SOAR: Semantic Multi-User MIMO Communications for Reliable Wireless Edge Computing

Sharon L.G. Contreras *, Khandaker Foysal Haque †, Francesco Restuccia †, and Marco Levorato *
*University of California, Irvine, †Northeastern University

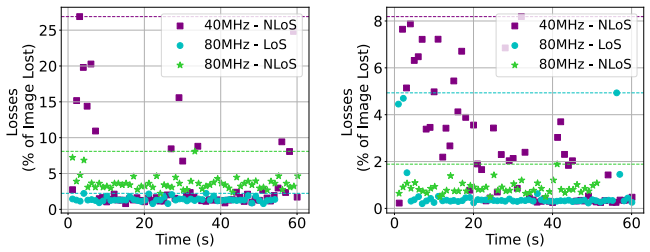
***Abstract*—Ensuring reliability in Mobile Edge Computing Systems (MECS) is challenging due to wireless channel fluctuations, which affect packet delivery rate and real-time task performance. Existing methods, such as packet replication strategies and deep neural network (DNN) splitting, often lack task-specific adaptability, oversimplifying the effects of channel dynamics in its performance. To address this issues, we propose Semantic Offloading through Reliability (SOAR), a task-oriented multi-user MIMO framework for wireless edge computing executing vision tasks, e.g. object detection and image classification. SOAR pipeline uses distributional deep reinforcement learning (DDRL) agents with a multi-branched context-aware neural network. Two neural gates analyze onboard features to identify the context and contextual features, enabling a DDRL agent to optimize resource usage and task-specific packet-loss objectives. We evaluated SOAR in real-world vehicular system under line-of-sight (LoS) and non-line-of-sight (NLoS) propagation scenarios, SOAR reduces resource utilization by 35–40% compared to fixed antenna configuration benchmarks [1].**

***Index Terms*—Mobile Edge Computing Systems, Task Offloading, Deep Reinforcement Learning**

I. Introduction

Robotic applications heavily rely on machine learning models for their operations, typically in the form of complex deep neural networks (DNN). For example, autonomous navigation incorporate computer vision tasks such as image segmentation, object detection, and classification, which are computationally intensive and delay sensitive [2]. Due to the limited computing capabilities that are usually available to a mobile edge device (MED), in mobile edge computing systems (MECS), the tasks are offloaded to a remote device, the edge server (ES) [3], [4]. Although task offloading can decrease MEDs' energy consumption, as well as the execution time, the offloading process requires the transmission of the input data to the ES and the response of the inference to be sent back to the MED. The large size of the computer vision input data and their timely and reliable transfer over wireless channels pose considerable challenges to communication and network systems.

The above issue is further exacerbated in applications where MEDs operate in challenging propagation environments, such as ground rovers that autonomously process visual information to navigate in urban and indoor settings. In such scenarios, unreliable communications degrade core performance metrics, and make data rate erratic [5], consequently gravely affecting task performance. We note how these applications impose fine-grain performance constraints connected to individual data points (e.g., a bound on per-image latency or quality) rather than requirements on average performance metrics such as throughput. The current literature adopts two main approaches in MECS to address this challenge. Both are based on limiting the amount of data to be transmitted from MED to the ES.



(a) 1x1 streams (b) 4x4 streams

Fig. 1: Packet loss using 40MHz and 80MHz within two types of outdoor scenarios: with and without obstacles.

Approach one: Partial offloading [6], [7] is one of the approaches to maintain the required quality of service (QoS) in challenging channel conditions. Partial offloading involves dividing the data processing and task execution between ESs and the MEDs. In poor channel conditions, the MEDs choose to partially rather than fully offload the data to the ES.

Approach two: Some solutions propose to compress input data and or intermediate features [8]–[10], for instance in the context of "split" convolutional neural network (CNN) architectures. This results in a smaller amount of data transmitted over the wireless channel, and thus a reduced sensitivity to poor or highly erratic channel conditions.

The above approaches focus on the modification of the computing pipeline, leaving the transmission layer unaltered. Conversely, many communication centered approaches focus on traditional traffic and network-based perspectives ignoring the important challenges and opportunities connected to the semantic structure of the data and computing tasks.

Focusing on modern multi-user multiple-input multiple-output (MU-MIMO) communication technologies, our rationale takes as a starting point fine-grain characteristics of task-level performance. We first illustrate the characteristics of the problem at hand with some preliminary tests on a real-world system (see the detailed description in Section IV). Fig. 1 shows patterns of per-image packet loss for different transmission configurations (number of parallel data streams and channel bandwidth) both in line of sight (LoS) and non-line of sight (NLoS) settings. The erratic nature of these fine-grain patterns is apparent, as well as their dependency on the specific environ-

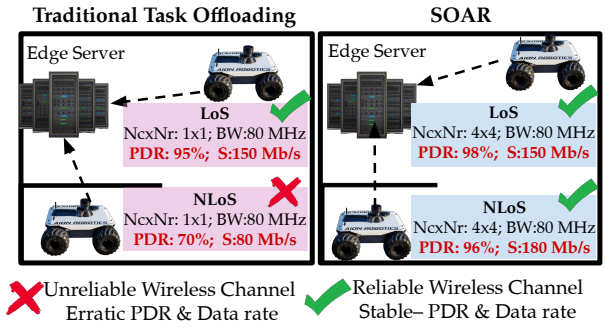


Fig. 2: SOAR vs traditional task offloading approaches. Here, N_c: no. of the transmit antenna, N_r: no. of the receive stream, BW: bandwidth, PDR: packet delivery ratio, and S: data rate of the wireless channel.

ment and transmission configuration. On the one hand, larger resource usage – a larger number of streams and bandwidth – can reduce both average and variance of per-image (as illustrated in Fig. 2).

