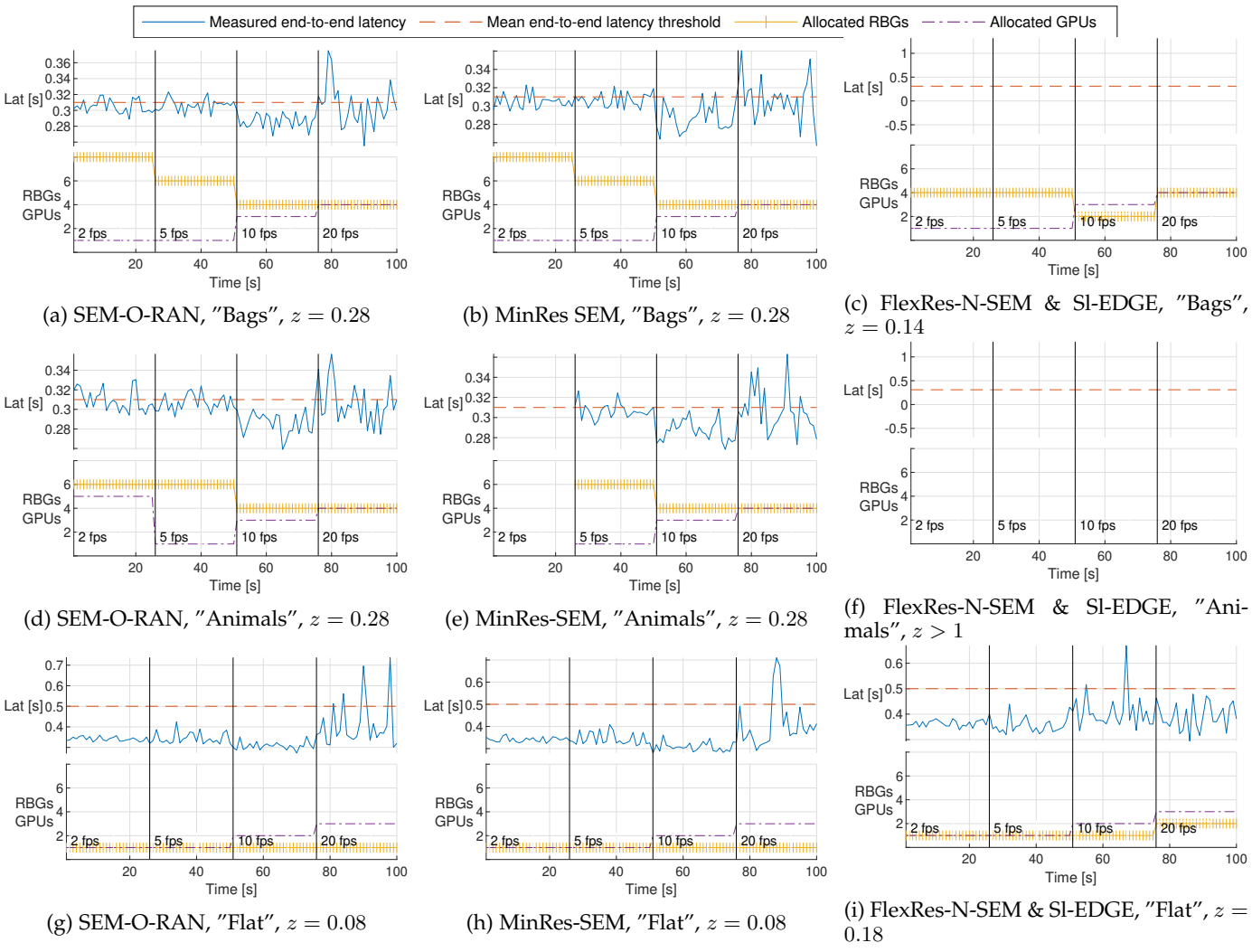
Solution	Successful tasks	Profit	Relative profit
SEM-O-RAN for-profit	11.31	46.37	152%
SEM-O-RAN non-profit	12.90	40.84	134%
Sl-EDGE for-profit	7.28	30.52	100%
Sl-EDGE non-profit	8.27	25.25	83%

of SEM-O-RAN for-profit profits, and this value decreases to 55% for Sl-EDGE non-profit. This difference is caused by the lower number of tasks executed successfully. Even when Sl-EDGE tries to maximize profits, it shows inferior performance than the non-profit version of SEM-O-RAN, proving again the valuable contributions of its key features. These results clearly demonstrate that the performance that our proposal offers is reflected in real economic value.

