Joint Communication Resource Slicing and Dynamic DNN Model Selection for Computer Vision-based Systems

Michael Lesko-Krleza
Electrical and Computer Engineering
Concordia University
Montreal, QC, Canada
m_lesk@live.concordia.ca

Rodolfo W. L. Coutinho
Electrical and Computer Engineering
Concordia University
Montreal, QC, Canada
rodolfo.coutinho@concordia.ca

Yousef R. Shayan
Electrical and Computer Engineering
Concordia University
Montreal, QC, Canada
yousef.shayan@concordia.ca

***Abstract*—Video analytics is a building block in envisioned computer vision-based smart systems. Video frames from user devices must be analyzed promptly for feature extraction aimed at supporting smart decisions or augmenting users' perception of the physical world. The local analysis of video frames at user devices faces resource limitations, while the remote analysis on the edge server might be impacted by high workload at the servers and communication constraints. In this paper, we propose a novel deep reinforcement learning (DRL) framework to jointly optimize the offloading rate, allocate uplink communication resources, and select the deep neural network (DNN) model at the edge server for each video flow in computer vision-based systems. The overall goal is to intelligently adapt the offload rate, allocate uplink communication resources for user devices when offloading video frames, and periodically select the most appropriate DNN model at the edge server to handle video frames offloaded by each device. Such intelligent decisions are aimed at ensuring that the accuracy and latency constraints for each user device are met, while the video frame drop rate is at acceptable levels. Preliminary numerical results demonstrate the feasibility of the proposed DRL solution in maintaining the experienced accuracy and latency over time close to the threshold requirements for video flows.**

***Index Terms*—Edge computing, computer vision-based systems, modeling and performance evaluation**

I. Introduction

Computer vision will play an important role in many of the envisioned mobile systems. The timely analysis of the video frames produced by devices will enhance information and decision-making in intelligent transportation systems, smart navigation, robotic visual inspections, and smart factories, for instance [1]–[3]. Computer vision will also be an important building block in mixed reality systems, where users' perception of the physical world is augmented with computationally generated information overlaid on their field of view. In a classical computer vision-based system, user devices will produce video frames that must be analyzed promptly. The video frames will be analyzed by deep neural network (DNN) models, for instance, deployed at the user device and remote servers. The classical approach deploys powerful, complex DNN models at cloud servers, which process video frames offloaded by user devices. This approach might render it infeasible due to the low-latency requirement in computer vision-based systems.

Hence, distributed video analytics at user devices and edge servers has emerged as a promising solution. In this approach, part of the video frames produced by a user device is processed locally at the device, whereas the rest is offloaded and processed at edge servers. This approach helps to cope with the stringent low-latency requirement in computer vision-based systems. Nevertheless, the problems of deciding the offloading rate and managing local and remote resources need to be addressed. Each device must decide the fraction of frames it will process locally and the fraction of frames to be offloaded to the edge server. Such decisions will have a direct impact on the latency, including the latency for local and remote processing, video frame offloading, and waiting times in the processing and offloading queues. Many studies in the literature have proposed optimized solutions for task offloading in computer vision-based systems [4], [5]. The typical approach is to optimize video frame offloading decisions based on network conditions and/or resource availability at the user device and edge/cloud servers [6]–[8]. Another typical trend consists of mechanisms to adaptively adjust video configuration, i.e., downsampling, upscaling, compression, and encoding, based on network conditions and resource availability, aimed at coping with the dynamic nature of network conditions and resource demand [9]–[11].

In contrast, this paper proposes a novel optimization framework for video frame scheduling, communication resource management, and DNN model selection in computer vision-based systems. While studies in the field mainly focus on offloading decisions and edge computing resource allocation, we concentrate on the management and allocation of uplink communication resources and the offloading rate, along with the dynamic selection of the DNN model at the edge server to support video inference for each user device. We propose a novel optimization problem aimed at minimizing quality of service (QoS) violations, in terms of inference accuracy and latency, of video analytics with different QoS requirements. We formulate the optimization problem as a Markovian decision

979-8-3195-0167-7/26/$31.00 ©2026 IEEE

problem and develop a Proximal Policy Optimization (PPO) reinforcement learning algorithm to optimally allocate uplink communication resources, to set a mobile device offloading rate, and to select the DNN model at the edge server to handle video frame analytics for each user device. Preliminary numerical results show the feasibility of the proposed solution in maintaining drop rate at low levels while maintaining latency and accuracy at acceptable levels.

The remainder of this paper is organized as follows. Section II presents related work. Section III develops a stochastic model for the computer vision-based mobile system under consideration. Section IV formulates the optimization problem aimed at ensuring the latency and accuracy QoS requirements for computer vision-based systems. Section V presents the Markovian Decision Process (MDP) formulation for the devised optimization problem. Section VI briefly discusses the deep reinforcement learning (DRL) framework used to solve the formulated MDP problem. Section VII provides a performance evaluation of the proposed solution. Finally, Section VIII presents the final remarks.

II. Related Work

Recent studies have focused on optimizing different aspects to enable computer vision-based mobile systems.

Zheng et al. [12] proposed the EAMEM framework, which employs classification, prioritizes video streaming tasks, and implements a multi-objective optimization solution for allocating bandwidth and computational resources to tasks based on their criticality, available resources, current network conditions, and historical performance of similar tasks.

Bai et al. [10] proposed a novel framework for knob configuration and resource allocation in edge-aided computer vision-based systems. The proposed solution dynamically adapts video sampling and resizing, allocates communication channels, and computational resources.

Li et al. [13] proposed the EPPO framework for jointly performing scheduling decisions, video frame compression, and DNN model caching in a device-edge collaborative system for computer vision. The proposed EPPO framework implements a utility-based function that weights accuracy satisfaction and latency when optimizing video frame scheduling, frame compression, and DNN caching.