In the context of perception for robotic platforms, our overarching goal is to create a context and task-aware predictive logic capable to dynamically and semantically control MU-MIMO resource usage to achieve task specific objectives. The proposed framework – SOAR– reasons at the granularity of individual data points to be delivered to the ES. Specifically, SOAR selects transmission configurations predicted to achieve a per-image packet loss, corresponding to a task specific performance, in a window of future images while minimizing resource usage based on observable features drawn from system components such as telemetry, network interface and application.

The engine of SOAR is an innovative context-aware distributional deep reinforcement learning (DDRL) agent [11] that embeds a multi-branched neural network. First, a neural selector processes general features to detect the context and select a context-specific model (branch) in a pre-trained set. The branch processes a specialized set of features to output the distribution of a value function based on a composite performance/resource usage cost function. We remark how traditional DRL agents focus on the estimation of the expectation of a value function conditioned on an action, without consideration for its variance. In the context of mission-critical systems, this may lead to catastrophic outcomes. Conversely, DDRL focuses on the distribution of the value function given the action, thus allowing more sophisticated reasoning – together with more robust learning.

Summary of Novel Contributions

• In the context of robotic applications offloading computer vision tasks over MU-MIMO Wi-Fi (IEEE 802.11ac) channels, we perform an in-depth analysis of the relationship between system-level features of MECS from logical blocks such as application, network, and telemetry, the characteristics of image transfer at the fine-grain temporal scale, and task performance.

• We develop a novel semantic, predictive and context-aware controller – SOAR– that dynamically configures MU-MIMO transmission parameters to meet packet loss ratio objectives set at the granularity of individual image/task while minimizing channel resource usage. The controller is task aware, meaning that the packet loss threshold is set to the specific computer vision task. To make an example, as demonstrated in our results, a satisfactory performance in image classification can be achieved with a larger loss of information compared to semantic segmentation, where individual pixels are classified. The SOAR framework is context-specific as the controller and its input features are dynamically adapted to the operating context – e.g., a rover navigating in a line-of-sight (rural open space) or non line-of-sight (urban with buildings) environments – to boost control effectiveness.

• At the core of the SOAR framework is an innovative multi-branch distributional deep reinforcement learning agent. This design uses context-specific agents trained on specialized features to maximize performance in their environments. A lightweight neural selector extracts context from a small feature set to select the appropriate agent. This approach significantly improves per-image packet loss prediction compared to non-specific predictors. By evaluating reward distributions rather than expectations, these agents enhance controller robustness, offering a novel approach for communication systems.

• We perform an extensive data collection campaign to inform both the construction and the evaluation of SOAR with a Linux workstation as ES and a rover as MED in 2 different environments with 4 different communication setups representing simultaneous user streams. We consider data offloading at different frames per second (FPS) – 1, 5, 15, and 30 FPS for each of the setups. Additionally, we develop a framework to synchronously capture the features from network, application and MED telemetry blocks.

• We present results demonstrating SOAR's context adaptation to real-world dynamics based on application, network, and telemetry features. SOAR's distributional deep reinforcement learning (DDRL) context agent achieves state-of-the-art performance for instance segmentation and image classification tasks, reducing resource utilization by 35% in NLoS and 40% in LoS, within a 20ms offloading deadline. We evaluate short-sighted (myopic) and long-term DDRL policies, finding that myopic limitations more significantly affect performance in NLoS compared to LoS propagation contexts.

II. Related Work and Motivation

Task offloading in MEC systems is hindered by volatile wireless channels and time-varying workloads. To address this, prior work has explored resource allocation frameworks, distributed computing, and dynamic reconfiguration of computing pipelines between MEDs and ESs [12]–[14].

Most recent literature in MECS focuses on optimizing key metrics such as energy consumption, bandwidth usage, and computational resources, primarily under reliable channel conditions. For instance, Guo et al. [15] and Fresa et al. [16] propose frameworks to maximize task accuracy in full offloading scenarios. Meanwhile, Matsubara et al. [17], Lakew et al.

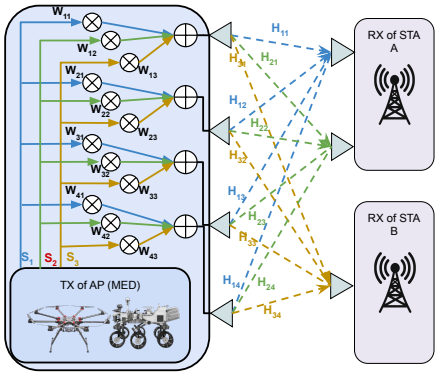


Fig. 3: Example of a 4×3 MU-MIMO system.

[18], and others explore partial offloading to balance end-to-end delay, energy consumption, and task performance through split computing, early exit, and data compression techniques.

Dynamic pipeline configuration is studied by Callegaro et al. [19], who leverage telemetry, network, and application data to predict edge servers (ESs) that maximize task execution success. Their model-free Markov Decision Process estimates the optimal ES for reliable offloading.

The effects of wireless channel conditions on MECS are analyzed by Ilhan et al. [20] and Ozer et al. [21]. Ilhan et al. examine LTE channel noise impacts on image segmentation, while Ozer et al. propose denoising to enhance transmitted image quality for edge-based computer vision task.

We then note how most task-aware algorithms mostly focus on the computing workload, while communication-oriented contributions are primarily centered on traditional traffic-based reasoning. In stark contrast, our contribution uniquely connects the semantic of the tasks and their fine-grain performance to the dynamic adaptation of communication parameters.