5.3 Experimental Results

Comparison of SEM-O-RAN and baselines. Fig. 10 shows our experimental results on Colosseum, in which we change the VNO slice requirements by updating the number of frames (i.e., jobs) per second (fps) that will be generated by each UE every 25 seconds, while latency and accuracy constraints are kept constant (values in Fig. 8). Whenever the requirements are updated, the SESM computes a new solution and enforces new slice configurations. Thus, we report the experimental end-to-end latency for each slice as a function of time, as well as the end-to-end latency threshold requirement for each task. To further investigate the advantage of flexible allocation and semantic slicing, we compare SEM-O-RAN to MinRes-SEM, FlexRes-N-SEM, and Sl-EDGE. Accordingly, we show the related output of the slicing algorithm in terms of RBGs (radio resources) and GPUs (computing resources). Notice that since FlexRes-N-SEM and Sl-EDGE take the same admission, resource allocation, and compression decisions, for them both we show only a single set of plots (Figs. 10(c),(f),(i)).

We see that SEM-O-RAN successfully allocates "Bags", "Animals", and "Flat". Interestingly, RBG allocation decreases as the fps request decreases because, for lower values of fps, the experienced latency increases as some time is spent for LTE uplink scheduling requests from the UEs [26]. With higher fps, the UE is able to use RBGs granted by the eNB to exchange traffic pertaining to multiple frames, thus leading to lower latency even if network utilization is higher. In the third and fourth periods, all three tasks are allocated by SEM-O-RAN. The impact of flexible resources is demonstrated in (e) where we see that MinRes-SEM does not allocate "Animals" in the first period. The reason is that SEM-O-RAN is balancing RBGs with GPUs, requesting 6 RBGs and 5 GPUs during the first period. Since MinRes-SEM would have requested 8 RBGs and 1 GPU, this would have led to 16 RBGs in total, which exceed the system capacity. Finally, from Figures 10(c), (f), and (i), it emerges that both FlexRes-N-SEM and Sl-EDGE, by not considering the semantics, perform worse than the former two approaches. By keeping in mind that the non-semantic solvers assume



(a) SEM-O-RAN, "Bags", $z = 0.28$

(b) MinRes SEM, "Bags", $z = 0.28$

(c) FlexRes-N-SEM & Sl-EDGE, "Bags", $z = 0.14$

(d) SEM-O-RAN, "Animals", $z = 0.28$

(e) MinRes-SEM, "Animals", $z = 0.28$

(f) FlexRes-N-SEM & Sl-EDGE, "Animals", $z > 1$

(g) SEM-O-RAN, "Flat", $z = 0.08$

(h) MinRes-SEM, "Flat", $z = 0.08$

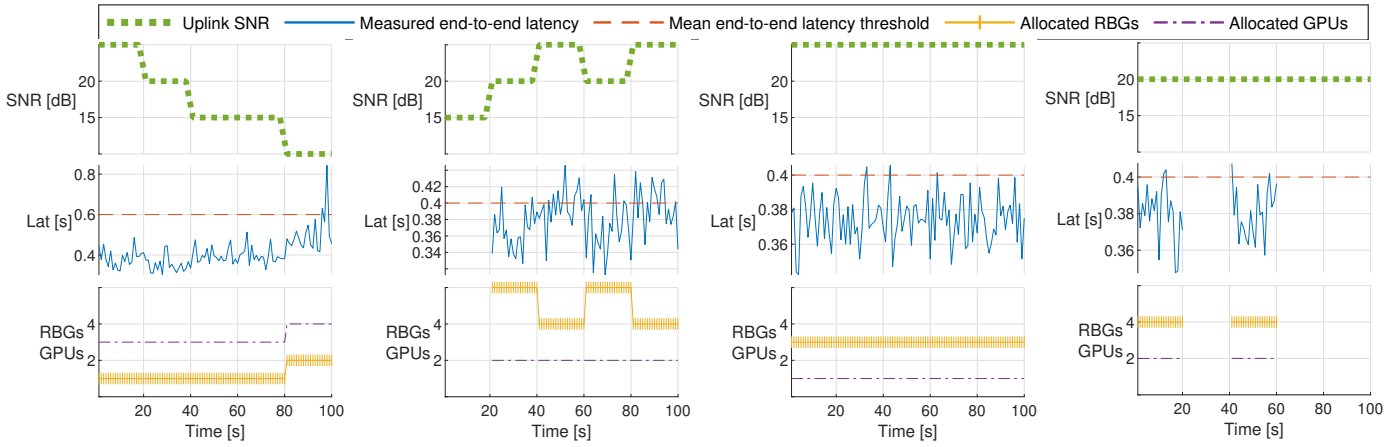
(i) FlexRes-N-SEM & Sl-EDGE, "Flat", $z = 0.18$