Qian et al. [14] designed a PPO-based framework for optimizing video frame scheduling and communication resource allocation in computer vision-based systems. The proposed solution implements a QoS satisfaction-based utility function to ensure that latency and accuracy for video frame analytics are within determined thresholds.

The authors in [15] developed a novel orchestrator for load balancing in multi-edge systems for computer vision analytics. The proposed solution considers the unique latency requirements of each video frame and decides the most suitable edge server to handle it. Moreover, task migration is implemented when computing resources of an edge server must be freed to support the analysis of an incoming video frame request.

Alqahtani et al. [16] proposed the ECORE framework for energy-efficient computer vision analytics. ECORE implements a routing algorithm to determine the most suitable edge server based on object characteristics and is aimed at reducing energy consumption.

Coutinho and Boukerche [5] designed a novel framework for distributed video frame analytics. The proposed framework enables computing resources of IoT devices to be shared and used to support the video analytics of neighboring devices when resources are idle.

In contrast, this paper aims to jointly optimize the video frame scheduling, communication resource allocation, and DNN model selection for computer vision systems. It implements a PPO-based framework for solving the devised optimization problem aimed at ensuring the accuracy and latency requirements for video flows. Nevertheless, the proposed solution implements a heuristic to assign communication resources that might be idle after the PPO agent assignment. This is to ensure that all resources are being used, reducing offloading latency and overall latency for video frame analytics.

III. System Model

We consider a system infrastructure designed to provide communication and compute resources for computer vision-based applications. At each discrete time step $t \in \mathcal{T} = \{1, \ldots, T\}$ with a duration of T_s seconds, the set of UEs served by the base station is defined as $\mathcal{U}(t)$. Each UE produces computer vision tasks as part of the application it is running. A computer vision task is a video frame to be analyzed by a DNN model. The video frames at a given UE u_i will be produced at a fixed rate of f_{u_i} fps and encoded at a resolution of $r_i \times r_i$ pixels. Each video frame task is characterized by a desired maximum latency l^a and minimum accuracy ϵ^a QoS requirement. A UE u_i can handle a video frame inference task locally or offload it to be handled at the edge server. Accordingly, a UE u_i will offload to the edge server a total of $\alpha_{u_i} \times 100\,\%$ of the video frames it produces. The remaining frames will be processed locally.

An edge server will provide compute resources for the inference of video frames offloaded by UEs. A set of DNN models will perform the video frame inference at the edge server. We define the set $\mathcal{D} = \{d_1, \ldots, d_L\}$ of L DNN models capable of handling computer vision tasks at the edge server. Each model $D_{u_i} \in \mathcal{D}$ is characterized by a complexity $C(D_{u_i})$, given in the number of floating-point operations (FLOPs) necessary to perform the video frame inference, and an accuracy $A(D_{u_i})$. DNN models at the edge server run concurrently, and therefore, processing resources at the server must be allocated for each model. It is assumed that the edge server presents a maximum processing capability of p_e cycles/second and an amount of $p_e^{D_{u_i}}$ cycles/second is allocated to the DNN model $D_{u_i} \in \mathcal{D}$.

The decisions to be made in the designed system are the following. First, the offloading rate at each user device must be determined. Next, the amount of communication resources, i.e., Physical Resource Blocks (PRBs), allocated to each UE

must be determined, which will impact the data rate each UE experiences when offloading video frames. Moreover, the DNN model to handle computer vision tasks offloaded by a UE during time slot t must be selected. The selection of the DNN model will impact the latency and accuracy experienced by the u_i's application.

A. User equipment modeling

We model each UE u_i as two independent queue systems. One queue system models the local inference of computer vision tasks, while the other models the component used to offload tasks to the edge server. We model the local processing system at UE u_i as a D/M/1 queue. The video frame arrival rate at the local processing system is deterministic and given as:

$$\lambda_p^{u_i} = (1 - \alpha_{u_i}) f_{u_i} T_s. \tag{1}$$

To model the service rate, we use an exponential distribution with the parameter given as:

$$\mu_p^{u_i} = \frac{1}{l_{u_i}^{d_0}} T_s, \tag{2}$$

frames per slot, where $l_{u_i}^{d_0}$ seconds is the amount of time the DNN model d_0 at the UE u_i requires to finish the inference of a video frame. This inference time is estimated as:

$$l_{u_i}^{d_0} = \frac{\rho_i C(d_0)}{p_i}, \tag{3}$$

where ρ_i is the number of CPU cycles required to complete a floating-point operation at u_i, $C(d_0)$ FLOPs is the complexity of the DNN model running at u_i, and p_i Hz is the processing capability of UE u_i. The parameter ρ_i depends on the processing unit in the UE.

To ensure a stable system, video frames must be dropped when the arrival rate approaches the service rate. Herein, video frames are dropped when the arrival rate at the local processing queue exceeds 95% of the UE u_i's service rate. Thus, the effective video frame arrival rate at the local processing queue is $\lambda_p^{u_i'} = \min(\lambda_p^{u_i}, 0.95\mu_p^{u_i})$. Hence, the rate at which frames will be dropped at the local processing queue of UE u_i is:

$$\delta_l^{u_i} = \max\left(0, \lambda_p^{u_i} - \frac{0.95 T_s}{l_{u_i}^{d_0}}\right). \tag{4}$$

The expected sojourn time of the local processing queue at a UE u_i can be approximated as:

$$E[R]_l^{u_i} \approx \frac{2\rho_l^{u_i} - 1}{2\mu_p^{u_i}(1 - \rho_l^{u_i})}, \tag{5}$$

where $\rho_l^{u_i} = \lambda_p^{u_i'} / \mu_p^{u_i}$ is the utilization of the local processing system.