III. MU-MIMO in MECS

A. A Walkthrough of MU-MIMO Wi-Fi Systems

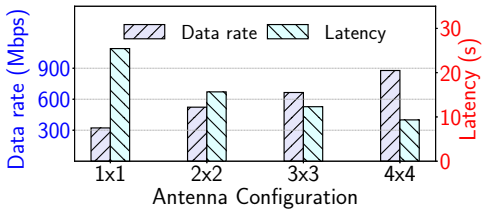
Wi-Fi technology provides reliable task offloading [22]. The MED's integrated Access Point (AP) enables simultaneous communication with multiple stations stations (STAs) connected to ESs, efficiently leveraging beamforming for parallel transmission streams . Using orthogonal frequency-division multiplexing (OFDM), Wi-Fi transmits across K partially overlapping sub-channels. Input bits are organized into OFDM samples, which are grouped into OFDM symbols $a = [a_{-k/2}, ..., a_{(k/2)1}]$ [23], [24], allowing simultaneous transmission of digitally modulated symbols across K sub-channels.

$$s_{tx}(t) = e^{j2\pi f_c t} \sum_{-K/2}^{(K/2)-1} a_k e^{j2\pi kt/T} \quad (1)$$

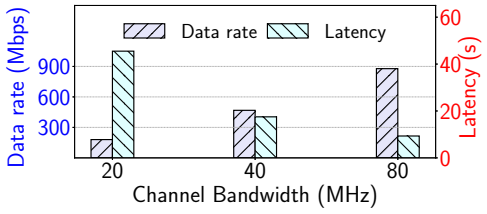
Equation (1) expresses the transmitted signal whereas f_c is the carrier frequency and $T = 1/(\Delta f)$ is the symbol time with Δf being the sub-channel spacing. To improve the signal quality, the transmitter performs beamforming to steer the transmission streams toward the intended receiver. To perform beamforming, multiple signal streams are combined at the transmitter by steering the weights $\mathbf{W}$. $\mathbf{W}$ is derived from the channel frequency response (CFR) matrix $\mathbf{H}$, which is estimated for every orthogonal frequency-division multiplexing (OFDM) sub-channels. The obtained $\mathbf{H}$ is of dimension $\mathbf{K} \times \mathbf{M} \times \mathbf{N}$, where $\mathbf{M}$ and $\mathbf{N}$ are the number of transmit and receive antennas. At the receiver (beamformee), the beamformed signals are retrieved from the fact that $[\mathbf{H}]_{\bar{l},\bar{i}} \times [\mathbf{W}]_{l,i} = 0$ where $\bar{l} \neq l$ or $\bar{i} \neq i$. Fig. 3 presents a 4×3 MU-MIMO system where Access Point (AP) (beamformer) with 4 antennas transmitting to two different STAs: STA A and STA B, having two and one receive antennas enabled respectively. The transmission signal $s_{tx}(t)$ from the beamformer bounces off different physical objects of the environment, and $\mathbf{P}$ different copies of the signal are received by the beamformees (STA A and STA B). If the signal is transmitted from $m \in \{0, 1, ...M-1\}$ antennas and received by $n \in \{0, 1, ...N-1\}$ antennas the CFR matrix $\mathbf{H}$ is represented equation 2 where $\mathbf{A_p}$ and $\tau_\mathbf{p}$ are the attenuation and delay experienced by path $\mathbf{P}$.

$$\begin{aligned} H_k(n) &= A_k(n)e^{j\phi k(n)} \\ &= \sum_{P=0}^{P-1} A_p(n)e^{-j2\pi(f_c+k/T)\tau_p(n)} \end{aligned} \quad (2)$$

The beamformer (AP) derives $\mathbf{W}$ from the $\mathbf{H}$ matrix to steer the transmission streams to enhance the power towards single or multiple beamformees (STA) simultaneously. This enhances signal strength, improving data rate and reducing transmission delay. MU-MIMO task offloading mitigates the impact of individual stream capacity limitations, preserving overall network performance.



(a) Data rate and latency at different antenna configuration (at 80 MHz)



(b) Data rate and latency at different bandwidths (with $4\times4 system$)

Fig. 4: Data rate and latency at different antenna configurations and bandwidths

B. MU-MIMO Limitations in MECS

Link quality, including data rate, packet loss, and latency are influenced by factors such as the number of transmission and

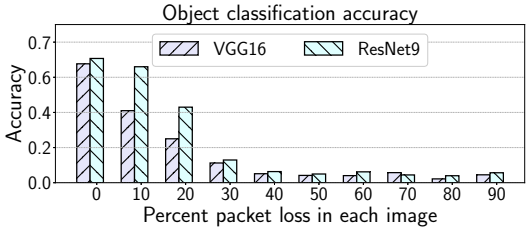


Fig. 5: Object (Image) classification accuracy at different percentages of packet losses with CIFAR 100 dataset.

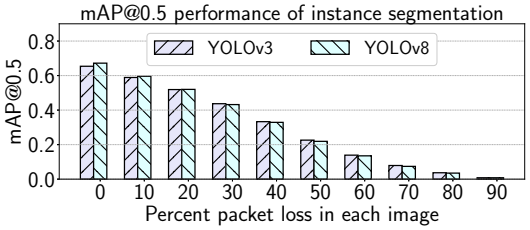


Fig. 6: mean average precision (mAP)@0.5 performance of instance segmentation at different percentages of packet losses.

reception streams, available bandwidth, and the number of connected STAs. Increasing bandwidth and stream count improves data rates and reduces latency (see Fig. 4). However, average performance metrics do not reflect temporal channel degradations, which are crucial for mission-oriented task offloading scenarios requiring per-transmission reliability guarantees.