Fig. 10: Experimental results obtained through Colosseum, where we report the end-to-end latency as a function of time, as well as the end-to-end latency threshold requirement. We change the slice requirements by updating the number of generated frames per second (fps) by each UE every period of 25 seconds, and show the related output of the slicing algorithm in terms of RBGs (radio resources) and GPUs (computing resources). Below each plot, we report the chosen compression rate: note that in (c) $z = 0.14$ is a too aggressive compression factor, while in (f) $z > 1$ represents the case where an infeasible compression factor would be required.

that every task is of type "All", they will compress the tasks in "Bags" to 14% of their original size to maximize the number of tasks allocated. Conversely, SEM-O-RAN and MinRes-SEM compress "Bags" to 28%, which leads to successful allocation since the mAP constraint will be met. Even worse, FlexRes-N-SEM and Sl-EDGE will allocate resources for "Bags" but the tasks will fail because they will not meet the required mAP. Thus, even if these solvers save resources by compressing more, excessive compression prevents the achievement of the required mAP. As shown in Fig. 10(f), the "Animals" task is never admitted by FlexRes-N-SEM and Sl-EDGE, as they assume that an mAP of 0.5 can never be reached by "All", while SEM-O-RAN and MinRes-SEM, by accounting for the semantics, compress the tasks to the optimal level and can successfully admit it. As for "Flat", FlexRes-N-SEM and Sl-EDGE can always allocate it successfully but, by assuming the type as the more complex "All", they do not select the same aggressive compression factor that instead is chosen by SEM-O-RAN and MinRes-SEM (18% vs. 8%), at the cost of higher RBGs consumption in the latest period in Fig. 10(i). Finally, it is worth mentioning that the reason why FlexRes-N-Sem and Sl-EDGE show the same behavior, even if better performance would be expected thanks to flexible resource allocation, is that the limited amount of resource types is not sufficient to get any performance gain over the minimum resource allocation of Sl-EDGE.

TABLE 6: Task configurations for SEM-O-RAN evaluation with devices experiencing variable radio channel quality

Task	O_τ	A_c	L_c	FPS	Object class	Allowed actions
τ_1	20	0.2	0.6	20	Urban	z: [1, 0.28, 0.08] RBG: [1..6,8,10] GPU: [1..5]
τ_2	20	0.5	0.4	10	Urban	
τ_3	5	0.6	0.4	3	Person	
τ_4	5	0.6	0.4	3	Person	



(a) Task τ_1, $z = 0.08$ (b) Task τ_2, $z = 1$ (c) Task τ_3, $z = 1$ (d) Task τ_4, $z = 1$

Fig. 11: Experimental results obtained in Colosseum showing the latency achieved by admitted tasks according to their allocated radio and edge resources. Devices offloading the tasks experience variable channel quality, which affects the resources required to meet the latency constraints.

Radio channel quality impact on SEM-O-RAN. In a real-world scenario, mobile devices experience different channel conditions. To show how SEM-O-RAN behaves in this situation, we use Colosseum to emulate a radio scenario where the devices' radio channels experience varying values of SNR, and then we feed SEM-O-RAN with task latency functions formulated according to the radio channel status of the requesting device. Specifically, the radio scenarios emulated by the MCHEM are configured to apply a variable path loss to uplink and downlink RF signals between the base station and the UEs. Limiting the total available resources to 10 GPUs and 12 RBGs, we consider 4 object detection tasks whose characteristics are summarized in Tab. 6, where also the available actions are listed. Tasks configurations are chosen to achieve a good balance between required accuracy and fps. Moreover, τ_1 and τ_2, which are those with the highest offer, experience an SNR that varies each 20 s period, while τ_3 and τ_4 are offloaded over a stable radio link.