We model the offloading system at UE u_i as an M/M/1 queue. The arrival of video frames in the offloading queue is modeled as a Poisson process with mean arrival rate, in frames per slot, as:

$$\lambda_o^{u_i} = \alpha_{u_i} f_{u_i} T_s. \tag{6}$$

To model the service rate, per slot, of the offloading system, we use an exponential distribution with parameter $\mu_o^{u_i} = (1/d_o^{u_i}) T_s$, where $d_o^{u_i}$ is the amount of time for video frame offloading at u_i. This offloading time is estimated as $d_o^{u_i} = s_i / r_{u_i}$, where $s_i = r_i^2 \times C$ bits is the frame size and r_{u_i} bits/second is the achievable data rate devised at Sec. III-C. The parameter C is the image color depth. If not otherwise specified, C is 24 bits.

To maintain the stability of u_i's offloading system, video frames are dropped when the arrival rate approaches 95% of the service rate. Thus, the effective video frame arrival rate at the offloading queue is $\lambda_o^{u_i'} = \min(\lambda_o^{u_i}, 0.95\mu_o^{u_i})$. Hence, the rate at which frames will be dropped at the offloading queue of UE u_i is:

$$\delta_o^{u_i} = \max\left(0, \lambda_o^{u_i} - \frac{0.95 T_s}{d_o^{u_i}}\right). \tag{7}$$

Finally, the expected sojourn time of a video frame at u_i's offloading system is:

$$E[R]_o^{u_i} = \frac{1}{\mu_o^{u_i} - \lambda_o^{u_i'}}. \tag{8}$$

B. Edge server modeling

The computing resources p_e Hz at the edge server are managed and allocated to handle the different DNN models $D_{u_i} \in \mathcal{D}$ running on the server. As already mentioned, a portion of $p_e^{D_{u_i}}$ Hz is allocated for each DNN model $D_{u_i} \in \mathcal{D}$. The DNN models will execute independently and be modeled as independent queuing systems. We use an M/M/1 queuing system to model the execution of each DNN model at the edge server. For a given model D_{u_i}, the arrival rate at a time slot t is modeled as a Poisson process with mean rate of $\lambda_e^l(t)$ frames per time slot. The rate at which computer vision tasks arrive in the queuing system of each DNN model at the edge server will depend on the rate at which video frames are offloaded from UEs to the considered D_{u_i} DNN model. We define a binary variable $I_{u_i, D_{u_i}}(t)$ to indicate whether the video frames offloaded by the device u_i at time slot t is addressed to the DNN model D_{u_i} (i.e., $I_{u_i, D_{u_i}}(t) = 1$) or not ($I_{u_i, D_{u_i}}(t) = 0$).

$$\lambda_e^l(t) = \sum_{i=1}^{|\mathcal{U}(t)|} \sum_{l=1}^{|\mathcal{D}|} I_{u_i, D_{u_i}}(t) \lambda_o^{u_i'}. \tag{9}$$

We use an exponential distribution with parameter $\mu_e^{D_{u_i}} = (1/l_e^{D_{u_i}}) T_s$ to model the service rate per slot for the DNN model D_{u_i} running at the edge server. The parameter $l_e^{D_{u_i}}$ is the inference time for the DNN model D_{u_i} when p_e^l Hz of the edge server's computing resources are allocated to the model. The D_{u_i} model inference time is estimated as Eq. 3. To maintain the queuing systems stable, we set the effective arrival rate, at a slot t, for the DNN model D_{u_i} as $\lambda_e^{l'}(t) = \min(\lambda_e^l(t), 0.95\mu_e^{D_{u_i}})$, i.e., at a maximum of 95% of the service rate. Consequently, frames at the DNN model D_{u_i} might be dropped at a fraction of:

$$\delta_e^{D_{u_i}}(t) = \max\left(0, \lambda_e^l(t) - \frac{0.95 T_s}{l_e^{D_{u_i}}}\right). \tag{10}$$

The utilization, at a time slot t, for each DNN model $D_{u_i} \in \mathcal{D}$ at the edge server is modeled as $\rho_e^{D_{u_i}}(t) = \lambda_e^{l'}(t)/\mu_e^{D_{u_i}}$. According to Little's Law, the sojourn time for a frame processed by the model D_{u_i} at the edge server during a time slot t is:

$$E[R]_e^{D_{u_i}}(t) = \frac{1}{\mu_e^{D_{u_i}} - \lambda_e^{l'}(t)}. \quad (11)$$

C. Communication resources

Due to the traffic nature of computer vision systems, we focus on resource allocation in the uplink direction. We consider an uplink orthogonal frequency-division multiple access (OFDMA) system with a total of K_{total} physical resource blocks (PRBs), where each PRB has a bandwidth B_{PRB} Hz. PRBs are communication resources allocated to UEs. Accordingly, the capacity for the channel between a UE u_i and the base station, in bits per second, is estimated as:

$$r_{u_i} = k_{u_i} \times B_{PRB} \times \log_2(1 + 10^{SINR_{u_i}/10}), \quad (12)$$

where k_{u_i} is the total number of PRBs assigned to the UE u_i and $SINR_{u_i}$ dB is the signal-to-interference-plus-noise ratio of a transmission from a UE u_i. The SINR is estimated as:

$$SINR_{u_i} = P_{rx}^{u_i} - 10\log_{10}\left(10^{P_{noise}^{u_i}/10} + 10^{I_{u_i}/10}\right), \quad (13)$$

where $P_{rx}^{u_i}$ dBm is the intensity of the power received at base station from a signal transmitted by UE u_i, $P_{noise}^{u_i}$ is the noise power of u_i and I_{u_i} is the aggregated inter-cell interference when users in adjacent cells re-use allocated PRBs. The noise power of u_i is estimated as:

$$P_{noise}^{u_i} = P_0 + 10\log_{10}(k_{u_i}), \quad (14)$$

where P_0 is the Gaussian white noise [17].