Additionally, while allocating more streams and bandwidth enhances performance, it increases power consumption [25]. This accelerated energy depletion limits multi-user support, particularly in energy-constrained devices.

Therefore, it is essential to configure MU-MIMO systems to ensure task-level reliability while efficiently managing both wireless resources and energy within a MECS. This need motivates the development of a control framework for task-aware, resource-efficient transmission management, as discussed in the following sections.

C. Computer Vision Task Performance with Packet Loss

We demonstrate the relationship between task semantics, performance, and packet loss for two common machine learning (ML)-based computer vision tasks: (i) object detection and classification, and (ii) instance segmentation. Results show that per-image loss tolerance depends on both the task type and the specific model employed.

1) Object detection and classification with packet loss: In object detection and classification, a neural model assigns a label to an image based on features like edges and corners. During training, these features influence class prediction performance. Accuracy, defined as the ratio of correct predictions (true positives and negatives), indicates performance, with scores near 1 reflecting better results on balanced datasets. Fig. 5 shows object detection and classification accuracy obtained using the VGG16 [26] and ResNet9 [27] models trained and tested on the CIFAR 100 dataset [28] as a function of the packet loss affecting the input image. The performance of both models decays drastically when the percentage of packet loss in an image is above 30%. The VGG16 architecture is composed of a uniform arrangement of convolution layers, this differs from the ResNet9 architecture [27], which is reflected in the performance variation, for instance, an image with 10% of packet loss downgrades the accuracy by half compared to the performance with no loss.

2) Instance segmentation with packet loss: In instance segmentation, a model identifies objects and classifies their pixels. The intersection over union (IoU) quantifies the overlap between predicted and ground truth bounding boxes, with an IoU of 0.5 indicating at least 50% overlap in bounding boxes and pixel-level segmentation.mAP at 0.5 IoU (mAP@0.5) computes the average precision (AP) for each class of the object and then takes the mean across all classes. AP accounts for both precision (how many of the predicted objects are correct) and recall (how many of the actual objects were detected). A higher mAP@0.5 score indicates more accurate localization and pixel-level classification.

Fig. 6 depict the mAP@0.5 performance and precision-recall (PR) curve respectively of instance segmentation with COCO [29] validation dataset for different percentages of packet loss in every image with two state of the art (SOTA) models: YOLOv5 and YOLOv8 [30]. mAP@0.5 performance is 0.67% when there is no packet loss in the images which degrades to 0.219% and 0.008% when the packet losses are 50% and 100% respectively with YOLOv8.

IV. SOAR Task Offloading System

A. Problem Formulation

We consider a MECS comprising a MED and multiple ESs connected via an IEEE 802.11ac MU-MIMO network, where the AP and STAs represent the MED and ESs, respectively. The system emulates real-time task offloading, offloading image frames from benchmark datasets from the MED (AP) to the ESs (STAs) at a predefined frame rate. The goal is to optimize wireless resource usage while meeting deadlines and task performance constraints. We denote the system control parameters as X_{b_i,p_i,r_i}, a vector composed of three key variables: the bandwidth $b_i \in B = \{x_1, \ldots, x_n\}$ MHz, the number of parallel streams in the antenna configuration $p_i \in A = \{1 \times 1, \ldots, n \times n\}$, and the packetization strategy used per device $r_i \in Y = \{y_1, \ldots, y_n\}$. We define two packetization strategies, they are duplication and parallelization; in duplication, a packet is duplicated as much as the number of streams in the configuration such that a $4{\times}4$ antenna configuration has 4 copies of a packet whereas parallelization implies that each stream transmits a unique packet. We define the task performance requirements (D_i) and map them to a target packet loss ratio per image as $D_i \in F{=}\{F_{type_1}, ..., F_{type_n}\}$. Each task has a latency constraint corresponding to a deadline T, which is the maximum time given to the system to complete the task and maps to a maximum time to deliver the information to the ES. We, then, denote the objective function as the expectation of the

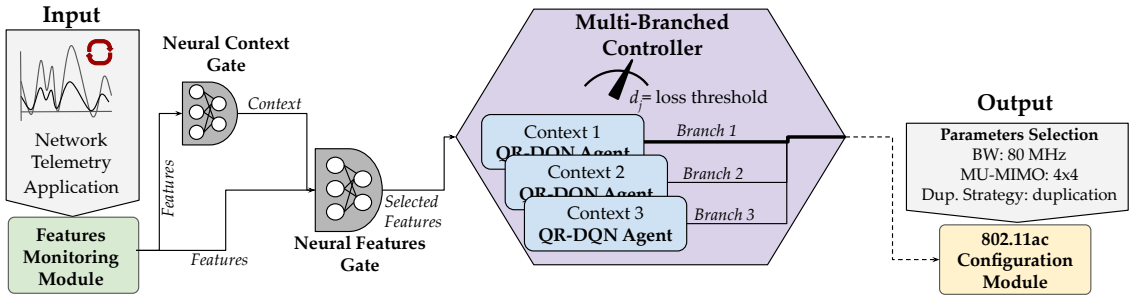


Fig. 7: Dataflow of the SOAR system: We process a set of features from a diversity of context traces to optimize the wireless configuration based on the ML task requirements.