Fig. 11 shows the task latency and assigned resources when the tasks are admitted and, consequently, executed. Initially, all tasks are admitted except for task τ_2, even if it offers the highest value, because no resource allocation (among the allowed ones) can satisfy the latency requirement when the SNR is as low as 15 dB. During the second period, the SNR measured by the device requesting task τ_2 rises to 20 dB, which allows for the admission of the task with a large resource allocation. Because of this, task τ_4 can no longer be admitted and, hence, it is stopped. During the third period, the SNR relative to τ_2 rises to 25 dB, which allows SEM-O-RAN to respect the latency requirement with a smaller resource allocation. The freed resources can now be used by the resumed τ_4. The fourth period is similar to the second one, except now τ_1 is executed with a lower SNR, which, however, does not require more resources to be allocated. This does not hold in the last period, where more resources are needed to execute τ_1. Coincidentally, the larger allocation required by τ_1 is balanced by the smaller one required by τ_2, thus there is no need to stop τ_3 to free resources for higher offering tasks. The only difference between τ_3 and τ_4 is the higher SNR of the former, which allows for a lower resource allocation ((3,1) vs. (4,2)) and thus, as we have seen, a lower probability of being stopped to yield to higher offering tasks.

6 Related Work

RAN slicing has attracted significant attention over the last years [5], [6], [8]. Moreover, as the RAN gets softwarized, mobile edge computing (MEC) becomes crucial to address the ever-stringent latency demands of mobile applications [27], [28]. We refer the interested reader to the surveys [29], [30].

Specific to the slicing of edge resources, Van Huynh *et al.* [31] present a mechanism for slicing of computation, networking, and storage through a deep dueling neural network that provides slices admission while avoiding over-provisioning and maximizing the VNO's reward. However, the authors in [31] do not focus on how to partition the MEC resources and only focus on admission control. Conversely, Ndikumana *et al.* [32] consider the allocation of heterogeneous resources for MEC task offloading, while in [33] Liu *et al.* propose a framework for MEC-enabled wireless networks called DIRECT, which however does not consider the case when MEC and networking resources are on the same edge node. Moreover, these frameworks are not O-RAN compatible, which is instead one of the primary goals of this paper.

Closely related to network slicing research, several studies on Service Function Chaining (SFC) have proposed solutions to the Virtual Network Function (VNF) placement and Network Function Virtualization (NFV) resource allocation problems. Early works focused mainly on Telecom Service Provider's and optical networks, whereas 5G and wireless networks have received more attention lately [34]. In [35], the authors formalize a wireless VNF placement problem and propose a heuristic algorithm to solve it considering the function requirements of wireless network bandwidth and computing resources. Although flexible resource allocation has been considered in the context of VNF [36]–[38], existing

formulations do not consider application semantics, and, in general, cannot be easily applied to address edge task offloading problems.

So far, most of the research focus in O-RAN has been on designing algorithms for RAN control and optimization. Bonati *et al.* [12] have developed an xApp running deep reinforcement learning (DRL) agents to select the best-performing scheduling policy for each RAN slice. In our work, we do not select scheduling policies but instead focus on RAN slicing. D'Oro *et al.* [11] propose an orchestration mechanism to select the optimal DL models and execution location for each model complying with timescale requirements, resource, and data availability. Conversely, we focus on properly slicing MEC resources for timely execution of CV-based DL models under strict accuracy constraints.

Task offloading at the edge has been considered in several recent works. In [39], the correlation between statistical QoS requirements, i.e., the completion of a task within the deadline with a certain probability and task offloading strategies, is quantified. In our work, we consider the average latency to derive the latency functions but, because of the general definition, other measures (e.g., maximum or percentiles) can be used as well to satisfy statistical requirements. Focusing more on task offloading revenue, the authors of [40] propose a minimum-cost-maximum-flow graph algorithm to minimize edge power consumption while maximizing offloaded task rewards. In [41], an alternative approach to MNOs revenues maximization is proposed through a pricing model designed for balancing the prices of required edge resources and task arrival rate. Adaptive quality optimization of CV tasks is considered in [42], which however focuses on the compression factor selection and task placement problem ignoring the resource capacity constraints of the computing platforms.