Herein, given the focus on the optimization of PRB resource allocation for users being served by a considered BS, we assume that the inter-cell interference during u_i's transmission can be represented by an average value given through the reception power and interference margin β. Thus, the inter-cell interference, in dB, during u_i's transmission is estimated as:

$$I_{u_i} = P_{rx}^{u_i} - \beta. \quad (15)$$

The parameter β is usually set between [10,15] dB to model the interference level of urban macro-cellular deployments [18]. If not otherwise specified, we assume $\beta = 12$ dB in our experiments.

The intensity of the signal received at base station when u_i transmits, i.e., $P_{rx}^{u_i}$ dBm, is estimated as:

$$P_{rx}^{u_i} = P_t^{u_i} - PL(d_{u_i}) + F_R + F_S + G_{BS} + G_{u_i}, \quad (16)$$

where $P_t^{u_i}$ dBm is the u_i's transmission power, $PL(d_{u_i})$ dB is the path loss at distance d_{u_i} from UE u_i to base station, F_R dB is small-scale Rayleigh fading modeling, F_S dB is the large-scale shadow fading, and G_{BS} and G_{u_i} are the antenna gains of the base station and UE u_i, respectively.

The path loss $PL(d_{u_i})$ over a distance d_{u_i} is calculated as:

$$PL(d_{u_i}) = PL_0 + 10\eta\log_{10}(d_{u_i}), \quad (17)$$

where PL_0 is a constant that represents the gain at 1 m and η is the path loss exponent. The small-scale Rayleigh fading F_R dB can be estimated as:

$$F_R = 20\log_{10}(r), \quad (18)$$

where $r \sim \text{Rayleigh}\left(\frac{1}{\sqrt{2}}\right)$. Similarly, the large-scale shadow fading F_S dB is estimated as:

$$F_S = 10\log_{10}(s), \quad (19)$$

where s follows a log-normal distribution such that $10\log_{10}(s) \sim \mathcal{N}(0, \sigma_{sf}^2)$. The parameter σ_{sf}^2 is the standard deviation set to 8 dB for urban macro scenarios [18].

IV. The Formulated Optimization Problem

In the considered system, the optimization problem consists of selecting the number of PRB resources and the DNN model at the server for each UE. Hence, the multi-objective optimization problem is formulated to minimize the weighted sum of the moving average of the latency and accuracy QoS violations for the UEs served by the system. The problem formulation is given as:

$$\min_{\{k_{u_i},\, D_{u_i}\}} \quad \sum_{\forall u_i \in \mathcal{U}(t)} a(-\sigma_1 \delta_{u_i}^{\epsilon}(t)) + b\sigma_2 \delta_{u_i}^{l}(t) + c\sigma_3 \delta_{u_i}^{d}(t), \quad (20a)$$

$$\text{subject to} \quad \sum_{u_i \in \mathcal{U}(t)} k_{u_i} \leq K_{\text{total}}, \quad (20b)$$

$$k_{\min} \leq k_{u_i} \leq k_{\max}, \ \forall u_i \in \mathcal{U}(t), \quad (20c)$$

$$a \geq 0, b \geq 0, c \geq 0, a + b + c = 1, \quad (20d)$$

where σ_1, σ_2, and σ_3 are normalization factors, and a, b, c are weighting factors to determine the importance that accuracy and latency QoS violations will have in the system, respectively. Besides, constraint (20b) limits the total number of PRBs that can be allocated to user devices, constraint (20c) limits the minimum and maximum number of PRBs that can be allocated to a single UE, and constraint (20d) determines the bounds of the weighting factors.

In Eq. 20a, the parameter $\delta_{u_i}^{\epsilon}(t)$ is the difference between the accuracy QoS requirement of the application running on UE u_i, i.e., $\epsilon_{u_i}^{a}$, and the moving average accuracy at u_i at time slot t. Similarly, the parameter $\delta_{u_i}^{l}(t)$ is the difference between the latency QoS requirement of the application running on UE u_i, i.e., $l_{u_i}^{a}$, and the moving average latency at u_i at time slot t. The difference between the accuracy QoS requirement of the application running on UE u_i and the moving average accuracy at u_i at time slot t, $\delta_{u_i}^{\epsilon}(t)$, is estimated as:

$$\delta_{u_i}^{\epsilon}(t) = \epsilon_{u_i}^{a} - Q_{u_i}(t), \quad (21)$$

where $Q_{u_i}(t)$ is the moving average of the accuracy experienced by u_i at time slot t, which is estimated as:

$$Q_{u_i}(t) = \gamma\epsilon_{u_i}(t) + (1-\gamma)Q_{u_i}(t-1), \quad (22)$$

where γ acts as a smoothing constant dictating that the average accuracy experienced by u_i's application at time slot t, i.e.,

$\epsilon_{u_i}(t)$, is weighted exponentially more heavily than older ones, i.e., $Q_{u_i}(t-1)$. The average accuracy experienced by u_i's application at time slot t is estimated from the accuracy of video frames inferred locally at u_i and at the edge server, given as:

$$\epsilon_{u_i}(t) = (1-\alpha_{u_i})A_0 + \alpha_{u_i}\sum_{l=1}^{|\mathcal{D}|} A_{D_{u_i}} \times I_{u_i,D_{u_i}}(t), \quad (23)$$

where A_0 is the accuracy of the DNN model running at the user device, and $A_{D_{u_i}}$ is the accuracy of the DNN model $D_{u_i} \in \mathcal{D}$ running at the edge server. The binary function $I_{u_i,D_{u_i}}(t)$ indicates whether the DNN model D_{u_i} at the edge server is the one processing the video frames offloaded by device d_i at time slot t. The binary function $I_{u_i,D_{u_i}}(t)$ is set to 1 if the DNN model D_{u_i} is handling the video frames offloaded by user device d_i and set to 0 otherwise.