cost of the control variables using the following resources consumption model adapted to MU-MIMO network configuration:

$$X_{b_t,p_t,r_t} = f_0 + f_1(b_t) + f_2(p_t) + f_3(r_t) \tag{3}$$

In the above equation, each f_i represents the raw resource usage i.e. bandwidth, number of transmission streams and packetization strategy to express the relevance of each component as well as the relationship between the usage of each component and the cost. We define the performance constraint as the probability of having the minimum number of packets d_j successfully received within the target time T. The probability M_j, defined as

$$M_j = P(d_j > D_j|\kappa), \tag{4}$$

is set based on application reliability demands. We define our optimization problem as follows:

$$\min \mathbb{E}[X_{b_i,p_i,r_i}], \tag{5}$$

$$\begin{aligned} \text{s.t. } & D_j \ \forall j \in \{\text{type}_1, \ldots, \text{type}_n\} \\ & \kappa \leq T, \ \ \kappa > 0 \\ & f_0, f_1, f_2, f_3 > 0 \end{aligned}$$

B. System Overview

To solve the problem defined in the previous section, we propose the context-aware control framework SOAR. The framework embeds a multi-branched neural model that selects the distributional DRL agent based on the perceived context extracted from readily observable features. Specifically, the components of the framework are (see Fig. 7): (i) features monitoring module, (ii) neural gates: context and features, (iii) multi-branched controller and (iv) 802.11ac configuration module.

The *Features Monitoring Module* collects the MED system state, including sensor readings, and telemetry, wireless communication and application-specific features. The inputs for this model encompass telemetry data, such as accelerometer, gyroscope measurements, network parameters, including transmitter and receiver signal strengths, data rate, and application-specific parameters such as frames per second and task type.

The neural gates process data from the *Features Monitoring Module* to identify context and key features. The lightweight *Context Gate*, a trained classifier, uses a Flatten layer to preprocess input data into a 1D vector, followed by three dense layers with Rectified Linear Unit (ReLU) activations, L2 regularization, and a 20% dropout layer to prevent overfitting. Its logits, mapped to three context classes (e.g., LoS and NLoS), optimize for binary cross-entropy loss. The *Neural Features Gate*, a random forest model, selects context-specific features to predict packet losses, tailoring feature selection based on the context identified by the context gate.

The *Multi-Branched Controller* module uses the contextual features from the neural feature gate to determine the quantile regression DQN (QR-DQN) agent required to optimize the wireless configuration. Each agent promotes policies that benefit the different contexts and task performances. The *802.11ac Configuration Module* modifies the bandwidth, number of streams to use for transmission, and the packet duplication strategy.

C. Distributional Deep Reinforcement Learning

We implement our controller as a Distributional Deep Reinforcement Learning (DRL) agent, with a reward function that encodes both the objective and constraints of our optimization problem. As with standard RL approaches, our formulation is based on a Markov Decision Process (MDP), defined by the tuple $(S, \mathcal{A}, R, P, \gamma)$, where S and $\mathcal{A}$ are the state and action spaces, R is the reward function, $P(s_{t+1}|s_t, a_t)$ defines the transition probability, and $\gamma \in (0, 1]$ is the discount factor for future rewards. A policy $\pi(\cdot|s)$ maps each state $s \in S$ to a distribution over actions. While standard RL optimizes the expected return $G^\pi = \sum_{t=0}^{\infty} \gamma^t R_t$, this approach often overlooks low-probability states and ignores the variability of long-term rewards. Moreover, learning expected returns directly typically requires a large number of samples.

We propose to adopt a distributional reasoning, and specifically use distributional RL algorithms, which explicitly consider the uncertainty in the long-term outcome of actions. This class of algorithms has been shown to improve the performance of the controller in many ways, from enabling faster learning to improving generalization. While DDRL has seen a limited investigation in robotics, its use in the optimization of wireless networks is almost completely unexplored. We take as starting point the distributional DRL model-free algorithm called QR-DQN [11]. The QR-DQN algorithm is based on the DQN (Deep Q Network) [31], a standard DDRL algorithm where the expected return is the q-value of a state-action combination for an environment $\mathcal{E}$ and it is calculated as follows:

$$Q^*(s,a) = \mathop{\mathbb{E}}_{s_{t+1}\sim\mathcal{E}}[R_t + \gamma \max_{a+1} Q^*(s_{t+1}, a_{t+1})|s, a]. \tag{6}$$

In QR-DQN, the q-value is expanded to a distribution of quantiles per action. The algorithm defines a quantile distribution Z that maps each state-action pair (s, a) to a uniform probability distribution supported on $\theta_i(s, a)$ as follows:

$$Z_\theta(s,a) = \frac{1}{N}\sum_{i=1}^{N}\delta_{\theta_i}(s,a), \tag{7}$$

where δ_z denotes a Dirac at $z \in \mathbb{R}$. Then, following this formulation [11],demonstrates that for a quantile $\tau \in [0,1]$ the minimizer for $F_z^{-1}(\tau)$ is given by the quantile regression loss:

$$L^\tau(\theta) = \mathop{\mathbb{E}}_{\hat{Z}\sim Z}[\rho_\tau(\hat{Z}-\theta)], \tag{8}$$

$$\rho_\tau(u) = u(\tau - \delta_{u<0})\forall u \in \mathbb{R}, \tag{9}$$

where δ_u denotes a Dirac in u. To solve our optimization problem, we define our action space $\mathcal{A} = B \times A \times Y$ as the number of possible wireless configurations for bandwidth, number of streams, and packetization strategy. Our state space S is defined by the context which represents a subset of the telemetry, network, and application features. The reward function is denoted as follows:

$$R = \begin{cases} \lambda(f(X_{b_t,p_t,r_t})) + (1-\lambda)d_j + \frac{1}{1+e^x} & \text{if } d_j > D_j \\ \lambda(f(X_{b_t,p_t,r_t})) + (1-\lambda)d_j & \text{otherwise.} \end{cases} \tag{10}$$

Where λ values are within the $(0, 1]$ range that represent the policy priorities and x is the task latency.