The closest work to ours is Sl-EDGE [5], a MEC slicing framework that allows network operators to instantiate heterogeneous edge slices. The key limitation of Sl-EDGE is that it considers neither DL semantics nor flexible resource allocation, which are instead the core advantages of our approach. Indeed, we show that our solution schedules up to 169% more tasks than Sl-EDGE and 52% higher profits.

Finally, we mention that a preliminary version of this work can be found in [1], where we propose a simpler SEM-O-RAN framework architecture and objective function; we present a minimal numerical analysis, not considering the effects of prior data degradation and the economic aspect of our proposal, and a limited experimental performance evaluation, which disregards radio channel effects on the communications of mobile devices.

7 Conclusions

Our paper proposes SEM-O-RAN, the first semantic slicing framework for task edge offloading in NextG O-RAN mobile networks. SEM-O-RAN achieves great performance by optimizing network usage through semantic adaptive compression. Furthermore, unlike existing methods, SEM-O-RAN does not treat tasks as monolithic entities, but rather allocates radio and computational resources in a flexible manner to maximize the number of admitted tasks. Considering these two key concepts, we mathematically formulated the SF-ESP, which we proved to be NP-hard. In light of the problem complexity, we proposed an efficient greedy heuristic algorithm to solve it. To evaluate the performance of SEM-O-RAN, we conducted extensive numerical analyses and compared it to several baseline algorithms, including the state-of-the-art scheme [5]. Our results demonstrate that SEM-O-RAN can improve the number of allocated tasks by up to 169%, while still satisfying accuracy and delay constraints. Additionally, we implemented a prototype of SEM-O-RAN using the Colosseum network emulator and the SCOPE framework for NextG RAN [24], showing the feasibility of our proposal and demonstrating that SEM-O-RAN can be seamlessly integrated into a NextG system. We demonstrated the economic value of our approach showing that SEM-O-RAN can improve the revenues of MNOs by 52% compared to the state of the art. We believe that our semantics-based approach can serve as a foundation for future research on the utilization of application-level features in the design and optimization of wireless networks, beyond the results presented in this paper.

Acknowledgment of Support and Disclaimer

This work is funded in part by the National Science Foundation (NSF) grant CNS-2134973, CNS-2120447 and ECCS-2229472, by the Air Force Office of Scientific Research under contract number FA9550-23-1-0261, by the Office of Naval Research under award number N00014-23-1-2221, by an effort sponsored by the U.S. Government under Other Transaction number FA8750-21-9-9000 between SOSSEC, Inc. and the Government, by the European Union's NextGenerationEU instrument, under the Italian National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3, enlarged partnership "Telecommunications of the Future" (PE00000001), program "RESTART", and by the European Commission through Grant No. 101095890 (Horizon Europe SNS JU PREDICT-6G project) and Grant No. 101095363 (Horizon Europe SNS JU ADROIT6G project).

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF, the Air Force Research Laboratory, the U.S. Government, or SOSSEC, Inc.

References

[1] C. Puligheddu, J. Ashdown, C. F. Chiasserini, and F. Restuccia, "SEM-O-RAN: Semantic and Flexible O-RAN Slicing for NextG Edge-Assisted Mobile Systems," in *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, 2023.

[2] H. Ye, L. Liang, G. Ye Li, J. Kim, L. Lu, and M. Wu, "Machine Learning for Vehicular Networks: Recent Advances and Application Examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, 2018.

[3] R. Ravindran, M. J. Santora, and M. M. Jamali, "Multi-object Detection and Tracking, based on DNN, for Autonomous Vehicles: A Review," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 5668–5677, 2020.

[4] X. Li, A. Garcia-Saavedra, X. Costa-Perez, C. J. Bernardos, C. Guimarães, K. Antevski, J. Mangues-Bafalluy, J. Baranda, E. Zeydan, D. Corujo, *et al.*, "5Growth: An End-to-End Service Platform for Automated Deployment and Management of Vertical Services over 5G Networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 84–90, 2021.

[5] S. D'Oro, L. Bonati, F. Restuccia, M. Polese, M. Zorzi, and T. Melodia, "Sl-EDGE: Network Slicing at the Edge," in *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, pp. 1–10, 2020.

[6] S. Mandelli, M. Andrews, S. Borst, and S. Klein, "Satisfying Network Slicing Constraints via 5G MAC Scheduling," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 2332–2340, IEEE, 2019.

[7] V. Mancuso, P. Castagno, M. Sereno, and M. A. Marsan, "Slicing Cell Resources: The Case of HTC and MTC Coexistence," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 667–675, IEEE, 2019.

[8] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia, "The Slice is Served: Enforcing Radio Access Network Slicing in Virtualized 5G Systems," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 442–450, IEEE, 2019.

[9] G. Garcia-Aviles, M. Gramaglia, P. Serrano, and A. Banchs, "POSENS: A Practical Open Source Solution for End-to-End Network Slicing," *IEEE Wireless Communications*, vol. 25, no. 5, pp. 30–37, 2018.

[10] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *arXiv preprint arXiv:2202.01032*, 2022.

[11] S. D'Oro, L. Bonati, M. Polese, and T. Melodia, "OrchestRAN: Network Automation through Orchestrated Intelligence in the Open RAN," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, May 2022.

[12] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, pp. 21–27, October 2021.

[13] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder, *et al.*, "Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 105–113, IEEE, 2021.

[14] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 740–755, Springer, 2014.

[16] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "BiSeNet V2: Bilateral Network with Guided Aggregation for Real-Time Semantic Segmentation," *International Journal of Computer Vision*, vol. 129, pp. 3051–3068, Nov 2021.

[17] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead," *Computer Networks*, vol. 182, pp. 1–28, December 2020.

[19] H. Kellerer, U. Pferschy, and D. Pisinger, *Multidimensional Knapsack Problems*, pp. 235–283. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[20] A. Frieze and M. Clarke, "Approximation Algorithms for the m-Dimensional 0–1 Knapsack Problem: Worst-Case and Probabilistic Analyses," *European Journal of Operational Research*, vol. 15, no. 1, pp. 100–109, 1984.

[21] K. Nip, Z. Wang, and Z. Wang, "Knapsack with Variable Weights Satisfying Linear Constraints," vol. 69, p. 713–725, nov 2017.

[22] Y. Toyoda, "A Simplified Algorithm for Obtaining Approximate Solutions to Zero-One Programming Problems," *Management Science*, vol. 21, no. 12, pp. 1417–1427, 1975.

[23] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, "Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming," *arXiv preprint arXiv:1907.07484*, 2019.

[24] L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "SCOPE: An Open and Softwarized Prototyping Platform for NextG Systems," in *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 415–426, 2021.

[25] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "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*, WiNTECH '16, p. 25–32, 2016.

[26] G. Pocovi, I. Thibault, T. Kolding, M. Lauridsen, R. Canolli, N. Edwards, and D. Lister, "On the Suitability of LTE Air Interface for Reliable Low-Latency Applications," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2019.

[27] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation Offloading in Multi-Access Edge Computing Using a Deep Sequential Model Based on Reinforcement Learning," *IEEE Communications Magazine*, vol. 57, pp. 64–69, May 2019.

[28] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task Offloading in Vehicular Edge Computing Networks: A Load-Balancing Solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2019.

[29] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[30] S. Wijethilaka and M. Liyanage, "Survey on Network Slicing for Internet of Things Realization in 5G Networks," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 957–994, 2021.

[31] N. Van Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and Fast Real-Time Resource Slicing with Deep Dueling Neural Networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455–1470, 2019.

[32] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint Communication, Computation, Caching, and Control in Big Data Multi-Access Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1359–1374, 2019.

[33] Q. Liu and T. Han, "DIRECT: Distributed Cross-Domain Resource Orchestration in Cellular Edge Computing," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 181–190, ACM, 2019.

[34] J. Gil Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[35] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling Wireless Virtual Networks Functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 240–252, 2016.