The difference between the latency QoS requirement of the application running on UE u_i and the moving average latency at u_i at time slot t is estimated as:

$$\delta^l_{u_i}(t) = l^a_{u_i} - L_{u_i}(t), \quad (24)$$

where $L_{u_i}(t)$ is the moving average of the latency experienced by u_i at time slot t, which is estimated as:

$$L_{u_i}(t) = \gamma l_{u_i}(t) + (1-\gamma)L_{u_i}(t-1). \quad (25)$$

The parameter $l_{u_i}(t)$ in Eq. 25 is the expected latency u_i is experiencing at time slot t. This parameter is estimated from the local processing latency, offloading latency, and latency at the edge server, as:

$$l_{u_i}(t) = \alpha_{u_i}(E[R]^{u_i}_o(t) + \sum_{l=1}^{|\mathcal{D}|} E[R]^{D_{u_i}}_e(t) \times I_{u_i,D_{u_i}}(t)) + (1-\alpha_{u_i})E[R]^{u_i}_l. \quad (26)$$

Finally, the parameter $\delta^d_{u_i}(t)$ in Eq. 20a is the drop rate at time slot t for the frames produced by UE u_i. This parameter is estimated as:

$$\delta^d_{u_i}(t) = \delta^{u_i}_l + \delta^{u_i}_o + \sum_{l=1}^{|\mathcal{D}|} \delta^{D_{u_i}}_e(t) \times I_{u_i,D_{u_i}}(t), \quad (27)$$

where $\delta^{u_i}_l$, $\delta^{u_i}_o$ and $\delta^{D_{u_i}}_e$ are the drop rate at the u_i's local processing queue, u_i's offloading queue, and DNN model D_{u_i} queue at the edge server during time slot t, respectively.

V. Markovian Decision Process Formulation

A typical Markov Decision Process (MDP) is formulated by a 4-tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$, where $\mathcal{S}$ is the set of all possible states, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the state transition probability, and $\mathcal{R}$ is the reward function. These elements are discussed below.

a) State space $\mathcal{S}$: At each time slot t, the system state is a combination of the moving average of the latency and accuracy at each UE, the expected latency at each DNN model running at the edge server, and the information regarding which DNN model at the edge server each UE is relying on for video frame inference. Hence, the system state s_t at a time slot t is defined as:

$$s_t = \{\mathcal{Q}(t), \mathcal{L}(t), \mathcal{E}(t), \mathcal{I}(t), \mathcal{B}(t), \mathcal{F}(t), \mathcal{O}(t)\}, \quad (28)$$

where $\mathcal{Q}(t) = \{Q_{u_1}(t), \ldots, Q_{u_{|\mathcal{U}|}}(t)\}$ is the moving average accuracy for UEs at time slot t, $\mathcal{L}(t) = \{L_{u_1}(t), \ldots, L_{u_{|\mathcal{U}|}}(t)\}$ is the moving average of the expected latency at UEs during time slot t, $\mathcal{E}(t) = \{E[R]^{d_1}_e, \ldots, E[R]^{d_{|\mathcal{D}|}}_e\}$ is the sojourn time experienced at each DNN model at the edge server during time slot t, and $\mathcal{I}(t) = [I_{u_i,D_{u_i}}(t) \mid u_i \in \mathcal{U}(t), D_{u_i} \in \mathcal{D}]$ is an association matrix indicating the current assignments of DNN models to user devices at time slot t. Additionally, $\mathcal{B}(t) = \{B_{u_1}(t), \ldots, B_{u_{|\mathcal{U}|}}(t)\}$ is the current pool of remaining unassigned PRBs at time slot t, $\mathcal{F}(t) = \{F_{u_1}(t), \ldots, F_{u_{|\mathcal{U}|}}(t)\}$ is the frame rate assigned to the user device during time slot t and $\mathcal{O}(t) = \{O_{u_1}(t), \ldots, O_{u_{|\mathcal{O}|}}(t)\}$ is the offloading rate assigned to the user device during time slot t.

b) Action space $\mathcal{A}$: At the beginning of a time slot t, the action space when the system is at a state s_t is:

$$\mathcal{A}_{s_t} = \{(k_{u_i}, D_{u_i}, \alpha_{u_i}) \mid \forall u_i \in \mathcal{U}(t),\ \forall D_{u_i} \in \mathcal{D},\ \alpha_{u_i} \in \mathcal{Z}_\alpha\}, \quad (29)$$

where k_{u_i} is the total number of PRBs to be allocated for UE u_i, D_{u_i} is the DNN model selected to handle, at the edge server, the video frames offloaded by u_i, and where ω_{u_i} is the offloading rate selected for UE u_i, discretized in steps of 0.1 over the set:

$$\mathcal{Z}_\alpha = \{0, 0.1, 0.2, \ldots, 0.9, 1\}. \quad (30)$$

c) State transition function $\mathcal{T}$: At a time slot t, the environment transitions from a state s_t to a state s_{t+1} when an action $a_t \in \mathcal{A}_{s_t}$ is taken according to a transition probability $p(s_{t+1} \mid s_t, a_t) = \mathcal{T}(s_{t+1} \mid s_t, a_t)$, given by a function $\mathcal{T}$ which is unknown.