Different from any existing DDRL algorithm, we train a set of context-specific agents, and use the *neural context gate* to select the agent, thus building the first multi-branched exemplar of DDRL. Notably, each agent will use a context-specific set of features as input to maximize its performance while minimizing its complexity.

V. Dataset and Setup

We build a dataset focused on monitoring the state of the system composed by MED and the ES within two different contexts for robotics applications. The dataset aims to provide insights into the correlation between the controllable parameters of the wireless channel such as the number of users. The maximum number of streams, the bandwidth used, the packetization strategy, and the inherent characteristics of the environment and application.

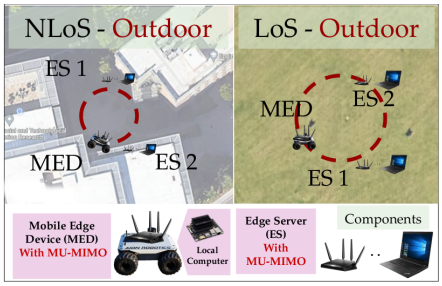


Fig. 8: NLoS and LoS contexts for SOAR framework.

We collected features from two different scenarios: LoS and NLoS. For the LoS scenario, we have deployed our equipment in an isolated park area with no interference coming from infrastructure or other mobile devices. Alternatively, for the NLoS scenario, we use an urban environment with buildings and wireless infrastructure hindering packet transmission.

In both scenarios, the MED follows a circular trajectory (with a ∼12m radius, see Fig. 8). Each experiment is characterized by completing twice the same trajectory with specific configuration parameters, for instance, a fixed number of streams, bandwidth, transmitted frames per second used to transmit data and the number of ES that the MED transmits its images.

We leverage the MU-MIMO capabilities of the wireless channel by equating the number of users to the number of simultaneous data streams it can support. Consequently, the maximum number of simultaneous users is constrained by the hardware's limitations. To evaluate performance under these conditions, we conducted experiments with multiple transmissions, ranging from 1 to 4 streams.

We augmented our dataset by applying multiple image deadlines, reflecting different packet loss patterns. Also, we dropped packets received after application deadline, and recorded the number of received images based on the packetization strategy. The hardware used for MED includes a custom-designed rover equipped with a Jetson Nano microcomputer deployed to oversee the execution of critical tasks, including telemetry management, network operations, and packet logging. The hardware used for the ES is a laptop with a GPU that could process the transmitted image at their end.

In the pursuit of performing 802.11ac data transmission, particularly for utilizing MU-MIMO technology, we established a local-area network (LAN) infrastructure by interconnecting multiple routers. Each MED and ES has its router to transmit images. We modify the wireless configuration by changing the channel bandwidth and number of transmission streams. Our network configuration is comprised of a mobile AP which is our MED attached to the rover, alongside two stationary STAs which are our ESs that transmit a rate of images per second using a non-reliable network protocol (UDP).

VI. Training and Evaluation

We implement custom machine learning models in multiple components of the SOAR framework, particularly in the *neural features gate* and *neural context gate* modules. In the following, we describe the implementation details, the evaluation of our approach, and an analysis of the DDRL approached used.

To build the *neural features gate*, we train a random forest classifier to select the useful features as mentioned in section IV-B. Our tests show that the optimal set of features is heavily dependent on the context; for instance, in the LoS context: fps, X_{accel}, Y_{accel}, distance, and signal strength are the relevant features while in the NLoS context: X_{gyro}, Y_{gyro}, X_{accel}, Y_{accel}, distance and Rx bitrate perform better toward packet loss prediction. In the *neural context gate*, we train a classification model based on the architecture of the features gate, adding two dense layers and using a Softmax activation for simplicity. We use 10% of our data and a 5 cross-validation split to avoid overfitting, after training we achieve 91.43% of accuracy which

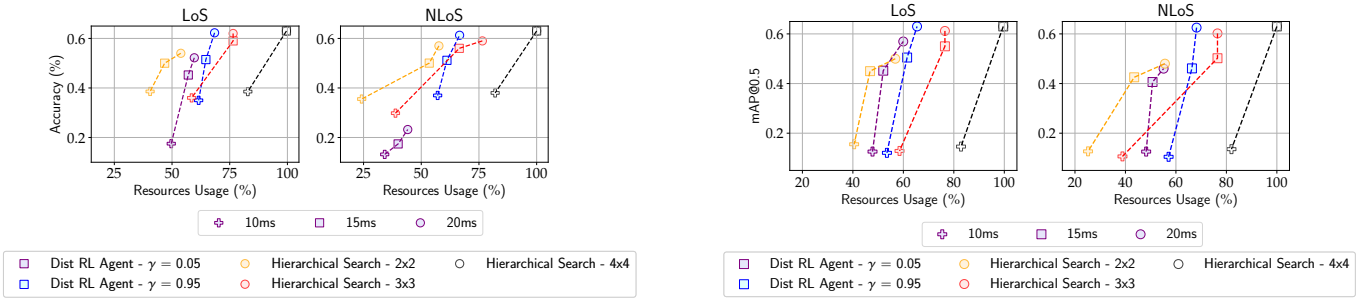


Fig. 9: Delay constraints are represented by shapes and optimization algorithms are distinguished by color. (Left) Image classification performance using ResNet9 [27] model within LoS and NLoS contexts, and (Right) Instance segmentation using YOLOv8 [30] model

infers that context prediction with the selected features is highly reliable.

We propose a distributional DRL-based algorithm to dynamically select optimal wireless configurations for a MED trajectory, taking into account the erratic behavior of channels in real world environments. Our custom environment supports context-specific features and hardware-compatible actions. We deploy a QR-DQN agent per context. The state space is derived from our dataset, and agent performance is evaluated on two robotics-related computer vision tasks: instance segmentation and image classification. As discussed in Section III, task performance varies with the portion of the image received, while wireless configurations and application deadlines significantly impact packet loss. As part of the evaluation of the disributional DRL agents, we train agents for two types of strategies: myopic and long-term. These policies are differentiated by the values assigned to the discount factor γ. The myopic policy corresponds to $\gamma = 0.05$, indicating a short-sighted approach where immediate rewards are prioritized over future gains. Contrarily, a higher γ value of 0.95 emphasizes future rewards.

Due to the lack of literature that optimizes the wireless channel with MU-MIMO using real-time systems we compare our solution to a baseline algorithim –a hierarchical tree search where the antenna configuration is fixed, e.g. 2×2 within the trajectory followed by the MED.

The baseline algorithm finds the minimum amount of resources (determined by the bandwidth and packetization strategy) that the MED can use to get the minimal number of packet loss per second. Our experiments consider all possible MU-MIMO antenna configurations as well as 3 different real-time systems deadline requirements: 10, 15 and 20 ms. The resource usage calculation is based on the components of the possible wireless configuration which are the bandwidth, number of streams, and packetization strategy. Each component spends a percentage of the total MED resources, in this way, the maximum resources expenditure in a wireless configuration corresponds to the 4×4 antenna arrange, 80 MHz bandwidth, and duplication as packetization strategy. From these variables the prioritization goes as the hierarchical search, being the antenna configuration the parameter that is more influential in the resource utilization, followed by bandwidth and packetization.

The classification task results shown in Fig. 9, we notice that within a 20 ms deadline, 3×3 and 4×4 fixed wireless configurations achieve the highest accuracy offered by ResNet9 [27] model in both contexts with their respective costs in the resources usage when the hierarchical search is implemented. The myopic policies compared among contexts demonstrated a better performance in the LoS context – our intuition is that this is due to the lower complexity environment compared to the NLoS. Additionally, we observe that given a comparable performance, the LoS agent reaches its best performance using 35% less resources than the fixed wireless configuration, and the NLoS agent reaches its best performance using 40% less resources than the fixed configuration.

The segmentation task results for LoS and NLoS are shown in Fig 9 , we notice that the highest accuracy of the YOLOv8 [30] model is reached – similarly to the classification task – within the 20 ms deadline and using a 3 and 4 streams fixed configuration. Contrary to the classification task, the myopic policy achieves a better mAP as the performance degrades gradually with the packet loss compared to the classification task. We also observe that the mAP remains above 0.5 for the 15ms deadline. Resources usage is reduced in both contexts, moreover for a certain number of resources SOAR can use smaller deadlines (thus decreasing the overall latency) without perceivably degrading the performance of the tasks.

Our algorithm optimizes reward distributions by efficiently identifying edge cases, improving resource allocation as reflected in the action distribution. Through 500-episode experiments, illustrated in Figure 10, we compare action distributions across line-of-sight (LoS) and non-line-of-sight (NLoS) contexts. The myopic policy generates a broader action range, with packet duplication more frequent in NLoS scenarios, indicating higher resource consumption due to increased complexity. Bandwidth preferences also differ: 80 MHz is primarily selected in NLoS contexts, while 40 MHz is favored in LoS situations.

VII. Conclusions

In this paper, we addressed the challenge of reliable task offloading within the context of robotic applications, where MED mobility, wireless communication, and real-time ML task requirements are critical. To tackle this, we proposed the SOAR framework, a context-aware wireless semantic configuration

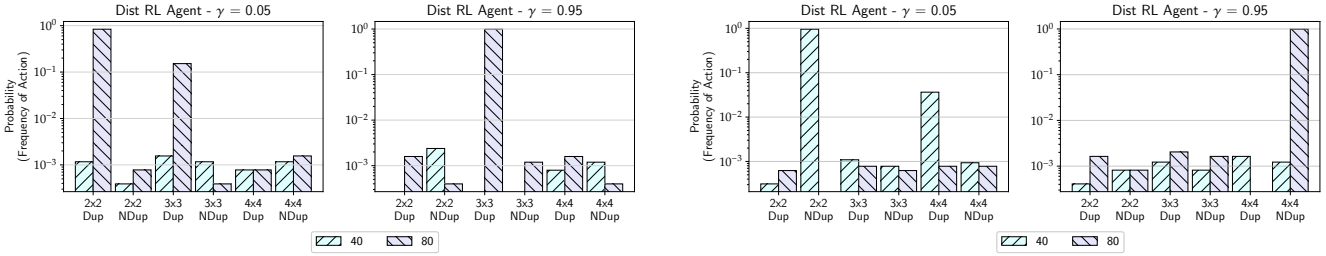


Fig. 10: Action distribution for 20ms for different γ parameters in NLos(Left) and LoS(Right) context. Dup and NoDup stand for each packetization strategy implemented.

solution for resource optimization in MECS. To assess the performance of SOAR, we assembled a comprehensive dataset comprising transmitted images through a MU-MIMO channel within a context (NLoS and LoS) trajectory. Our analysis involved the implementation and development of multiple data gates and modules to extract features with predictive capabilities, and to identify the context to which a subset of features are more relevant than others. Results demonstrated that SOAR can identify contexts using real-world data through feature selection and a neural context gate implemented within its design. SOAR's DDRL context agent maximizes the reward distribution in a horizon which is reflected in the adaptability of its policy to the tasks performance while reducing the resources utilization over the MED trajectory. Specifically, SOAR achieves a task performance using 35-40% of resources when compared to a fixed wireless configuration optimization for an offloading deadline of 20ms within both contexts.

ACKNOWLEDGEMENTS

This work is funded in part by the National Science Foundation (NSF) grant CCF 2140154, CNS-2134973 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.

REFERENCES

[1] R. Group, "Mu-mimo drone offloading." https://github.com/Restuccia-Group/MU-MIMO-Drone-Offloading, 2024. Accessed: 2025-04-14.

[2] M. Y. Akhlaqi and Z. B. M. Hanapi, "Task offloading paradigm in mobile edge computing-current issues, adopted approaches, and future directions," *J. Netw. Comput. Appl.*, vol. 212, p. 103568, 2023.

[3] K. F. Haque, F. Meneghello, M. E. Karim, and F. Restuccia, "Sawec: Sensing-assisted wireless edge computing," *arXiv*, 2024.

[4] H. Li, P. Zheng, T. Wang, J. Wang, and T. Liu, "A multi-objective task offloading based on bbo algorithm under deadline constrain in mobile edge computing," *Cluster Comput.*, vol. 26, no. 6, pp. 4051–4067, 2023.

[5] F. Li, X. Wang, Z. Wang, *et al.*, "A local communication system over wi-fi direct: Implementation and performance evaluation," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5140–5158, 2020.

[6] A. Acheampong, Y. Zhang, X. Xu, and D. A. Kumah, "A review of the current task offloading algorithms, strategies and approach in edge computing systems," *Comput. Model. Eng. Sci.*, vol. 134, pp. 35–88, 2023.

[7] D. S. Lakew, N.-N. Dao, S. Cho, *et al.*, "Adaptive partial offloading and resource harmonization in wireless edge computing-assisted ioe networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3028–3044, 2022.

[8] Y. Matsubara, R. Yang, M. Levorato, and S. Mandt, "Supervised compression for resource-constrained edge computing systems," in *IEEE/CVF WACV*, 2022.

[9] N. Larrakoetxea, J. Astobiza, I. Lopez, *et al.*, "Efficient machine learning on edge computing through data compression techniques," *IEEE Access*, vol. 11, pp. 31676–31685, 2023.

[10] Y. Park, U. Gim, and M. J. Kim, "Edge storage management recipe with zero-shot data compression for road anomaly detection," 2023.

[11] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional Reinforcement Learning with Quantile Regression," 2017.

[12] F. Saeik, M. Avgeris, D. Spatharakis, *et al.*, "Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions," *Comput. Netw.*, vol. 195, p. 108177, 2021.

[13] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *J. Syst. Archit.*, vol. 118, p. 102225, 2021.

[14] M. Ahmed, S. Raza, M. A. Mirza, *et al.*, "A survey on vehicular task offloading: Classification, issues, and challenges," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 7, pp. 4135–4162, 2022.

[15] H. Guo, J. Liu, and J. Lv, "Toward intelligent task offloading at the edge," *IEEE Netw.*, vol. 34, no. 2, pp. 128–134, 2019.

[16] A. Fresa and J. P. Champati, "Offloading algorithms for maximizing inference accuracy on edge device under a time constraint," *arXiv*, 2021.

[17] Y. Matsubara, D. Callegaro, S. Singh, M. Levorato, and F. Restuccia, "Bottlefit: Learning compressed representations in deep neural networks for effective and efficient split computing," *arXiv*, 2022.

[18] D. S. Lakew, A.-T. Tran, N.-N. Dao, and S. Cho, "Intelligent self-optimization for task offloading in leo-mec-assisted energy-harvesting-uav systems," *IEEE Trans. Netw. Sci. Eng.*, pp. 1–14, 2024.

[19] D. Callegaro, M. Levorato, and F. Restuccia, "Seremas: Self-resilient mobile autonomous systems through predictive edge computing," in *IEEE SECON*, pp. 1–9, 2021.

[20] H. E. Ilhan, S. Ozer, G. K. Kurt, and H. A. Cirpan, "Offloading deep learning empowered image segmentation from uav to edge server," in *IEEE TSP*, pp. 296–300, 2021.

[21] S. Ozer, E. Ilhan, M. Ozkanoglu, and H. Cirpan, "Offloading deep learning powered vision tasks from uav to 5g edge server with denoising," *IEEE Trans. Veh. Technol.*, 2023.

[22] L. Niu, X. Chen, N. Zhang, *et al.*, "Multi-agent meta-reinforcement learning for optimized task scheduling in heterogeneous edge computing systems," *IEEE Internet Things J.*, 2023.

[23] I. 802.11ac, "Ieee standard for information technology local and metropolitan area networks part 11: Wireless lan medium access control (mac) and physical layer (phy)," 2014.

[24] "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN," *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*, pp. 1–767, 2021.

[25] P. Silva, N. T. Almeida, and R. Campos, "A comprehensive study on enterprise wi-fi access points power consumption," *IEEE Access*, vol. 7, pp. 96841–96867, 2019.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, pp. 770–778, 2016.

[28] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-100 (canadian institute for advanced research, 100 classes)," tech. rep., CIFAR-100, 2009.

[29] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," 2015.

[30] G. Jocher, A. Chaurasia, and J. Qiu, "Yolo by ultralytics," 2023.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," 2013.