[36] M. Golkarifard, C. F. Chiasserini, F. Malandrino, and A. Movaghar, "Dynamic VNF Placement, Resource Allocation and Traffic Routing in 5G," *Computer Networks*, vol. 188, p. 107830, 2021.

[37] J. Martín-Pérez, F. Malandrino, C. F. Chiasserini, M. Groshev, and C. J. Bernardos, "KPI Guarantees in Network Slicing," *IEEE/ACM Transactions on Networking*, vol. 30, no. 2, pp. 655–668, 2021.

[38] A. Pentelas, G. Papathanail, I. Fotoglou, and P. Papadimitriou, "Network Service Embedding Across Multiple Resource Dimensions," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 209–223, 2021.

[39] Q. Li, S. Wang, A. Zhou, X. Ma, F. Yang, and A. X. Liu, "QoS Driven Task Offloading With Statistical Guarantee in Mobile Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 278–290, 2022.

[40] M. Song, Y. Lee, and K. Kim, "Reward-Oriented Task Offloading Under Limited Edge Server Power for Multiaccess Edge Computing," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13425–13438, 2021.

[41] M. Mukherjee, V. Kumar, Q. Zhang, C. X. Mavromoustakis, and R. Matam, "Optimal Pricing for Offloaded Hard- and Soft-Deadline Tasks in Edge Computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9829–9839, 2022.

[42] A. Toma, J. Wenner, J. E. Lenssen, and J.-J. Chen, "Adaptive quality optimization of computer vision tasks in resource-constrained devices using edge computing," in *2019 19th IEEE/ACM Interna-*

tional Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 469–477, 2019.

Corrado Puligheddu [M'20] is an Assistant Professor at Politecnico di Torino, Turin, Italy, since 2023. His research interests include 5G networks, Open RAN and Machine Learning. Puligheddu received his Ph.D. in Electrical, Electronics and Communication Engineering from Politecnico di Torino in 2022. He obtained the B.S and M.S. in Computer Engineering from the same institution in 2017 and 2019 respectively.

Jonathan Ashdown [M'13, SM'21] received the B.S., M.S., and Ph.D. degrees from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2006, 2008, and 2012, respectively, all in electrical engineering. He received the Best Unclassified Paper Award at the IEEE Military Communications Conference in 2012. From 2012 to 2015, he worked as an electronics engineer with the Department of Defense (DoD), Naval Information Warfare Center (NIWC) Atlantic, Charleston, SC, USA where he was involved in several basic and applied research projects for the U.S. Navy, mainly in the area of software defined radio. In 2015, he transferred within DoD to the Information Directorate of the Air Force Research Laboratory (AFRL), Rome, NY, USA, where he serves as a senior electronics engineer and is involved in the research and development of advanced emerging communications and networking technologies for the U.S. Air Force.

Carla Fabiana Chiasserini [F'18] is a Professor at Politecnico di Torino, Italy. She worked as a visiting scholar and researcher at UCSD from 1998 till 2003. She was also a Visiting Professor at the Monash University (Australia) in 2012 and 2016, and at the Technical University in Berlin (Germany) in 2021 and 2022. Carla serves as EiC of Computer Communications, and as Associate EiC for the IEEE Transactions on Network Science and Engineering.

Francesco Restuccia [M'16, SM'21] is an Assistant Professor in the Department of Electrical and Computer Engineering at Northeastern University. He received his Ph.D. in Computer Science from Missouri University of Science and Technology in 2016, and his B.S. and M.S. in Computer Engineering with highest honors from the University of Pisa, Italy in 2009 and 2011, respectively. His research interests lie in the design and experimental evaluation of next-generation edge-assisted data-driven wireless systems. Prof. Restuccia's research is funded by several grants from the US National Science Foundation and the Department of Defense. He received the Office of Naval Research Young Investigator Award, the Air Force Office of Scientific Research Young Investigator Award and the Mario Gerla Award in Computer Science, as well as best paper awards at IEEE INFOCOM and IEEE WOWMOM. Prof. Restuccia has published over 60 papers in top-tier venues in computer networking, as well as co-authoring 16+ U.S. patents and three book chapters. He regularly serves as a TPC member and reviewer for several top-tier ACM and IEEE conferences and journals.