d) Reward function $\mathcal{R}$: The reward function is designed to guide the PPO reinforcement learning agent decisions aimed at solving the optimization problem formulated in Sec. IV, i.e., minimize the difference between the latency requirement and moving average of the expected latency while maximizing the difference between the moving average of the expected accuracy and the accuracy QoS requirement. In this regard, after performing an action $a_t \in \mathcal{A}_{s_t}$, the agent gets a penalty as feedback, given by:

$$\mathcal{R}_t = -\sum_{\forall u_i \in \mathcal{U}(t)} a(-\sigma_1 \delta^\epsilon_{u_i}(t)) + b\sigma_2 \delta^l_{u_i}(t) + c\sigma_3 \delta^d_{u_i}. \quad (31)$$

In addition, the designed agent is penalized when PRBs available in the system are not allocated to the users. This is aimed at maximizing the utilization of PRBs and improving the performance of UEs during the video frame offloading

process. The additional penalty added to the agent when PRBs available in the system are not allocated is given as:

$$R_{PRB} = \begin{cases} (-P_1) \times \dfrac{K_{total} - K_t}{K_{total} - k_{min}\,|\mathcal{U}(t)|}, & \text{if } K_t < K_{total} \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

where $K_t = \sum_{u_i \in \mathcal{U}(t)} k_{u_i}$ is the total PRBs allocated for all user devices at time t.

VI. DRL Implementation

We use a Proximal Policy Optimization (PPO) framework for solving the optimization problem formulated in Sec. IV. In our implementation, the PPO agent operates in an episodic manner, with each episode corresponding to a system time slot. At each step, the agent observes the current state (Eq. 28), selects an action from the action space (Eq. 29), receives a penalty (Eq. 31), and transitions to the next state, as shown in Algorithm 1.

Algorithm 1 PPO-based Joint Communication Resource Allocation, DNN Model and Offloading Rate Selection

1: Initialize policy parameters θ_0 and value function parameters ϕ_0
2: **for** each episode **do**
3: Reset environment and obtain initial state s_0;
4: **for** each time step $t \in \mathcal{T}$ **do**
5: Sample action $a \in \mathcal{A}_{s_t}$ from policy $\pi_\theta(\cdot|s_t)$
6: Execute action a, observe reward/penalty $\mathcal{R}_t$ and next state s_{t+1}
7: **end for**
8: Update actor and critic networks using the PPO algorithm with collected trajectories
9: **end for**

Furthermore, we design a novel heuristic (see Algorithm 2) that complements the PPO agent during the allocation of communication resources. The variable defined in Line 2 is used to determine which UE must be considered at a given moment. The UEs are sorted in a set based on their offloading rate (Line 3). Line 4 determines the number of PRBs that were not allocated by the PPO agent and are available to be allocated. Lines 5-10 assign to the considered UE the maximum number of PRBs possible that are available to be assigned.

VII. Performance Evaluation

In this section, we evaluate the performance of the proposed solution for jointly allocating communication resources and selecting the most appropriate DNN model at the edge server for user devices. In the scenario considered, the duration of a time slot is set to 1 second. A base station is assumed to cover an area with a radius of 80 m and to provide connectivity to 10 users. The Random Waypoint (RWP) model is used to model the mobility of user devices. User speeds are set uniformly sampled between 1 and 2 meters per slot, and the average dwell time is uniformly sampled between 0 to 5 time slots. The rate in which video frames are produced is set to 10 fps encoded at resolution of 640×640 pixels and color depth of 24 bits/pixel. It is assumed one edge server is available to handle video frames offloaded by user devices. The weighting factors in Eq. 20d are set equally as 1/3.

Algorithm 2 Communication Resource Allocation Heuristic

1: Obtain the amount of PRB resources that were not allocated by the PPO agent and the offloading rate for each UE
2: $index \leftarrow 0$
3: Sort UEs in descending order based on their offloading rate
4: $PRBs_{available} = K_{total} - \sum_{u_i \in \mathcal{U}(t)} k_{u_i}$
5: **while** $PRBs_{available} > 0$ **do**
6: $P \leftarrow \min(k_{max} - k_{u_{index}}, PRBs_{available})$
7: Assign extra P PRBs to the UE u_{index}
8: $PRBs_{available} = K_{total} - \sum_{u_i \in \mathcal{U}(t)} k_{u_i}$
9: $index \leftarrow index + 1$
10: **end while**

The accuracy $\epsilon^a u_i$ and latency $l^a u_i$ QoS requirements for each user device u_i are set to 43% and 100 ms, respectively, while the drop rate is minimized directly rather than constrained to a fixed threshold. The sub-6 GHz (FR1) band is used with subcarrier spacing of 60 kHz and total bandwidth of 100 MHz. Hence, the total number of PRBs is 135, and the bandwidth of each PRB is 720 kHz. For each user device, the maximum number of PRBs that can be assigned is set to 35, along with a minimum of 1. Additionally, the PRB reward-function constant P_1 is set to 0.1 to penalize underutilization of PRBs. The transmission power is $P_t^{u_i} = 23$ dBm. The background noise is set to $P_0 = -114$ dBm. The constant PL_0 in the path loss log-distance model is set to 38 dB at 1 m, and the path loss exponent η is set to 3.76.

The frequency at the edge server is set to 2.4 GHz, and the user device is set to 1.173 GHz. The CPU capacity at user devices per cycle is set to 1/2048 cycles/FLOPs, and the edge server is set to 1/8192 cycles/FLOPs. It is assumed that the CPU resources at the edge server are equally distributed among the DNN models. We assume each user device runs the YOLO11N model locally, which provides an accuracy of 39.5% and has a complexity of 6.5 billion FLOPs (BFLOPs). On the server, the following models are deployed: YOLO11S, YOLO11M, YOLO11L, YOLO11X. These models have accuracies of 47%, 51.5%, 53.4%, and 54.7%, respectively, and complexities of 21.5 BFLOPs, 68 BFLOPs, 86.9 BFLOPs, and 194.9 BFLOPs, respectively [19]. We train the model for a total of 175,000 time slots with an episode and batch size of 80 steps each, with gradients updated after 10 episodes (800 steps). Lastly, we run 30 replications for each experiment and plot the average results with a confidence interval of 95 %.

A. Results

This section discusses the results obtained when the proposed PPO agent is used with and without the proposed heuristic that assigns communication resources that might be available.

Fig. 1a shows the deviation of the obtained average latency from the latency QoS threshold set for the application. As can be seen, the use of the PPO agent without the developed heuristic led to the worst result. This happens due to underutilized communication resources that might increase the offloading latency at some UEs.

Fig. 1b depicts the average latency per frame for processing a video frame locally at the UE. As can be seen, the developed PPO agent combined with the proposed heuristic led to increased local processing latency when compared to the PPO agent without the heuristic that allocated remaining PRBs to UEs. This trend is expected and is explained by the fact that the offloading latency is higher when the developed heuristic to allocate remaining PRBs is not used (see Fig. 1c). In this case, the PPO agent tries to decrease the local processing latency by discarding video frames (see Fig. 1f).

Fig. 1c illustrates the average latency experienced by a frame in the offloading queue of a UE. As can be seen, the average latency decreased when the designed heuristic is used in conjunction with the PPO agent. This first indicates that the PPO agent fails to allocate all communication resources to UEs and leaves some PRBs unallocated, even though the system has been designed to assign a high penalty to the agent when latency QoS violations are present and PRB resources are not allocated in their totality. Hence, it is important to verify if there is any PRB that has not been allocated, and a deterministic allocation must be performed, such as the one done by the proposed Algorithm 2.

Fig. 1d shows the average latency for the processing of video frames at the edge server. As expected, both solutions have comparable performance. However, it is worth noticing that the solution that combines the designed PPO agent with the heuristic to allocate possibly remaining PRBs results in slightly increased remote processing latency as compared to the pure PPO agent allocation. This is because more frames are offloaded and successfully received at the edge server when this approach is used, whereas the system management with the pure PPO agent drops more frames (see Fig. 1f).

Fig. 1e depicts the deviation of the average accuracy from the accuracy QoS threshold implemented in the system. This result shows that the deviation is less when the pure PPO agent is used. However, it is important to note that the observed accuracy is estimated only for video frames that are successfully processed. When accounting for frame drops (see Fig. 1f), this solution could show worse performance.

Finally, Fig. 1f shows the frame drop rate when both considered solutions were used. As can be seen, the drop rate is higher when the pure PPO agent is used. This is because some frames are dropped in the offloading queue to keep the system stable. The drop rate when the PPO agent is combined with the proposed heuristic is lower, as any unused PRB is allocated to UEs, prioritizing UEs with a high offloading rate.

VIII. Conclusion

This paper addressed the problem of jointly optimizing the offloading rate, communication resources, and DNN model in edge-aided computer vision systems. We proposed a mathematical framework to model the considered problem, as well as an optimization formulation to jointly optimize different aspects in a distributed computer vision system. The optimization formulation aimed at minimizing QoS violations in terms of accuracy and latency. Besides, we proposed a PPO-based agent to solve the formulated optimization problem and determine the offloading rate, number of communication resources, and the DNN model at the edge server for handling video frames of each user equipment. We also proposed a heuristic to be used in conjunction with the PPO agent aimed at allocating, in an additional step, any communication resource that might not be in use by UEs. The obtained results showed the feasibility of the proposed solution in maintaining drop rate at low levels while keeping latency and accuracy at acceptable levels.

As future work, we plan to optimize the proposed solution to use multiple DNN models to handle the video frames of the same video flow, as well as to minimize energy consumption for local, offloading, and remote processing.

References

[1] K. Aminiyeganeh, R. W. Coutinho, and A. Boukerche, "Iot video analytics for surveillance-based systems in smart cities," *Computer Communications*, vol. 224, pp. 95–105, 2024.

[2] H. Wu *et al.*, "R-mddqn: A v2v-based secure computation offloading algorithm for video analytics in vehicle edge networks," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 7, pp. 10 209–10 224, 2025.

[3] M. Ghasemi *et al.*, "Real-time video analytics for urban safety: Deployment over edge and end devices," in *Proc. of the 10th ACM/IEEE Symposium on Edge Computing (SEC)*, 2025, pp. 1–17.

[4] M. Lesko-Krleza, R. W. L. Coutinho, and Y. R. Shayan, "An analytical model for distributed video analytics in mobile computer vision-based systems," in *Proc. of the 23rd Int'l Symposium on Network Computing and Applications (NCA)*, 2025, pp. 69–73.

[5] R. W. Coutinho and A. Boukerche, "Distributed video analytics for iot intelligent systems," *Ad Hoc Networks*, vol. 184, p. 104149, 2026.

[6] T.-T. Nguyen *et al.*, "Octopinf: Workload-aware inference serving for edge video analytics," in *Proc. of the IEEE Int'l Conf. on Pervasive Computing and Communications (PerCom)*, 2025, pp. 128–137.

[7] V. Sohn *et al.*, "Joint frame drop and object detection task offloading for mobile devices via rl with lyapunov optimization," *IEEE Transactions on Mobile Computing*, vol. 24, no. 7, pp. 6168–6182, 2025.

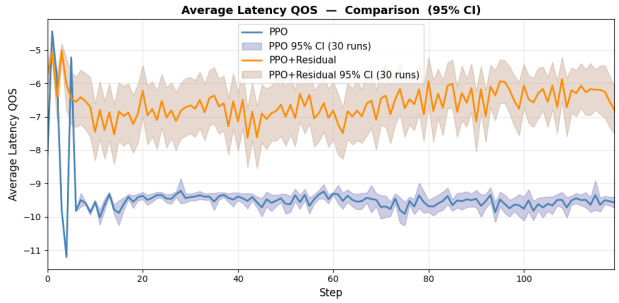
[8] J. Chen *et al.*, "A lyapunov optimization-based online algorithm for scheduling cloud-edge collaborative real-time video stream analytics tasks," *IEEE Internet of Things Journal*, vol. 12, no. 14, pp. 27 713–27 727, 2025.

[9] Y. Wang *et al.*, "JSFBA: joint segment and frame bitrate adaptation for real-time video analytics," *IEEE Transactions on Mobile Computing*, vol. 24, no. 6, pp. 4874–4888, 2025.

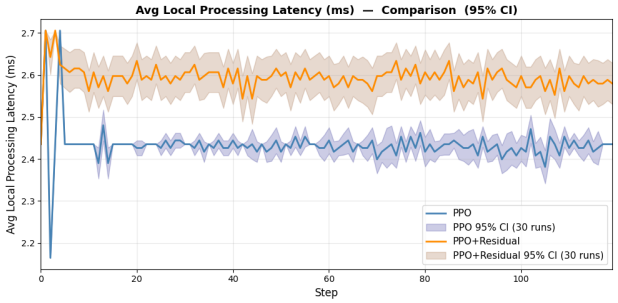
[10] T. Bai *et al.*, "Content-aware joint knob configuration and resource allocation for edge video analytics," *IEEE Transactions on Mobile Computing*, vol. 24, no. 6, pp. 5536–5550, 2025.

[11] Z. He *et al.*, "Qoi-aware configuration adaptation and heterogeneous resource allocation for edge video analytics," *IEEE Internet of Things Journal*, vol. 12, no. 12, pp. 19 215–19 230, 2025.

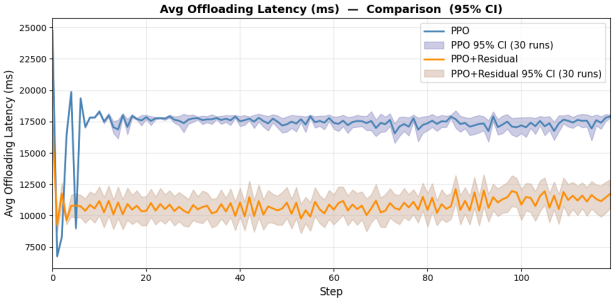
[12] Y. Zheng *et al.*, "Adaptive network slicing and hierarchical data management for edge-assisted multi-task video enhancement," *IEEE Network*, vol. 39, no. 3, pp. 63–71, 2025.



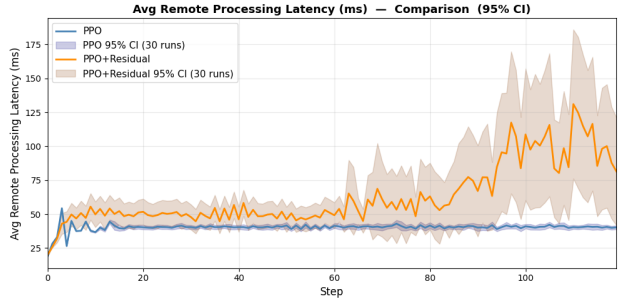
(a) Deviation from Latency QoS



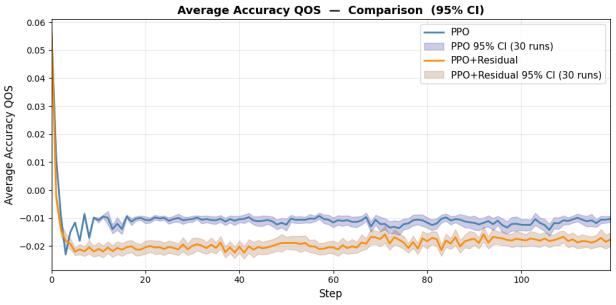
(b) Avg. Local Processing Latency



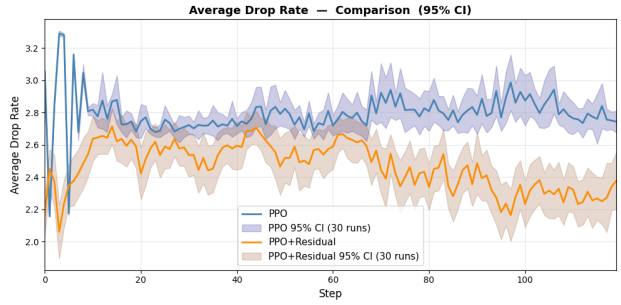
(c) Avg. Offloading Latency



(d) Avg. Remote Processing Latency



(e) Deviation from Accuracy QoS



(f) Average Frame Drop Rate

Fig. 1: Obtained results.

[13] J. Li *et al.*, "Joint video frame scheduling and resource allocation for device-edge collaborative video intelligent analytics," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2025, pp. 1–6.

[14] W. Qian, R. W. Coutinho, and A. Boukerche, "Communication resource allocation and multi-dnn inference optimization in edge computing-aided video analytics," *Computer Communications*, vol. 257, p. 108607, 2026.

[15] ——, "A novel drl-based orchestrator for edge computing resource allocation in mobile computer vision systems," *Computer Networks*, vol. 285, p. 112380, 2026.

[16] D. K. Alqahtani *et al.*, "Ecore: Energy-conscious optimized routing for deep learning models at the edge," in *Proc. of the IEEE Int;l Conf. on Pervasive Computing and Communications (PerCom)*, 2026, pp. 1–10.

[17] M. Setayesh *et al.*, "Joint prb and power allocation for slicing embb and urllc services in 5g c-ran," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–6.

[18] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz," 3rd Generation Partnership Project, Tech. Rep. TR 38.901, 2025.

[19] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics