Intelligent UAV Surveillance Networks with Edge-Assisted Execution of Computer Vision Tasks

N. Varshney, *Member, IEEE,* C. Puligheddu, *Member, IEEE,* C. Casetti, *Senior Member, IEEE* S. De, *Senior Member, IEEE.* C. F. Chiasserini, *Fellow, IEEE*

***Abstract*—Developing an edge-assisted UAV surveillance system that ensures low-latency, high-accuracy execution of computer vision tasks while optimizing energy consumption and network resources remains a complex challenge. In this paper, we address the limitations of existing research by leveraging image semantics, image ensemble processing, and mmWave UAV-edge channel statistics. We do so by focusing on joint optimization of UAV speed, camera images per second (IPS) rate, offloading policy, and transmission rates with the aim to minimize the UAV's energy consumption. Given the NP-hardness of the problem, we propose an algorithmic solution, named Intelligent UAV Network (IntUNe), which is based on an innovative constrained reinforcement learning strategy that dynamically and effectively adjusts to real-time conditions. Our results demonstrate that, in delay-constrained UAV surveillance networks, IntUNe closely matches the optimum in small-scale scenarios, and it increases inference accuracy by significantly reducing violations of the accuracy constraint by up to 96.19% compared to state-of-the-art alternatives. Also, it reduces UAV propulsion energy consumption by 34.67% and total UAV energy consumption by up to 37.3%.**

***Index Terms*—Edge computing, task offloading, UAVs**

I. Introduction

Uncrewed aerial vehicles (UAVs) are essential to surveillance operations, like smart farm monitoring, traffic monitoring, and disaster management, owing to their agility and adaptability in navigating diverse environments. Indeed, with advanced sensing and communication technologies like cameras and radars, UAVs can execute complex computer vision (CV) tasks such as object detection, tracking, and obstacle avoidance. However, fully autonomous UAV surveillance relies heavily on efficient, low-latency object detection capabilities and sufficient UAV energy resources. The hardware required for such tasks often poses challenges due to its bulkiness and energy-intensive nature, particularly for small-sized UAVs.

In outdoor operations, offloading CV tasks to a ground control station (GCS), equipped with a high-end server, can significantly reduce processing time while maintaining detection accuracy. Additionally, the required transmission power is generally lower than computational power for object detection [1], which makes the offloading of CV tasks to the GCS an attractive alternative to UAV onboard processing. On the other hand, the quality and reliability of the channel directly impacts data transmission efficiency, making it essential to consider the communication channel condition when deciding whether to offload images to the GCS. To address this issue in multi-UAV-aided surveillance networks, several works have adopted reinforcement learning (RL)-based approaches, which mainly fall into two categories: binary offloading and partial offloading [2]. In the former case, the CV task is either executed locally on the UAV or offloaded to an edge server. In the latter case, the task is split in local computing and execution at the edge. However, binary offloading is often preferred due to the energy-intensive nature of running object detection algorithms on the UAV.

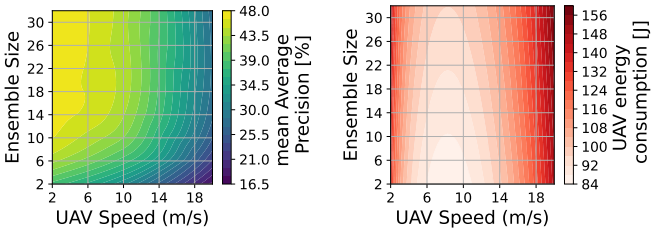

Fig. 1. Video object detection accuracy (left) and UAV energy consumption (right) as UAV speed and ensemble size vary. The plots have been generated from interpolated data obtained using experimental results as detailed in Sec. VI-A (left), and using the energy consumption model in Sec. III-E (right).

Despite the recent advances, developing a reliable system that anticipates future demands and enhances UAV surveillance performance by intelligently adapting multiple parameters remains challenging. Indeed, while CV task offloading and transmission rate optimization have been widely studied [3], primarily focusing on reducing data processing delays and energy consumption, *the existing research has limitations and significant challenges still need to be addressed.*

Firstly, most studies focus on UAV energy optimization for object detection and transmission but overlook propulsion energy. While prior work optimizes total UAV energy via trajectory, transmission power, and resource allocation, surveillance UAVs often follow predetermined paths (Fig. 2) [4]. Instead, dynamically optimizing UAV speed along with the offloading strategy and transmission rates can enhance the energy efficiency, extend battery life, and maximize coverage of the UAV. Also, *UAV speed impacts object detection accuracy*, thus introducing another layer of complexity.

Secondly, optimizing UAV speed and images per second (IPS) rate of the camera onboard UAV is crucial for efficient surveillance. Object detection is pivotal to accurately identifying target objects within images, and its performance metrics – including inference accuracy, energy efficiency, and latency – are influenced by IPS and UAV speed. While traditional

N. Varshney is with Birla Institute of Technology and Science, Pilani Hyderabad Campus, 500078 Hyderabad, India (e-mail: nancy.varshney@hyderabad.bits-pilani.ac.in). C. Puligheddu, and C. Casetti are with Politecnico di Torino, 10129 Torino, Italy (e-mail: corrado.puligheddu@polito.it; claudio.casetti@polito.it). S. De is with the Department of Electrical Engineering, Indian Institute of Technology Delhi, 110016 New Delhi, India (e-mail: swadesd@ee.iitd.ac.in). C. F. Chiasserini is also with CNIT, Parma, Italy, and Chalmers University of Technology, Sweden (e-mail: carla.chiasserini@polito.it).

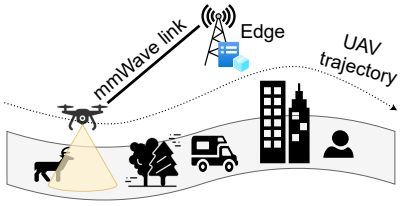

Fig. 2. A UAV traversing a fixed trajectory and connecting to the edge by a highly directional mmWave link.

object detection methods process each image independently, analyzing image groups enhances accuracy by leveraging contextual and temporal coherence [5]. This requires dynamic IPS adjustments to adapt to environmental changes. A higher IPS improves accuracy but increases processing load and energy consumption, while UAV speed impacts both detection quality, UAV energy consumption, and coverage. Although 8 m/s minimizes propulsion energy [6], high speeds cause image jitter, reducing accuracy. Thus, *jointly optimizing IPS and UAV speed is essential to balance accuracy, energy efficiency, and coverage*. An example is shown in Fig. 1, which highlights the joint impact of UAV speed and ensemble size on the mean average precision (mAP) of object detection accuracy and UAV's energy consumption.

Thirdly, while channel statistics have been used for binary offloading and energy-delay optimization [7], *the use of image semantics has not yet been considered.* We address this by categorizing images based on criticality. As noted in, e.g., [8], surveillance videos often contain redundant images with no important objects, which we denote as low-criticality images, while some images contain objects of high importance, which we denote as high-criticality images. Although high-criticality images are fewer, they demand timely and accurate analysis to meet the strict latency and reliability requirements of UAV systems. This typically requires higher IPS rates (i.e., larger ensembles), resulting in increased processing delays and energy usage. In contrast, low-criticality images can be handled with reduced computational effort or delayed offloading, conserving energy without compromising overall objectives. By incorporating criticality, the UAV can adjust its IPS, speed, and offloading strategy, ensuring a more effective balance between detection accuracy, responsiveness, and energy efficiency.

Lastly, existing work predominantly considers sub-6 GHz links for offloading tasks and aims to minimize uplink transmission time [9]. Transmission time can further be minimized by using high bandwidth millimeter wave (mmWave) links. However, such channels are highly susceptible to blockages, resulting in non-line-of-sight (NLoS) conditions lasting several seconds. During these periods, local image processing on the UAV may again be needed to ensure timely detection, at the cost of an increased energy consumption at the UAV.

A. Our approach

To address the above challenges and ensure timely detection while maintaining required object detection accuracy, our approach foresees intelligently offloading CV tasks from the UAV to the edge by considering the following key factors:

- Accounting for image criticality: We categorize captured images into different criticality levels based on their content and optimize their handling and processing accordingly. This ensures efficient resource utilization and timely detection;
- Employing an ensemble of images for higher detection accuracy: High-criticality of images and rapid environmental changes necessitate adjustments in the IPS rate operation to achieve higher detection accuracy. We thus consider the IPS rate as a decision variable that eventually controls the ensemble size for a given time period;
- Controlling UAV speed: Besides the IPS rate, adjusting UAV speed is also crucial for maintaining detection quality and capturing finer details. Additionally, optimizing UAV speed is essential to maximize the distance covered while conserving energy. Hence, we control the UAV speed so as to optimally trade off energy consumption with detection accuracy;
- Use of mmWave backhaul channel: We leverage the directional mmWave links between the UAV and the edge to provide large bandwidth, thereby mitigating latency issues due to large image transfers;
- Managing edge load: When the edge server is overloaded, local image processing becomes crucial. We thus dynamically adjust the offloading decision based on the edge server load to maintain system efficiency;
- Optimizing the offloading scheme and image transmission rate: This is vital to minimize UAV energy consumption while meeting target latency and accuracy. By fine-tuning the offloading rate and transmission rate, we ensure that critical data is processed promptly under any channel conditions.

Overall, our approach dynamically adjusts the offloading strategy, UAV speed, and camera IPS rate in real time to meet object detection accuracy and latency requirements, while accounting for image criticality based on image semantics and mmWave link conditions. Under favorable channel conditions, the UAV can increase its IPS and speed, offload more images, and transmit them at higher rate to fully utilize the available bandwidth and reduce latency. Instead, in case of channel degradation or high edge server load, the system can lower the IPS and UAV speed to reduce processing and transmission load, save energy, and prevent delays. The transmission rate is also tuned to the channel state for link adaptive transmission.

B. Summary of novel contributions and paper organization

By considering the above factors, we develop an efficient solution to be run at UAVs for edge-assisted surveillance. Specifically, the main contributions of this work are as follows:

- We mathematically formulate the problem of intelligent execution of CV tasks in UAV surveillance networks to (i) minimize the UAV's consumption of propulsion and processing energy, and (ii) maximize the total distance covered by the UAV during its flight, by controlling UAV speed, IPS rate of the camera on board the UAV, and the offloading strategy. Our approach ensures accurate, low-latency inference and uniquely accounts for image criticality levels, optimizing over image ensembles rather than individual frames.
- To cope with the complexity of the above problem, we first decompose the optimization problem for intelligent execution of CV tasks across the entire time horizon into

smaller subproblems over shorter durations. Subsequently, we introduce Intelligent UAV Network (IntUNe), a novel real-time decision-making framework for UAV-based aerial surveillance, to be run at UAVs to solve each subproblem. The solution is built on an extended Constrained Soft Actor-Critic (CSAC) reinforcement learning framework. Unlike standard CSAC, our method handles a multi-dimensional discrete action space, jointly optimizing IPS rate, UAV speed, and offloading decisions, while explicitly enforcing practical constraints on energy, latency, and inference accuracy at every time step through indicator-based cost functions.

• We present experimental findings that illustrate the impact on achievable accuracy as a function of UAV speed and IPS rate using a state-of-the-art video object detection model.

• We evaluate IntUNe through an extended numerical analysis. Our findings reveal that, by accounting for UAV speed, IPS rate, offloading decisions, and transmission rates, a 34.67% reduction in UAV propulsion energy and an up to 37.3% reduction in total UAV energy consumption can be achieved compared to the state-of-the-art solution in [10]. Additionally, our approach results in a decrease in violations of inference accuracy constraints by up to 96.1% relative to [10].

In the following, Sec. II discusses related work, highlighting the novelty of our study. Sec. III presents the system dynamics, while Sec. IV formulates the problem for intelligent execution of CV tasks by UAVs and proves its NP-hardness. Sec. V introduces our algorithmic solution, named IntUNe; its performance is shown and benchmarked against state-of-the-art alternatives in Sec. VI. Finally, Sec. VII concludes the paper.

II. Related Work

Energy-efficient UAV surveillance has attracted significant attention [11], with some works focusing on intelligent UAV navigation and control [12] while others focusing on joint optimization of trajectory and resource allocation [13]. For instance, [11] introduces a cutting-edge UAV surveillance system endowed with cognitive capabilities, enabling the rapid identification of critical situations. [9] explores various modes for image processing and feature extraction, considering factors like network conditions and processing time to determine the optimal offloading strategy. [3] computes the binary offloading decision of CV tasks as a function of network and computation load parameters of a UAV. This prior art highlights that there exists a trade-off between power consumption, computational complexity, and processing time when running high-end CV algorithms on UAVs. However, although using a low-capability edge server reduces power consumption, it increases processing time, which is critical for delay-intolerant UAV operations.

Similarly, [1] compares the processing delay and energy consumption of a face recognition algorithm over videos of different lengths on a Raspberry Pi unit onboard UAV and on a laptop as the GCS server at sub-6 GHz, verifying that the transmission power is generally lower than the computational power required for object detection. A comprehensive survey of decision-making approaches considering factors such as service latency, energy consumption, and processing delay can be found in [2]. [3] optimizes offloading decisions for CV tasks to minimize both the task execution time and UAV energy consumption using deep RL while considering the service load at the edge. However, [3] does not consider various other time delays, such as transmission and propagation time of large-sized packets or waiting time in case of unsuccessful transfers to the edge. Nevertheless, it is the closest state-of-the-art to our approach, and, hence, we use a modified version of it as a benchmark, as explained in detail in Sec. VI-D.

With regard to multi-UAV networks, [7] proposes a deep RL-based computation binary offloading scheme, to minimize energy consumption and task execution delay. [14] proposes a differential-rate SAC deep RL algorithm to dynamically adapt offloading rates in UAV-assisted device–edge–cloud environments to jointly minimize task completion cost and UAV energy consumption under resource constraints. Building on this, [15] presents an enhanced dependency-aware computation offloading strategy within a device–edge–cloud collaborative architecture to jointly optimizes delay, energy consumption, and UAV endurance using an improved actor–critic DRL method. More recently, [10] presents a Lyapunov-based framework for delay-aware and energy-efficient task offloading from mobile devices to the cloud. Likewise, [16] introduces a RL–based task offloading solution for UAV swarms, where trust region policy optimization is used to manage computation offloading to ground edge servers under stringent latency and energy constraints. However, again, these approaches primarily address energy–delay trade-offs in offloading decisions at sub-6 GHz frequencies, without accounting for image semantics, the vulnerability of high-frequency links (despite their greater bandwidth), or the impact of UAV speed and IPS rate on CV detection accuracy in challenging environments.

Finally, a preliminary version of our work was presented in our conference paper [17], where we optimized just the binary offloading strategy at mmWave, while incorporating image semantics, for each frame while assuming fixed UAV speed and IPS, and we showed a limited set of results under simplified scenarios.

Progress beyond the state of the art. We present an approach that incorporates multiple levels of image criticality based on image semantics, which drives the optimization of various UAV parameters. To enhance object detection accuracy, we process CV tasks across an ensemble of images. Unlike previous approaches that focus solely on optimizing offloading strategies and transmission rates for UAV processing energy efficiency, our solution also adapts UAV speed and camera IPS rate in real-time. This adjustment ensures that the object detection accuracy and latency thresholds for CV tasks are met, while also optimizing UAV propulsion and processing energy. Therefore, *our primary focus is to jointly adapt the offloading strategy, UAV speed, and IPS rate, so that UAV energy efficiency and the effectiveness of CV detection are maximized. Also, we account for the challenges posed by mmWave channel conditions and blockage severity—factors that have often been overlooked in prior art.*

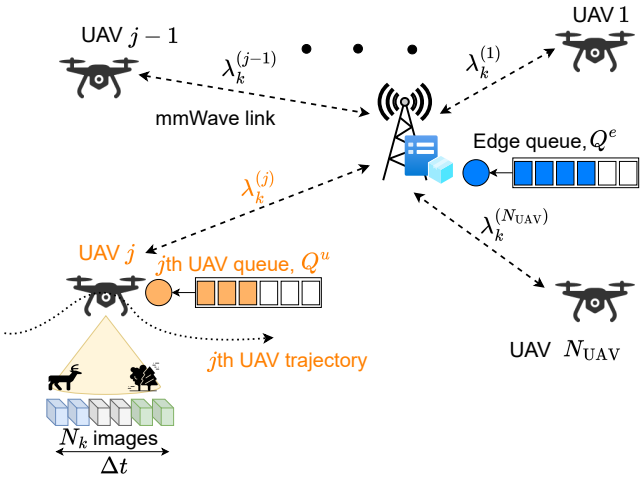

Fig. 3. Illustration of the system with N_{UAV} UAVs connected to the edge server via mmWave links in time period k. Each UAV transmits collected image data to the edge server while following its designated trajectory.

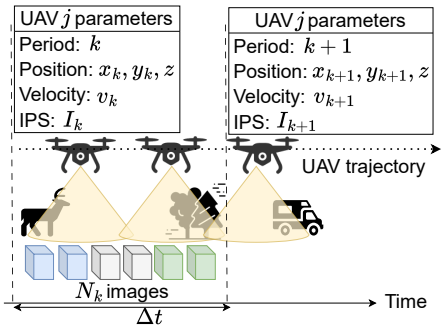

Fig. 4. Update of UAV j's parameters over two consecutive time periods.

III. System Dynamics and Model

This section presents the key components of the edge-assisted UAV surveillance network under study. We first give an overview of the system model (Sec. III-A), and then detail the model of the mmWave communication link (Sec. III-B) and the image criticality (Sec. III-C). Finally, we describe the task load, queuing, and processing behavior at the edge server (Sec. III-D), and the UAV-side task queuing, processing, and energy consumption model (Sec. III-E).

A. Overview

We consider an outdoor scenario with N_{UAV} camera-equipped surveillance UAVs connected to a single GCS hosting an edge server[1], as shown in Fig. 3. Each UAV captures images and can choose to process them locally using the onboard processor or offload them to the edge. Also, we consider that each UAV has a learning agent that makes this decision independently, since reliable communication and continuous coordination with other UAVs may be hard to guarantee. Our study focuses on the operational dynamics of a specific UAV, tagged as UAV j ($j{=}\{1,\ldots,N_{\text{UAV}}\}$). Despite operating independently, as explained later, this UAV accounts for statistical data regarding the offloading rate of images to the edge server from other UAVs in the system.

Next, we consider that UAV j follows a predetermined path (that can be a straight line or an orbit [4]) at a constant altitude z to follow mission instructions. Maintaining a fixed altitude allows the UAV to conserve energy by avoiding constant adjustment to vertical movement [6]. Moreover, having a fixed altitude aids to achieve reliable image resolution and sensor readings [18]. A predetermined flight path ensures that the UAV consistently monitors the same areas at expected times, which is important for real-time surveillance and data collection. The UAV is equipped with a battery capable of supplying a maximum energy of E_{M}, which is utilized for local image processing, image offloading, and UAV navigation. We segment the total flight duration T_F into intervals of Δt seconds, resulting in $K{=}\lceil T_F/\Delta t\rceil$ time periods. During each period k, the UAV has a position (x_k, y_k, z), moves with speed $v_k{\in}\mathcal{V}$, and captures images at rate $I_k{\in}\mathcal{I}$. These parameters are updated at every period k, as illustrated in Fig. 4, thereby explicitly modeling the temporal evolution of the UAV's state. Thus, the number of images captured in interval k is $N_k{=}\lceil I_k\Delta t\rceil$, forming one ensemble.

The N_k images captured by the UAV in period k, of duration Δt, can be processed either onboard or offloaded to the edge server. Both the UAV and the edge maintain separate first-in-first-out (FIFO) processing queues, denoted as Q^e and Q^u, resp. (Fig. 3). Images selected for offloading are transmitted over the UAV-to-GCS backhaul link at a fixed power P_t. Let $\lambda_k^{(j)}$ as be the image offloading rate of UAV j in period k where $\lambda_k^{(i)}{\in}\{0,\mathcal{I}\}$, optimized based on detection accuracy requirements, current channel conditions, and edge load. The backhaul communication operates on a bidirectional mmWave link with a bandwidth of B MHz. These highly directional links are inherently robust against interference. Also, the UAV carries its onboard processor, which has limited computational capabilities, and hence longer service time than the server at the edge, as confirmed by the study in [1]. The edge server or the UAV processor processes the images and provides feedback to the corresponding UAV regarding image criticality, detection accuracy, channel status, and edge load.

B. MmWave link

To provide a mathematical model for the signal-to-noise ratio (SNR) over the mmWave backhaul link, we define a discrete-time Markov chain with two states: LoS (Line-of-Sight) and NLoS. Let X_t denote the state of the system at time t, where $X_t{=}1$ ($X_t{=}0$) represents LoS (NLoS). The probability of transition from LoS to LoS is $P(X_{t+1}{=}1|X_t{=}1){=}p$, from LoS to NLoS is $P(X_{t+1}{=}0|X_t{=}1){=}1{-}p$, from NLoS to NLoS is $P(X_{t+1}{=}0|X_t{=}0){=}q$, and from NLoS to LoS is $P(X_{t+1}{=}1|X_t{=}0){=}1{-}q$.

Further, the probability of a LoS link in an urban environment depends on the ratio of built-up land area to the total land u_1, the mean number of buildings per unit area u_2

[1]While we consider a single GCS for presentation simplicity, the IntUNe framework is inherently generalizable to multi-GCS scenarios. Moving between GCS coverage zones primarily manifests as dynamic variations in the mmWave channel conditions, to which our RL agent is already able to seamlessly adapt.

(buildings/km^2), and the parameter u_3 describing the buildings heights distribution according to Rayleigh probability density function. The expression is given by [19], which is based on the ITU Recommendation for propagation data and prediction methods in the 3-60 GHz frequency range [20]:

$$P_{LoS} = \prod_{n=1}^{m} \left[1 - \exp\left(- \frac{\left[z - \frac{(n+0.5)z}{(m+1)} \right]^2}{2u_3^2} \right) \right] \quad (1)$$

where $m{=}(r_{2D}\sqrt{u_1 u_2} - 1)$, r_{2D} is the UAV-GCS ground distance, and we consider the UAV height z as constant. In steady-state conditions, given P_{LoS}, the transition probabilities p and q are related as follows: $p{=}1{-}\frac{(1-P_{LoS})(1-q)}{P_{LoS}}$. The SNR values are generated based on the current state. Let γ_t denote the SNR distribution parameter at time t. If $X_t{=}1$ (LoS), then $\gamma_t{\sim}\mathcal{N}(\bar{\gamma}_{\text{LoS}})$; if $X_t{=}0$ (NLoS), then $\gamma_t{\sim}\mathcal{N}(\bar{\gamma}_{\text{NLoS}})$. Here, $\bar{\gamma}_{\text{LoS}}$ ($\bar{\gamma}_{\text{NLoS}}$) is the mean SNR value for the LoS (NLoS) scenario, defined as [21], [22]: $\bar{\gamma}_{\text{LoS/NLoS}} = \frac{|\mathbf{A}_{GCS}\bar{\mathbf{h}}\mathbf{A}_{UAV}|^2 P_t}{\sigma^2}$ and $\bar{\mathbf{h}} = \sqrt{N_t N_r}\bar{\alpha}\mathbf{a}_r\mathbf{a}_t^H 10^{\frac{-\text{PL}}{10}} \in \mathbb{C}^{N_r \times N_t}$. Also, $\bar{\alpha}$ is the mean fading parameter taking on the value $\bar{\alpha}_{\text{LoS}}$ ($\bar{\alpha}_{\text{NLoS}}$) for the LoS (NLoS) case, while $\mathbf{a}_t$ ($\mathbf{a}_r$) is the transmit (receive) array response vector. In our mmWave backhaul model, both the UAV and the GCS employ directional beamforming using planar antenna arrays. The UAV features a $\sqrt{N_t}{\times}\sqrt{N_t}$ transmit array and GCS employs $\sqrt{N_r}{\times}\sqrt{N_r}$ receive array. Consistent with common practice in mmWave LoS/NLoS modeling focused on large-scale propagation effects, we assume perfect beam alignment between the transmitter and receiver. Under this aligned-beam assumption, the array response vectors satisfy $|\mathbf{a}_r\mathbf{a}_t^H|{=}1$, and therefore the total beamforming gain reduces to $N_t \cdot N_r$ [22]. This gain is embedded in the above channel expression $\sqrt{N_t \cdot N_r}\,\mathbf{a}_r\mathbf{a}_t^H$.

For the mean 3D GCS-UAV distance D, the path loss factor PL (in dB), for LoS and NLoS conditions, is given by [21]: PL$=61.4 + 20\log_{10}(D) + \mathcal{N}(0, 33.64)$, and PL$=72.0 + 29.2\log_{10}(D){+}\mathcal{N}(0, 75.69)$, respectively. Given advanced transmission schemes that allow achieving Shannon's limit, if $\gamma_t{\in}(\gamma^i, \gamma^{i+1}]$, the UAV transmission rate (in bits/s) is given by: $R_t{=}B\log_2(1{+}\bar{\gamma}^i)$ where $\bar{\gamma}^i$ is the mean SNR of the i-th SNR quantization interval ($i{=}\{1, \ldots, M\}$), with M being the number of quantization levels.

C. Image criticality

Images captured by the UAV are classified into different levels of criticality based on their content. Importantly, different levels of criticality often require varying accuracy levels in detection. Higher criticality levels necessitate higher accuracy thresholds to minimize the risk of false positives or false negatives, hence they typically require more resources. Also, the UAV expects feedback for each image within a specified delay time, including the image's criticality status and an acknowledgment of reception.

Therefore, we represent criticality of the l-th image in the k-th period, denoted by $\chi_{k,l}$, within a finite set of M_c criticality degrees, i.e., $\chi_{k,l}{\sim}\{\chi^1, \chi^2, \ldots, \chi^{M_c}\}$, with 1 being the lowest and M_c the highest. We assign one criticality level to each time-correlated image using an M_c-state MDP with the following transition probability matrix:

$$P_c = \begin{bmatrix} p_1 & 1-p_1 & 0 & \cdots & 0 \\ \frac{1-p_2}{2} & p_2 & \frac{1-p_2}{2} & \cdots & 0 \\ 0 & \frac{1-p_3}{2} & p_3 & \frac{1-p_3}{2} & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1-p_{M_c} & p_{M_c} \end{bmatrix}.$$

Each p_i denotes the probability of remaining in state i, with transitions to adjacent states defined by $(1{-}p_i)/2$. This matrix structure captures the dynamics of transitioning between states in the MDP, facilitating the generation of state values based on the specified probabilities.

Each criticality is associated with a different threshold of detection accuracy $A_T(\chi)$ and all images have a constant delay budget T_W. Also, due to fixed camera resolution, all images have the same size L. Thus, the image processing times by an object detection algorithm at the UAV processor and at the GCS server are fixed, irrespective of the image's criticality.

The processing times at the UAV and the edge are denoted by t^u and t^e, respectively. As explained in Sec. I, the achievable accuracy of an ensemble is a function of the ensemble size and UAV speed, represented by $A_k{=}f(N_k, v_k)$. As the relationship is highly non-linear and depends on factors such as UAV speed, motion blur, and IPS rate, it cannot be derived analytically. We thus characterize it with a deep neural network–based CV method and determine the mapping experimentally, as shown in Sec. VI-A. Furthermore, the total latency of image l of ensemble k must be less than the delay budget T_W. If the latency exceeds this threshold, the image is considered timed out, which impacts the ensemble size and, hence, the detection accuracy. Specifically, if N'_k images are dropped, then the achievable accuracy of detection accuracy is adjusted to $A_k{=}f(N_k{-}N'_k, v_k)$.

D. Computational load, task queuing, and processing time at the edge

Given that the system has N_{UAV} UAVs, in period k, UAV i selects its image offloading rate $\lambda_k^{(i)}{\in}\{0, \mathcal{I}\}$, depending on accuracy requirements, channel conditions, and edge traffic load. The total arrival rate at the edge server, as seen by the j-th UAV from the rest of the UAVs in period k, is the sum of the image offloading rates of all UAVs except for UAV j, i.e., $\lambda_k{=}\sum_{i=1, i\neq j}^{N_{\text{UAV}}} \lambda_k^{(i)}$.

To gain insights into the distribution of the total arrival rate at the edge, we simulated the inter-arrival times at the edge resulting from the offloading of tasks from all UAVs in the surveillance network and analyzed their distribution. We varied the number of UAVs ($N_{\text{UAV}}{=}\{5, 10, 15, 20, 25, 30\}$), with each scenario comprising 2,000 time slots. Offloading rates for each UAV over time were generated with a different random distribution for each UAV and independently from other UAVs. We then computed the inter-arrival times between successive UAV arrivals at the edge. As shown in Fig. 5, the obtained inter-arrival time fits an exponential distribution. Thus, we can assume that UAV j sees the arrival of the tasks at the edge

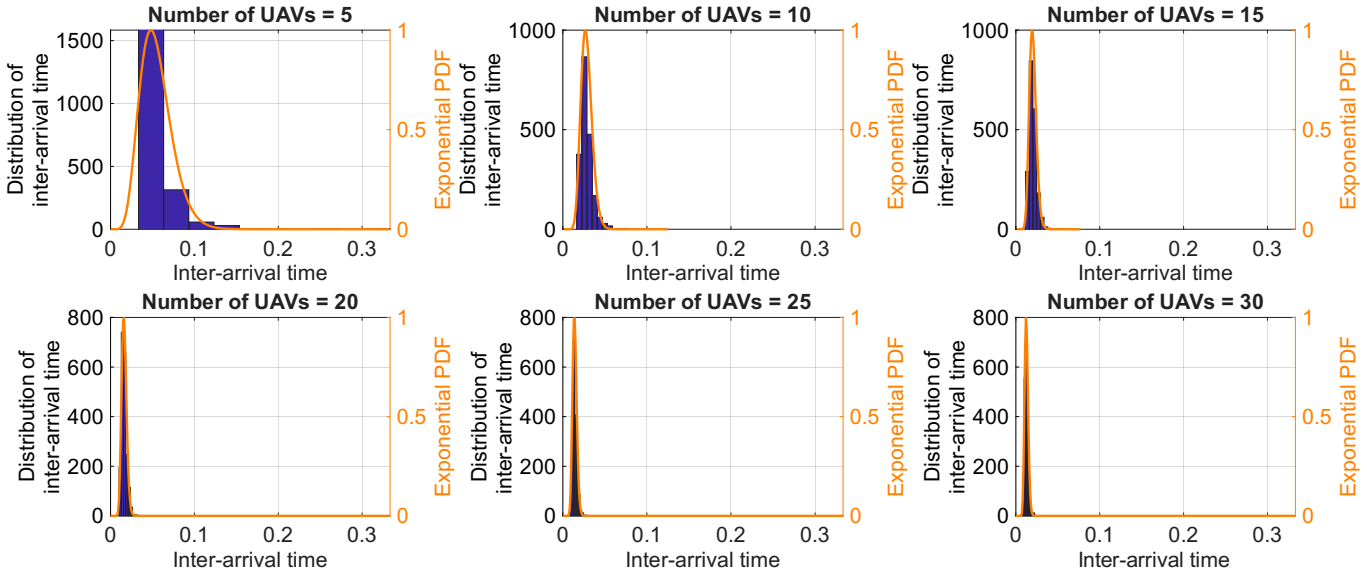

Fig. 5. Distribution of inter-arrival time at edge server as a function of the number of UAVs in the system: experimental probability distribution function (PDF) in blue and exponential fit in orange.

from the rest of the UAVs as a Poisson-distributed process, with mean arrival rate denoted by $\lambda_k{=}\lambda\,, \forall k$.

Given the above observations and considering a fixed computational resource allocation at the edge (hence a fixed service time t^e), the image processing system can be modeled as an M/D/1 queue with service rate $\mu^e{=}1/t^e$ tasks/s. During period k, if the decision is to offload, $N_k{=}\lceil I_k \Delta t \rceil$ image processing requests (or tasks) arrive at the edge from the j-th UAV at equal inter-arrival time of $1/I_k$. The captured images are offloaded as soon as possible and queued at the edge. Therefore, we implement a FIFO rule for processing these queued images, although other schemes like parallel processing could be considered as well.

Additionally, the number of task requests arriving from the other UAVs at the edge during the inter-arrival time of the j-th UAV is λ/I_k. Let $Q^e_{k,l}$ denote the number of tasks in the queue at the edge when image l is captured at UAV j in period k. The number of task requests arriving from the rest of the UAVs at the edge during the inter-arrival time of images at the j-th UAV is λ/I_k. Therefore, the sum of waiting and processing delay of image l is

$$\tilde{t}^e_{k,l} = \frac{Q^e_{k,l} + \lambda/I_k + 1}{\mu^e}. \tag{2}$$

Then, when image $l{+}1$ is captured, the queue is updated using the following relation: $Q^e_{k,l+1}{=}\max(0, Q^e_{k,l}{-}\mu^e/I_k{+}\lambda/I_k{+}1{-}N'_k)$ where N'_k represents the number of dropped tasks at the edge whose waiting time exceeds a certain threshold.

Calculation of N'_k: When image l is offloaded to the edge, the UAV expects feedback within a specified delay time T_W. This feedback includes the image's criticality status and the reception acknowledgment. The total latency $t^e_{k,l}$ of image l of period k is the sum of its transmission, queuing, and execution time at the GCS, i.e.,

$$t^e_{k,l} = \underbrace{\frac{L}{R_{k,l}}}_{\text{transmission time}} + \underbrace{\tilde{t}^e_{k,l}}_{\text{queuing and processing time}} \tag{3}$$

where $\tilde{t}^e_{k,l}$ is the time the image spends at the edge (also called system delay), which includes both processing and queuing delay. Image l offloaded by UAV j is then dropped (hence removed from the queue) if $t^e_{k,l}{>}T_W$. It follows that the threshold system delay of image l is $T_{W,l}{=}T_W{-}L/R_{k,l}$. Thus, $N'_k{=}1$ if $t^e_{k,l}{>}T_{W,l}$, and 0 otherwise.

We also count the number of images from other UAVs that timeout before updating the queue. Let N''_k represent the count of tasks from other UAVs in the queue that timeout during the $1/I_k$ inter-arrival time in the period k. Additionally, the edge server experiences a load from all UAVs $i{\neq}j$, with a mean task arrival rate λ and service rate μ^e. Let $T_{W,\lambda}$ be the mean queuing and processing delay threshold of the tasks from other UAVs arriving at rate λ. Then, the average sum of queuing and processing time of a task at the edge is given by: $t^e_\lambda = \frac{\lambda}{2\mu^e(\mu^e-\lambda)} + \frac{1}{\mu^e}$.

Following the PASTA (Poisson Arrival Sees Time Average) rule, if the average queuing and processing time t^e_λ for an image exceeds the threshold $T_{W,\lambda}$, then the queuing and processing time for other images will also exceed this threshold. This results in all images violating the latency requirement. Therefore, we select the value of λ such that $t^e_\lambda{\leq}T_{W,\lambda}$, i.e., the total queuing and processing delay does not cross the threshold and none of them will time out, implying $N''_k{=}0\,, \forall k$.

E. Task queuing, processing time and energy consumption at the UAV

When the tasks are to be processed locally, the arrival rate of image processing tasks at the UAV in period k is equal to the images captured in that period at the I_k IPS rate; hence, the inter-arrival time is uniform. Furthermore, for fixed

TABLE I
UAV PROPULSION POWER PARAMETERS AND THEIR VALUES [6]

Notation	Description	Value
P_0	Blade profile power in hovering status	17.6 W
P_i	Induced power in hovering status	43.8 W
U_{tip}	Tip speed of the rotor blade	80 m/s
v_0	Mean rotor induced speed	3.19 m/s
d'	Fuselage drag ratio	0.81
ι	Air density	1.23 kg/m^3
s'	Rotor solidity	0.037 m^3
A'	Rotor disc area	0.5 m^2

service time t^u, the service rate of the processor at the UAV is $\mu^u{=}1/t^u$ tasks/s. The UAV maintains a queue where images are immediately placed upon capture, thus employing FIFO processing. Let $Q^u_{k,l}$ denote the number of tasks in the UAV queue when image l of period k is captured. Then the sum of waiting and processing time of image l captured at UAV in period k is $\tilde{t}^u_{k,l}{=}(Q^u_{k,l}{+}1)/\mu^u$. The queue update rule is:

$$Q^u_{k,l+1} = \max(0, Q^u_{k,l} - \mu^u/I_k + 1 - N'_k) \tag{4}$$

where N'_k is the number of dropped tasks whose waiting time exceeds a given threshold. Since no image transmission is required, the image latency at a UAV is $t^u_{k,l}{=}\tilde{t}^u_{k,l}$. Finally, as in the above analysis, $N'_k{=}1$ if $t^u_{k,l}{>}T_W$, and 0 otherwise.

With respect to energy consumption at the UAV, we break down the total UAV energy consumption into two components: UAV propulsion energy and energy consumption from processing or offloading images.

For a rotary-wing UAV moving at speed v_k, the propulsion energy consumption during period k can be calculated as [6]:

$$P(v_k) \approx \underbrace{P_0\left(1+\frac{3v_k^2}{U_{tip}^2}\right)}_{\text{blade profile}} + \underbrace{\frac{P_i v_0}{v_k}}_{\text{induced}} + \underbrace{\frac{1}{2}d'\iota s' A' v_K^3}_{\text{parasite}}. \tag{5}$$

Here, the first and third components are blade profile power and parasite power, respectively. The second part is the induced power to overcome induced drag, which decreases with v_k. The parameters and their values are detailed in Table I. Then, the energy consumed in the period k is given by:

$$E_{k,1} = P(v_k)\Delta t. \tag{6}$$

As for the energy consumption from processing or offloading images, we recall that P_t denotes the UAV transmission power; also let P_u be the UAV power needed for processing image locally onboard and $R_{k,l}$ be the transmission rate to offload image l, of size L captured in period k, from the UAV to the edge. Hence, the UAV's energy consumption for processing or offloading during period k is:

$$E_{k,2} = \begin{cases} P_t \sum_{l=1}^{N_k} \frac{L}{R_{k,l}}; & \text{offloading} \\ P_u \min(\frac{Q^u_k + N_k}{\mu^u}, \Delta t); & \text{local processing.} \end{cases} \tag{7}$$

In general, the energy for data processing is much larger than for data acquisition; thus, we neglect the latter for simplicity. Also, energy consumption at the UAV from data offloading is significantly less compared to the energy spent in processing images locally at the UAV.

IV. PROBLEM FORMULATION

We now define the optimization problem that each UAV must solve for intelligent execution of CV tasks. The goal is to minimize the UAV's propulsion and processing energy consumption while maximizing the distance covered during its flight. Additionally, the UAV must meet the following key requirements: (i) achieving the necessary detection accuracy based on the criticality of the images to be captured and (ii) ensuring that the latency constraints for each CV task are satisfied, all within the UAV's total energy budget. The optimization process accounts for the stochastic nature of channel conditions and image semantics. For each decision period k, we consider the following decision variables:

- the binary task offloading variable d_k, where $d_k{=}1$ if images are offloaded to the edge and $d_k{=}0$ if processed at the UAV;
- the discrete UAV speed variable, $v_k{\in}\mathcal{V}$;
- the discrete IPS variable, $I_k{\in}\mathcal{I}$;
- the uploading rate vector $\boldsymbol{R}_k{=}\{R_1,\ldots,R_{N_k}\}$ where N_k is the number of images in ensemble k.

Intelligent execution of CV tasks

$$\max_{\{I_k, v_k, d_k, \mathbf{R}_k\}_k} \sum_{k=1}^{K} \frac{\beta_1 E_{k,1}}{\Delta t P(v_{opt})} + \frac{\beta_2 E_{k,2}}{\bar{E}_2} + \frac{\beta_3 \rho_k}{\Delta t v_{\max}} \tag{8a}$$

s.t.

$$A_k \geq A_{\mathrm{T}}(\max\{\chi_{k,l}\}),\ \forall k, \tag{8b}$$

$$(1{-}d_k)t^u_{k,l} + d_k t^e_{k,l} \leq T_W,\ \forall k, l = \{1,\ldots,N_k\}, \tag{8c}$$

$$\sum_{k=1}^{K}(E_{k,1} + E_{k,2}) \leq E_{\mathrm{M}} \tag{8d}$$

The overall optimization problem is formulated as shown in the colored box above. The objective (8a) has three terms corresponding to: (i) UAV propulsion energy $E_{k,1}$, given by (6), that is influenced by the UAV speed, v_k in period k; (ii) UAV processing energy $E_{k,2}$, given by (7), that accounts for the energy used in either local processing or offloading of images in period k and is influenced by all decision variables I_k, v_k, d_k, and $\mathbf{R}_k$; (iii) distance covered by the UAV ρ_k, which is affected by the UAV's speed v_k in period k.

These terms capture the key variables controlling energy consumption in both propulsion and data processing, as well as the UAV's movement and IPS rate across each period. Here, $\beta_1, \beta_2{\in}\{-1, 0\}$, and $\beta_3{\in}\{0, 1\}$ are scalar weights assigned to each of the three terms, determining their relative importance in the optimization. Furthermore, each term of the objective function is normalized. The UAV propulsion energy consumption is normalized by the minimum UAV propulsion energy consumption, represented by $\Delta t P(v_{\text{opt}})$, where $P(v_{\text{opt}})$ is the UAV propulsion power at the optimal speed v_{opt}. Consequently, the term $E_{k,1}/(\Delta t P(v_{opt})){\in}[1, 2]$ for $2{\leq}v_k{\leq}20$ m/s. Similarly, the UAV processing energy consumption is normalized by the mean UAV processing energy $\bar{E}_2$, which is the average of the UAV transmission energy for offloading the ensemble and the UAV processing energy for processing the maximum allowed ensemble size in a period of duration time Δt. As a result, $E_{k,2}/\bar{E}_2{\in}[0, 1]$. Finally, the distance

covered by the UAV is normalized by the maximum distance it can cover when moving at maximum speed $v_{\max}$; hence, $\rho_k/(\Delta t v_{\max}) \in [0,1]$.

For each period, the detection accuracy $A_k = f(N_k, v_k)$ of the entire image ensemble cannot drop below the required threshold for the image with the highest criticality level in the ensemble. This ensures that the most critical image in the group is detected with sufficient accuracy, given the UAV's speed v_k and the number of images N_k in the ensemble. This is ensured by constraint (8b), where the required accuracy threshold $A_{\mathrm{T}}(\max\{\chi_{k,l}\})$ reflects the semantic criticality of the most important image in the ensemble, ensuring that scenes with higher importance are detected with greater accuracy. Constraint (8c) ensures that the total latency at UAV or at the edge does not exceed the specified threshold $T_W \; \forall k, l$. Constraint (8d) restricts the total energy consumed by the UAV over K periods from surpassing the available amount at the UAV for the mission.

To solve (8a), global knowledge of the future channel conditions and image semantics over the entire time horizon are required, which is not practical. We thus decompose the problem into subproblems, each corresponding to a duration Δt. Accordingly, we transform the cumulative energy constraint (8d) into an instantaneous power constraint, i.e., (8d) is updated to $P_k \leq P_{\mathrm{M}} \forall k$, where $P_{\mathrm{M}} = E_{\mathrm{M}}/K\Delta t$. Importantly, decomposing the original problem into subproblems overcomes the impracticality of using an energy constraint over the entire time horizon, which would require future information on parameters like criticality and channel SNR. Nevertheless, each subproblem still requires information on future channel and image semantics for that period. Moreover, as proved below, each subproblem remains NP-hard.

Theorem 1. *Each of the above subproblems is NP-hard.*

Proof: We show that the Vehicle Routing Problem (VRP) [23], which is NP-hard, can be polynomially reduced to an instance of the above subproblems. VRP determines the routes for a fleet of vehicles to deliver goods that minimize total distance or cost, subject to various constraints. We map the VRP to our subproblem as follows: each route in VRP corresponds to decisions on UAV speed v_k, task offloading d_k, IPS rate I_k, and uploading rates R_k in our problem. Constraints on route length and customer demands in VRP are analogous to the accuracy (8b) and latency (8c) constraints in our subproblem. Minimizing the total distance in VRP is equivalent to optimizing the weighted sum of energy consumption and maximizing distance covered in our problem. ■

In light of the problem complexity and its stochastic elements, we develop below a heuristic solution, named Intelligent UAV Network (IntUNe).

V. The IntUNe Framework

This section introduces the IntUNe solution to the above NP-hard subproblems. We give an overview of IntUNe and detail the formulation of the Constrained Markov Decision Process (CMDP) for the subproblem obtained for a given period k in (8a)–(8d). Next, we present the CSAC algorithm for IntUNe, and the solution to predict the transmission rate.

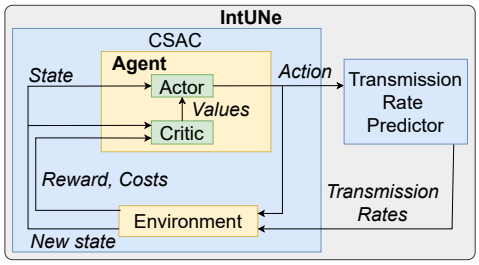

Fig. 6. IntUNe block diagram.

A. Overview

The discrete and binary nature of decision variables (namely, UAV speed v_k, IPS rate I_k, offloading decision d_k, and transmission rate vector $\mathbf{R}_k$ in (8a)), along with the stochastic characteristics of SNR and image criticality and the constraints of the inference accuracy and latency, complicates the optimization of the subproblems. Discrete variables introduce combinatorial challenges, while stochastic variables add uncertainty. We thus adopt an RL algorithm, as this approach is well suited for problems with non-linear relationships and discrete decision variables. Moreover, RL agents learn from experience, enabling them to discover effective solutions without requiring explicit knowledge of the system dynamics.

Further, in our optimization problem, the decision variable $\mathbf{R}_k$ depends on the offloading decision d_k and IPS rate I_k. If the decision is to offload, $\mathbf{R}_k$ is non-zero, and its size depends on the ensemble size for that period. Thus, the joint determination of all decision variables is practically impossible. Consequently, in each period, we first use the RL framework to find the values of I_k, v_k, and d_k. Subsequently, based on the action taken by the agent, we optimize the transmission rates for all the images in the k-th ensemble.

Fig. 6 illustrates the IntUNe block diagram, comprising two main blocks: CSAC and the transmission rate predictor. For a given state in a period, the CSAC agent optimizes the decision variables (actions) I_k, v_k, and d_k to maximize the expected sum of rewards and minimize costs. The decision variable d_k is then fed into the transmission rate predictor block, which outputs the transmission rate vector $\mathbf{R}_k$ for period k. All the decision variables I_k, v_k, d_k, and $\mathbf{R}_k$ are finally given as input to the environment, which executes them and outputs the next state, reward, and costs for the CSAC agent.

In relation to the CSAC equivalent of the original problem in (8a), employing an energy constraint over the entire time horizon in an RL problem, this introduces dynamics that evolve with cumulative energy consumption, thus affecting the agent's objectives and constraints. Initially, the agent experiences greater action flexibility as energy constraints are less restrictive. As the level of available energy diminishes, constraints become tighter, potentially impeding specific actions and objectives. This dynamic nature challenges RL algorithms, as optimal policies fluctuate with variations in the environment's state and remaining energy. In contrast, utilizing a power constraint, as in each subproblem, ensures consistency throughout UAV surveillance episodes, maintaining problem coherence. Furthermore, implementing a power constraint in

each period provides consistent and immediate feedback, augmenting the real-time policy adaptation of the agent. Moreover, the subproblem strategy enables more frequent updates and adaptations in RL policy in response to evolving environmental conditions. By optimizing variables over shorter time intervals, the agent can swiftly respond to dynamic changes in the system, fostering more agile and responsive decision-making.

B. CMDP formulation

A CMDP can be represented by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{C}, \kappa)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}$:$\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ and $\mathcal{C}$:$\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ denote the reward function and cost function, respectively, $\mathcal{P}$ is the transition probability function with $p(\mathbf{s}_k|\mathbf{s}_{k+1}, \mathbf{a}_k)$ denoting the transition probability from state $\mathbf{s}_k$ to state $\mathbf{s}_{k+1}$ given action $\mathbf{a}_k$, and κ is the discount factor in range $[0, 1]$. More specifically,

- The state vector is $\mathbf{s}_k$=$[N_{k-1}, h_\chi, \bar{\gamma}_{k-1}]$=$[\mathbf{x}_k, \mathbf{y}_k]$. $\mathbf{x}_k$=N_{k-1} constitutes the controllable part of the state variable where N_{k-1} is the ensemble size during period $(k-1)$. $\mathbf{y}_k$=$[h_\chi, \bar{\gamma}_{k-1}]$ constitutes the uncontrollable part of state variable vector where h_χ is the predicted highest criticality among ensemble that will be captured for the period k and $\bar{\gamma}_{k-1}$ is average of SNR in period $(k-1)$. The value of h_χ is predicted using an Autoregressive model (AR) of order $\mathcal{O}$ [24]. Since N_k and h_χ are discrete while $\bar{\gamma}_k$ is continuous, the $\mathcal{S}$ is hybrid in nature;
- The action vector is $\mathbf{a}_k$=$[I_k, v_k, d_k]$$\in\mathcal{A}$. Here, $I_k\in\mathbb{Z}^+$ is the IPS at which the camera captures the images during period k, $v_k\in\mathbb{Z}^+$ is the speed of UAV during period k, and d_k is the binary offloading decision variable that takes value d_k=0 when the decision is to process all N_k images locally, and d_k=1 when the decision is to offload all of them. Here, I_k, v_k and d_k are discrete in nature, therefore $\mathcal{A}$ is a multi-dimensional discrete action space;
- The reward function corresponding to (8a) is

$$r_k(\mathbf{s}_k, \mathbf{a}_k)=\frac{\beta_1 E_{k,1}}{\Delta t P(v_{opt})} + \frac{\beta_2 E_{k,2}}{\bar{E}_2} + \frac{\beta_3 \rho_k}{\Delta t v_{\max}};$$

- The cost functions for the constraints (8b)–(8d) are denoted by $c_k^{(1)}(\mathbf{s}_k, \mathbf{a}_k)$, $c_k^{(2)}(\mathbf{s}_k, \mathbf{a}_k)$, and $c_k^{(3)}(\mathbf{s}_k, \mathbf{a}_k)$ (resp.). The costs are 1 if the constraints are satisfied, and 0 otherwise:

$$\begin{aligned} c_k^{(1)}(\mathbf{s}_k, \mathbf{a}_k) &= \mathbb{1}\left(A_k < A_{\mathrm{T}}(\max\{\chi_{k,l}\})\right) \\ c_k^{(2)}(\mathbf{s}_k, \mathbf{a}_k) &= \mathbb{1}\left((1-d_k)t_{k,l}^u + d_k t_{k,l}^e > T_W\right) \\ c_k^{(3)}(\mathbf{s}_k, \mathbf{a}_k) &= \mathbb{1}\left(P_k > P_{\max}\right). \end{aligned} \tag{9}$$

The UAV acts as an RL agent, with a stochastic policy $\pi(\mathbf{s}_k|\mathbf{a}_k)$:$\mathcal{S}\to\mathcal{A}$. The agent interacts with the environment in each period, i.e., the UAV observes the state $\mathbf{s}_k$ and takes action $\mathbf{a}_k$. Then the environment moves to the next state s_{k+1} and provides the UAV with the reward $r_k(\mathbf{s}_k, \mathbf{a}_k)$ and costs $c_k^{(i)}(\mathbf{s}_k, \mathbf{a}_k)\,\forall i$. The reward and cost are known by the end of each period, not later than time T_W. The tuple $(\mathbf{s}_k, \mathbf{a}_k, r_k, c_k^{(i)}{}_{i=1,2,3}, \mathbf{s}_{k+1})$ is stored in the experience replay buffer $\mathcal{B}$. The UAV samples from $\mathcal{B}$ periodically to learn the optimal policy that maximizes the discounted reward while satisfying the constraints. Upon reaching a terminal state, the agent starts a new episode beginning in an arbitrary state in a new environment instance.

C. CSAC formulation

SAC is an off-policy RL algorithm based on the actor-critic framework, integrating entropy into the reward function to enhance exploration. CSAC extends SAC by using multiple critics and adaptive weights to balance rewards and constraints [25]. It optimizes objectives while minimizing constraint violations, which are managed effectively via the Lagrange-multiplier method. The key feature of SAC is entropy regularization, where the policy maximizes long-term rewards and entropy to ensure sufficient exploration. This approach helps prevent the policy from settling into suboptimal local optima. The original CMDP problem can be generalized into a more robust framework by incorporating entropy as [25]:

$$\begin{aligned} \max_{\pi} \quad & \sum_{k=1}^{K} \mathbb{E}_\pi \left[\kappa^k r_k(\mathbf{s}_k, \mathbf{a}_k) + \alpha \mathcal{H}(\pi(.|\mathbf{s}_k))\right] \\ \text{s.t.} \quad & c_k^{(i)}(\mathbf{s}_k, \mathbf{a}_k) \le b,\ i \in \{1,2,3\}, \forall k \end{aligned} \tag{10}$$

where $\mathcal{H}(\pi(.|\mathbf{s}_k))$=$-\log\pi(\mathbf{a}_k|\mathbf{s}_k)$ is the entropy of the current policy and b=0. The constrained problem (10) is converted to an unconstrained problem as:

$$\max_{\boldsymbol{\xi},\alpha} \mathbb{E}_\pi \left[\sum_k \left(\kappa^k r(\mathbf{s}_k, \mathbf{a}_k) + \alpha \mathcal{H}(\pi(.|\mathbf{s}_k))\right) - \sum_i \xi_i (c_k^{(i)} - b)\right]. \tag{11}$$

The temperature parameter $\alpha\ge$0 determines the relative importance of the entropy term against the reward, thus controlling the stochasticity of the optimal policy. $\boldsymbol{\xi}$=$\{\xi_i\}_{i=1,2,3}\ge$0 are Lagrangian penalty coefficients that can dynamically adjust the relative importance of each cost versus reward.

The critical periods for deriving the optimal policy π^* in CSAC apply to both continuous and discrete actions. However, in a discrete action space, $\pi(.|\mathbf{s}_k)$ is a mass function, not a density function. The policy directly outputs the action distribution as a probability vector for each action [26]. Thus, π maps states to a probability vector with $|\mathcal{A}|$ elements. Since the action set is discrete, the agent can fully recover the action distribution without needing a Monte Carlo estimate to determine loss functions, allowing for direct calculation of expectations over actions. Therefore, only the expectation over states remains as an estimate. Moreover, in CSAC for local constraints, four separate critics (Q functions) are trained, where one is a reward critic Q_ϕ for the reward and the entropy, and the rest are constraint critics $C^{(1)}, C^{(2)}, C^{(3)}$, one for each constraint term. Subsequently, similar to the formulation used in [26], critic parameters ϕ are trained to minimize the loss:

$$J_Q(\phi) = \mathop{\mathbb{E}}_{\mathbf{s}_k\sim\mathcal{B}} \left[\frac{1}{2}\left(Q_\phi(\mathbf{s}_k) - y_k\right)^2\right], \tag{12}$$

where y_k=$r(\mathbf{s}_k, \mathbf{a}_k)$$-\sum_i \xi_i C^{(i)}(\mathbf{s}_k, \mathbf{a}_k)$$+\kappa\pi(\mathbf{s}_k)^K(Q_{\bar{\phi}}(\mathbf{s}_{k+1})$ $-\alpha\log(\pi(\mathbf{s}_{k+1})))$ is calculated according to the cost-modified Bellman equation, and $\bar{\phi}\leftarrow\nu\bar{\phi}+(1-\nu)\phi$ with $\nu\ll$1.

Also, we get the actor loss function for discrete actions parameterized by θ as [26]–[28]:

$$J_Q(\theta) = \mathop{\mathbb{E}}_{\mathbf{s}_k\sim\mathcal{B}} \left[\pi(\mathbf{s}_k)^K \left[\sum_i \xi_i C^{(i)} - Q_\phi(\mathbf{s}_k) + \alpha\log\pi(\mathbf{s}_k)\right]\right]. \tag{13}$$

Algorithm 1 CSAC with discrete actions

Input: $\kappa, \alpha, \phi_1, \phi_2, \theta, \eta_{\pi_i}, \eta_\alpha, \eta_{\xi_i}, \tau, \mathbf{s}_k$

Initialize:
Replay buffer: $\mathcal{B} \leftarrow \{\}$
Local networks: $Q_{\phi_1} : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$,
$Q_{\phi_2} : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$, $Q_\theta : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$
Target networks: $\tilde{Q}_{\phi_1} : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$,
$\tilde{Q}_{\phi_2} : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$
Constraint networks: $C^{(1)} : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$,
$C^{(2)} : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$, $C^{(3)} : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$
Equalize target and local network weights: $\tilde{Q}_{\phi_1} \leftarrow Q_{\phi_1}$, $\tilde{Q}_{\phi_2} \leftarrow Q_{\phi_2}$
for each episode **do**
  for each environment period **do**
    $\mathbf{a}_k \sim \pi(\mathbf{a}_k|\mathbf{s}_k)$
    $\mathbf{s}_{k+1} \sim p(\mathbf{s}_{k+1}|\mathbf{s}_k, \mathbf{a}_k)$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{s}_k, \mathbf{a}_k, r_k, \{c_k^{(i)}\}_{i=1,2,3}, \mathbf{s}_{k+1})\}$
  end for
  for each gradient period **do**
    $\pi \leftarrow \pi - \eta_\pi \nabla_\pi J_\theta(\theta)$
    $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha J_\alpha(\alpha)$
    $\xi_i \leftarrow \xi_i - \eta_{\xi_i} \nabla_{\xi_i} J_{\xi_i}(\xi_i)$ for $i = \{1, 2, 3\}$
    $\tilde{Q}_{\phi_i} \leftarrow \tau Q_{\phi_i} + (1-\tau)\tilde{Q}_{\phi_i}$ for $i = \{1, 2\}$
  end for
end for

Output: $\tilde{Q}_{\phi_1}, \tilde{Q}_{\phi_2}, \pi$

Each Lagrange multiplier ξ_i is learned by minimizing its loss function: $J_{\xi_i}(\xi_i) = \mathop{\mathbb{E}}_{\mathbf{s}_k \sim \mathcal{B}} \left[\pi(\mathbf{s}_k)^K (\xi_i (b - C^{(i)}(\mathbf{s}_k, \mathbf{a}_k))\right]$, and the temperature loss parameter α is learned by minimizing loss function [26]: $J_\alpha(\alpha) = \mathop{\mathbb{E}}_{\mathbf{s}_k \sim \mathcal{B}} \left[\pi(\mathbf{s}_k)^K (-\alpha \log \pi(\mathbf{s}_k))\right]$.

Our strategy incorporates two Q-functions to reduce positive bias during policy improvement, a technique derived from the work presented in [25]. Specifically, we parameterize two Q-functions with parameters ϕ_i and train them separately to optimize $J_Q(\phi_i)$. For the value gradient in (12) and the policy gradient in (13), we use the minimum of the two Q-functions. Importantly, this dual Q-function strategy greatly accelerates training. The complete procedure is detailed in Algorithm (1). The method alternates between gathering experience from the environment using the current policy and updating the function approximations by employing stochastic gradients from batches drawn from a replay buffer.

Complexity analysis. In Algorithm (1), the action space is discrete. Thus, the action selection takes $\mathcal{O}(1)$ time per period, and the agent computes the expectation over all actions directly, reducing the need for Monte Carlo estimates. For each step in the environment, the complexity for the update of each Q-table of policy function is $\mathcal{O}(K \times |\mathcal{A}| + T_{gradP})$, where $|\mathcal{A}|$ is the size of the discrete action space and T_{gradP} is the time taken for updating the policy. For each critic Q-value function, Q-value update requires evaluating $|\mathcal{A}|$ possible actions at each state, so the complexity is $\mathcal{O}(K \times |\mathcal{A}| + T_{gradQ})$ where T_{gradQ} is the time taken for the critic Q-value gradient update. The temperature parameter α is updated using gradient descent and the time complexity for this operation depends on the number of states and actions, thus it has $\mathcal{O}(K \times |\mathcal{A}|)$ complexity. Similarly, the computation complexity of each Lagrange multiplier ξ_i is $\mathcal{O}(K \times |\mathcal{A}|)$. Thus, the total computational complexity for the CSAC with discrete actions algorithm with three constraints and two Q-functions for value and policy gradients per episode is: $\mathcal{O}(8 \times K \times |\mathcal{A}| + 2 \times T_{gradP} + 4 \times T_{gradQ})$.

D. Transmission rate predictor

The CSAC agent optimizes the decision variables v_k, I_k, and d_k in each period. The decision d_k is input to the transmission rate predictor block, which outputs $\boldsymbol{R}_k$ if d_k=1 and **0** if d_k=0. The predictor block uses an AR model of order O to forecast the mean SNR value for each image in the N_k ensemble [24]. The communication module of the considered UAV continuously monitors the channel, providing the AR model with historical channel SNR data. Using the predicted mean SNR for each image, the image transmission rate is then calculated using the rate expression in Sec. III-B.

VI. Performance Evaluation

This section first presents the experimental findings used to empirically derive the function $f(N_k, v_k)$. We recall that such a function maps the ensemble size N_k (determined by the IPS rate) and the UAV speed v_k to the classification accuracy A_k achieved by the video object detection model on a real-world videos dataset (see the accuracy function detection block of Fig. 7). This relationship accounts for motion blur, image correlation, and temporal consistency in the image ensemble, and was obtained through extensive experimentation using a state-of-the-art video object detection framework (Temporal ROI Align with ResNet-50-DC5), as detailed in Sec. VI-A.

Then we describe the setup configuration used for both the small-scale and the large-scale scenarios that we consider in Sec. VI-B. The simulation operates in discrete time steps and follows the workflow illustrated in Fig. 7. At each step, the UAV selects an action (IPS rate, speed, offloading decision, and transmission rates) based on the current state. The simulator then updates the environment, computing accuracy, energy usage, latency, and queuing dynamics, and feeds the resulting state and reward back into replay buffer that is used by the IntUNe agent. This interaction continues until the predefined maximum number of steps per episode is reached.

After training, the IntUNe policy is evaluated by running multiple episodes and recording classification accuracy, energy consumption, latency constraint violations, and distance covered. These metrics are then compared to optimal and baseline benchmark policies, i.e., exhaustive search for small-scale scenarios and state-of-the-art offloading schemes for larger-scale setups (see Secs. VI-C and Sec. VI-D, resp.).

A. Experimental results

State-of-the-art CV methods are implemented through deep neural networks (DNNs), which have been proven to be highly effective in modeling complex non-linear relationships, such as that between the pixels array of an image and the class labels and the coordinates of the objects represented therein.

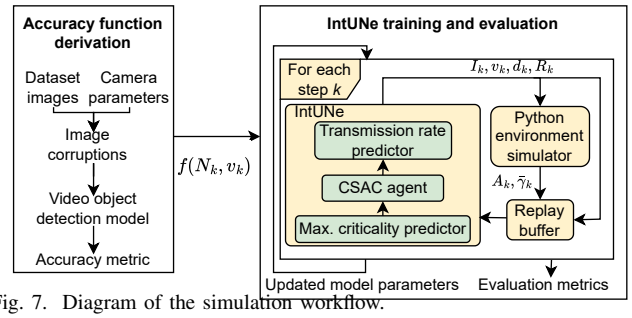

Fig. 7. Diagram of the simulation workflow.

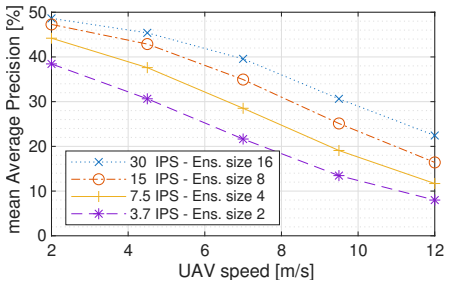

Fig. 8. Video object detection accuracy as UAV speed, IPS, and ensemble size vary. IPS and ensemble size are changed concurrently, so that Δt=0.53 remains constant.

Because of this complex relationship, formulating in closed form a function that maps the image capture conditions (i.e., the UAV speed and IPS settings) into attainable accuracy is not feasible. We thus conducted dedicated experiments to derive an accurate and realistic mapping for our performance evaluation.

To design the experiments, UAV images are captured sequentially by the same onboard camera sensor, with static orientation, which thus frames the same area. It is therefore essential to account for the temporal correlation between subsequent images, which is affected by the UAV speed. Specifically, the higher the UAV speed, the lower the correlation, as the UAV camera would pan the environment more quickly, making the scene more dynamic. Moreover, high speed can introduce artifacts such as motion blur, which lower the image quality and degrade the prediction accuracy. Similarly, a high IPS increases the temporal correlation, as it allows for more samples (i.e., images) for the same scene. Also, it decreases the "age of information" [29], as it provides more frequent updates on the object classes and positions in the scene. Finally, it is critical to account for the ensemble size, which varies the number of images that, because of their temporal correlation, can be leveraged to make a prediction in the current period. Remarkably, leveraging a higher number of images can increase the CV task accuracy, as it exploits information from previous images, especially when such information is no longer available (e.g., in case of partial occlusion).

To exploit the temporal correlation of subsequent images in an ensemble, we use video object detection. Contrary to object detection, which operates on independent images, this method identifies object classes and positions leveraging its own predictions on past images to improve accuracy. As video object detection model, we use the state-of-the-art Temporal ROI Align [30] with the ResNet-50-DC5 backbone. The model has been trained on the ImageNet VID dataset [31], that we also utilize as the test dataset. To account for the UAV speed and both the camera-related artifacts and the variable panning speed, we added motion blur to the test dataset through the *imagecorruptions* python library, with an effect severity proportional to the UAV speed (Fig. 7). Further, to consider the higher panning speed and, hence, accelerate the scene, we applied image skipping. To account for the IPS setting and the age of information, the dataset has been modified to "hold" the same image for a number of sampling periods inversely proportional to the IPS. For instance, for an IPS of 1/4 of the maximum, an image is held for 4 consecutive sampling periods. This introduces an age of information error, since model predictions are frozen even though the scene changes. The ensemble size has been considered by changing the number of past images that can be exploited as a reference by the object detection model to make predictions. As metric, we consider the mean Average Precision (mAP).

The results, shown in Fig. 8, highlight that the accuracy decreases at higher UAV speeds because of the quality degradation of the motion blur and the higher scene dynamics obtained through dataset image skipping. Specifically, at 30 IPS and an ensemble of size 16, the mAP goes from 48.6% to 22.4% when increasing the UAV speed from 2 m/s to 12 m/s. Moreover, higher accuracy can be obtained for higher IPS values and ensemble sizes because more images, which are also more temporarily-correlated, are considered by the object detection model while performing inference. Sudden and temporary adverse conditions (e.g., object occlusion, glares, low lighting, background clutter) have lower impact on accuracy performance. Indeed, up to 180% higher accuracy can be obtained when processing at once 16 images instead of 2, when captured over the same period Δt. Next, we use such results to evaluate IntUNe considering a realistic relationship of the system parameters with object detection accuracy.

B. Parameter settings

We consider a small-scale and a large-scale scenario. In both cases, we set the UAV flying altitude at 50 m, along a fixed circular trajectory with 0.3 km radius and the GCS at the center of the circle. The UAV is equipped with a $4,280$ mAh lithium-ion battery of 100 Wh of energy, thus able to provide $P_{\max}$=200 W for 30 minutes [32]. The UAV camera captures raw images at 8K resolution and the image file size following from a 10:1 compression is 14 MB [32]. Also, the UAV and the GCS feature a planar antenna array with, respectively, N_t=8×8 and N_r=16×16. The UAV transmission power is P_t=1 W and the required CV processing power is P_u=5 W. The backhaul mmWave channel has a bandwidth of B=200 MHz at 28 GHz and is simulated as described in Sec. III-B. We also consider M=10 quantized transmission rate values. The CV processing service rate of the edge server and processor onboard UAV are, respectively, μ^e=40 tasks/s and μ^u=33 tasks/s [33]. The duration of each period in IntUNe is Δt=0.5 s. Additionally, we assume that after each mission, the UAV's images are validated through external resources (e.g., human review or comparison with secondary data). This feedback helps refine the CV detection model at the UAV and edge, gradually improving its accuracy for future missions.

TABLE II
SMALL-SCALE SCENARIO: INTUNE VS. THE OPTIMUM

Metric	IntUNe	Optimum
Average reward	-1.9052	-1.9038
Reward variance	0.010	0.004
Ave. norm. UAV proc. energy cons.	1.67	1.46
Ave. norm. UAV propulsion energy cons.	1.01	1.04
Ave. norm. distance covered by UAV	0.77	0.59
Perc. of accuracy constraint violations	1.99%	0%
Perc. of latency constraint violations	5.1%	0%

During each mission, the model is assumed to be trained sufficiently and thus identify image criticality levels reliably. Furthermore, using the historical data of image criticality, the AR at the UAV accurately predicts the highest criticality level, h_χ, within the ensemble. This prediction provides the input state to IntUNe, enabling it to take the necessary action.

The edge load from UAVs other than the j-th UAV is set to λ=1 tasks/s, which represents a scenario with approximately $N_{\text{UAV}} = 10$, where each UAV other than the j-th contributes an offloading rate of λ=0.11 tasks/s. This is consistent with a scenario where UAVs are selective in their offloading decisions (as facilitated by IntUNe), transmitting primarily high-criticality images and adapting to link availability, rather than offloading every captured frame. Also, the target maximum latency for all criticality levels is set to T_W=2 s. The other parameters for the IntUNe algorithm are set as follows: $\beta_1, \beta_2, \beta_3$ are equal to -1, -1, and 1, respectively, while the RL parameters are κ=0.9, α=3×10^{-6}, $|\mathcal{B}|$=100, and $\mathcal{O}$=5.

C. Small-scale scenario

Benchmark. We now compare IntUNe against the optimal solution obtained via exhaustive search under perfect predictions. Since exhaustive search returns the true globally optimal solution, it represents the best achievable performance against which we can compare the performance of our solution.

Numerical results. In this case, the action space includes $\mathcal{V}_k$=$\{4, 6, 8, 10\}$, $\mathcal{I}_k$=$\{4, 8, 16\}$ with M_c=3 having p_1=0.95, p_2=0.9, p_3=0.85 and 100 periods, while keeping other parameters the same. Also, we simulate the mmWave backhaul channel with p=0.96, q=0.4, and σ^2= $-$ 140 dBm. Based on the accuracy values provided in Sec. VI-A for specific v_k and I_k, we interpolate to determine the accuracy values for the remaining v_k and I_k pairs. We then set the following threshold accuracy values $A_T(\chi)$=$\{30\%, 34\%, 42\%\}$ for the three criticality levels. To mitigate stochasticity, we assume perfect knowledge of future SNR and image semantics when computing the optimum (thus assuming perfect prediction).

Table II compares the metrics achieved by IntUNe against the optimum. IntUNe's performance is calculated as averages over inference episodes after IntUNe has converged, where only the rewards from periods in which all constraints are satisfied are considered. One can observe that IntUNe closely matches the optimum: notably, IntUNe shows higher processing energy consumption for UAVs but performs better in terms of UAV propulsion energy consumption and distance covered.

The RL approach in IntUNe prioritizes increased UAV speed, leading to higher accuracy violations but extended coverage and saving in UAV propulsion energy. Hence, IntUNe, leveraging RL, adapts to the stochastic environment, achieving good energy efficiency and enhanced distance coverage, albeit with occasional constraint violations. Also, by design, SAC uses a stochastic policy, inherently introducing variability in the actions taken. This stochasticity is crucial for effective exploration but also contributes to the reward variance.

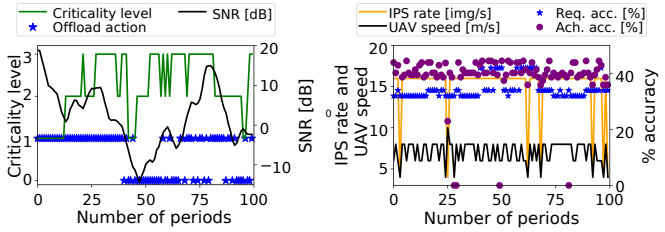

Fig. 9. Offloading decision (0/1) with IntUNe, mean SNR per period, and max ensemble criticality in 100 periods (left). Target accuracy, achieved accuracy, IPS rate, and UAV speed for 100 periods under IntUNe (right).

We further investigate the learning of IntUNe in each period of an evaluation episode in Fig. 9. As explained in Sec. I-A, images with high criticality have higher accuracy thresholds. Meeting the required accuracy for these high-criticality images requires a large ensemble size, necessitating higher IPS and lower UAV speed. However, high IPS can cause queuing delays; thus, balancing IPS and UAV speed for each ensemble is crucial. This correlation is illustrated in Fig. 9. Fig. 9(left), presenting the correlation between the offload action taken by the IntUNe RL agent and the SNR of the backhaul mmWave channel and the criticality level of the images in each period of an episode, confirms that images are offloaded when the channel SNR is favorable. For images of lower criticality, the RL sometimes offloads them to save processing energy, even at the cost of dropping some of the images on the poor channel link leading to reduced accuracy. Similarly, Fig. 9(right) demonstrates the correlation between the IPS rate and UAV speed to achieve the required accuracy levels. It underscores that the values of IPS and UAV speed are (resp.) increased and decreased when high accuracy is required, corresponding to high criticality images and large ensembles. Conversely, when lower accuracy is sufficient, the UAV speed is increased and IPS is lowered to save UAV propulsion and processing/transmission energy, and to increase the distance the UAV covers on its trajectory.

D. Large-scale scenario

Benchmarks. We compare IntUNe to enhanced adaptations of the approaches presented in [3] and [10], hereinafter referred to as *Benchmark1* and *Benchmark2*, respectively. We remark that, since some of the aspects we address and the problem we pose have not been previously tackled in the prior art, [3] and [10] are the closest to ours.

• *Benchmark1*: The approach in [3] optimizes task offloading decisions and the uploading rate to minimize a weighted sum of computation time and UAV energy consumption using a deep RL framework. Also, as we do in our study, it accounts for the edge server load and the UAV-edge link SNR. However, [3] does not consider delays caused by transmission, propagation of large packets, and queue waiting times. Also, it

TABLE III
P_{LOS} SIMULATION PARAMETERS [19]

Scenario	u_1	u_2	u_3	P_{LoS}	q	p
Sub-urban	0.1	750	8	0.956	0.4	0.97
Urban	0.3	500	15	0.392	0.8	0.69

does not optimize UAV speed to minimize propulsion energy consumption, and it considers a fixed IPS rate and single-image processing. Hence, for fair comparison, we modify the original scheme in [3] to minimize UAV processing energy while keeping delay and detection accuracy requirements as constraints, as in our problem, by fixing v_k and I_k and $N_k{=}1$ $\forall k$. The corresponding optimization problem is then:

$$\max_{\{d_k,R_k\}_{\forall k}} \quad \sum_{k=1}^{K'} \frac{E_{k,2}}{\bar{E}_2} \quad \text{s.t.} \quad (8b), (8c), (8d) \qquad (14)$$

where the $\Delta t{=}1/I_k\,\forall k$, i.e., one image per period and, thus, $K'{=}T_F I_k$. We then use as a benchmark for our proposed scheme the solution to the above problem, obtained by using the CSAC approach as mentioned in Sec. V-C, with the state being $\mathbf{s}_k{=}[h\chi, \bar{\gamma}_{k-1}]$ and action being $a_k{=}d_k$. The rate R_k is calculated using predicted SNR value, same as in IntUNe.

• *Benchmark2:* The approach in [10] jointly optimizes task offloading decisions and transmission rates to minimize processing delay and UAV energy consumption, using predicted channel SNR. However, it does not incorporate image semantics or UAV speed into the model. For a fair comparison, we adapt the original formulation in [10] by fixing the UAV speed v_k and IPS rate I_k for all periods, and by incorporating constraints on processing delay and energy consumption, consistent with (14). The resulting problem is solved using the Lyapunov drift-plus-penalty method, which decomposes the long-term stochastic optimization into a sequence of per-slot deterministic subproblems. Moreover, as in IntUNe, we use the AR regressor of order $I_k\Delta t$ to predict the channel SNR.

Numerical results. In the large-scale scenario, we configure the environment with five distinct criticality levels, i.e., $M_c{=}5$, with the value of p_i's for the 5-state MDP being $p_1{=}0.95$, $p_2{=}0.9$, $p_3{=}0.85$, $p_4{=}0.85$, and $p_5{=}0.7$. For each of these states, we define accuracy thresholds $A_T(\chi){=}\{30\%, 34\%, 36\%, 40\%, 42\%\}$. This threshold setting ensures that each criticality level is aligned with different accuracy expectations, reflecting the importance of accuracy at each level. To evaluate the robustness of IntUNe, we consider two scenarios: suburban and urban. The 2-state MDP values and other required parameters to simulate the mmWave UAV-GCS backhaul channel for each scenario, as detailed in Sec. III-B, are provided in Table III, with $\sigma^2{=}-140$ dBm [19].

The action space in this scenario consists of $\mathcal{I}_k{=}\{4, 8, 16, 32\}$ img/s, $\mathcal{V}_k{=}\{2, 4, 6, 8, 10, 12\}$ m/s. To estimate the accuracy for every possible combination of image rate I_k and speed v_k, we utilize specific accuracy values from Sec. VI-A. For any combinations not explicitly detailed, we apply interpolation to derive the expected accuracy, allowing for a complete mapping of accuracy outcomes across all values of v_k and I_k. Each training episode is set to run for a maximum of 300 periods, with the agent undergoing 3,000 episodes to ensure adequate learning over a broad range of conditions. Given the complexity of this large-scale scenario, achieving an optimal solution through exhaustive computation is impractical. Therefore, we assess the performance by comparing it to a benchmark method, focusing specifically on two UAV speeds, 2 m/s and 6 m/s, while keeping the image capture rate fixed at 4 img/s for consistency. These speeds are chosen to illustrate the energy-accuracy trade-offs in operation. At a slower speed of 2 m/s, the UAV consumes more propulsion energy, but it can generally meet the accuracy requirements across all criticality levels due to the more stable capture conditions. Conversely, at a higher speed of 6 m/s, the UAV conserves energy by spending less time in flight; however, it struggles to achieve the required inference accuracy, particularly for more critical states. Testing at even higher speeds while maintaining a capture rate of 4 img/s consistently results in failure to meet target accuracy, leading to near-zero rewards.

First, we show the convergence of IntUNe in large-scale scenarios. Fig. 10 illustrates IntUNe's convergence of the actor loss and critic loss (as in (13) and (12), resp.) in the two considered scenarios. The actor loss initially oscillates because of the entropy regularization term, which ensures that the policy maintains some level of exploration. The critic loss in SAC stabilizes around a small value rather than reaching 0 due to the fact that the target Q-values are estimates and may vary because of the stochastic nature of the policy and environment.

Next, Fig. 11(left) shows the correlation between offloading decisions, backhaul mmWave SNR, and image criticality. Offloading occurs predominantly when the channel SNR is favorable, demonstrating the agent's ability to exploit good radio link conditions. The trace also captures LoS-to-NLoS transitions, where SNR drops sharply due to mmWave blockage, prompting the agent to suppress offloading to avoid excessive delay or packet loss. When image criticality is high, the agent often processes locally rather than offloading, since increasing IPS and reducing UAV speed can meet accuracy requirements without risking latency violations. Occasional offloading under poor SNR happens only when both the local accuracy prediction is insufficient and the estimated transmission time remains within the latency constraint, or

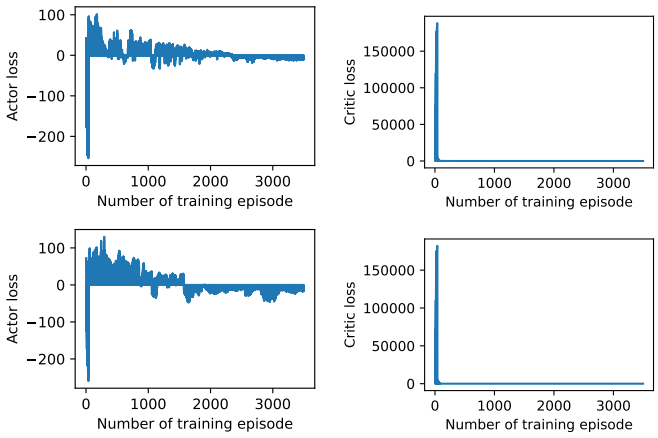

Fig. 10. Convergence of IntUNe's actor loss (left) and critic loss (right) during the training episodes, for the suburban (top) and urban (bottom) scenarios.

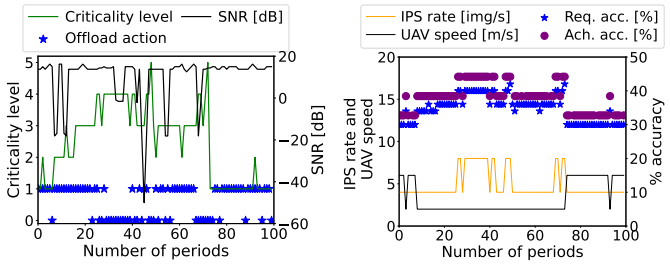

Fig. 11. Suburban scenario: (left) offloading decision (0/1) with IntUNe, mean SNR per period, and maximum ensemble criticality level (from 1 to 5) in 100 periods of an episode; (right) required accuracy, achieved accuracy, IPS rate, and UAV speed for 100 periods of an episode under IntUNe.

as a result of the SAC policy's inherent exploration during learning. Fig. 11(right) shows the corresponding IPS rate, UAV speed, and achieved versus required accuracy. During high-criticality periods, IntUNe lowers UAV speed and increases IPS to enlarge the ensemble size, ensuring the accuracy target is met even when LoS-to-NLoS transitions degrade the link.

Lastly, Table IV and Table V compare IntUNe performance in suburban and urban environments under two configurations: (i) default configuration where $d_k, v_k, I_k, \boldsymbol{R}_k$ are jointly optimized and (ii) fixed-speed where UAV speed is fixed at v_k=6 m/s $\forall k$ while the remaining variables are optimized. Fixing the UAV speed also fixes the distance covered and propulsion energy consumption, leaving only UAV processing energy in (8a) as the variable term, optimized via d_k, I_k, and $\boldsymbol{R}_k$. For fairness, we still include the normalized propulsion energy and normalized distance terms together with normalized mean UAV processing energy when calculating the average reward in Table IV and Table V. Values are calculated as averages over evaluation episodes, considering only the rewards in the periods where all constraints are satisfied. The results underline that IntUNe maximizes the reward compared to the benchmarks. With $\beta_1 = -1$ and $\beta_2 = -1$ to minimize energy consumption in propulsion and processing, and β_3=1 to maximize distance, the total reward from (8a) is maximized but takes a negative value due to the negative coefficients. We provide the discussion for the results in case of suburban and urban scenarios.

Suburban scenario. As shown in Table IV, the proposed IntUNe in the default configuration achieves a balanced trade-off between propulsion efficiency, processing cost, and constraint satisfaction. Adaptive speed control along the UAV trajectory reduces normalized propulsion energy to 1.272 and increases normalized distance covered to 0.35 compared to Benchmark1 at v=2 m/s (fixed at 1.562 and 0.167), while maintaining an average UAV speed of 4.20 m/s. The higher IPS rate (4.38 img/s) slightly increases normalized processing energy (0.184) relative to benchmarks but enables the lowest accuracy violation rate (0.72%) and lower latency violations (5.28%) than both benchmarks at v=2 m/s and v=6 m/s. Despite using deterministic constraints, the stochastic nature of state variables such as SNR and criticality inherently leads to a stochastic optimal policy in RL. The optimization process converges to a policy that performs well on average but occasionally violates constraints due to random fluctuations. Nonetheless, the percentage of accuracy violations with IntUNe is the lowest (see Table IV).

In the fixed-speed case (v = 6 m/s), IntUNe has constant propulsion energy (1.034) and distance covered (0.5), but UAV processing energy rises to 0.648 due to more frequent on-board processing at an elevated IPS rate (18.67 img/s) to meet the accuracy requirement at the fixed UAV speed. While accuracy violations remain low (1.55%), this configuration exhibits slightly higher latency violations (4.64%) than the default IntUNe due to higher IPS.

Similarly, in Benchmark1 and Benchmark2, variations are observed only in UAV processing energy, which fluctuates due to the stochastic nature of the SNR and IPS rate. Benchmark1 at v=2 m/s suffers from high accuracy violations (9.28%) despite lower processing energy, as the fixed IPS rate (4 img/s) under-provisions accuracy for high-criticality images. Benchmark1 at v=6 m/s achieves its best average reward (−0.91) due to reduced energy consumption but incurs very high accuracy violations (41.55%), making it unsuitable for missions with strict accuracy constraints.

Benchmark2 maintains a very high offloading ratio (0.94) at both v=2 m/s and v=6 m/s because its decision rule depends solely on predicted SNR and a fixed latency threshold; with identical SNR traces in both suburban cases, the offloading behavior remains unchanged. Moreover, it fails to optimize at sharp LoS–NLoS transitions at mmWave, causing frequent transmissions under suboptimal SNR. This leads to very high accuracy violations at v=2 m/s compared to the default IntUNe. At v=6 m/s, accuracy violations rise sharply to 44.19%, similar to Benchmark1 at the same speed. Latency violations for Benchmark2 remain high (11.45%) in both speed settings because aggressive offloading causes packet losses and delay spikes when the channel is degraded.

For completeness, we also include in Table IV the mean and root mean squared error (RMSE) of accuracy and latency achieved by IntUNe and the benchmarks. The RMSE refers to the error between achievable and required values across all periods in the episode. Both IntUNe variants surpass the mean target accuracy of 32.67% per episode (achieving 35.81% for default and 42.69% for fixed-speed). The default IntUNe also achieves the lowest RMSE in accuracy (3.89%) among all methods, showing consistent per-period performance under stochastic SNR and image criticality.

Urban scenario. As shown in Table III, we set P_{LoS}=0.392 to reflect the more obstructed urban environment. Despite these challenges, IntUNe in default mode outperforms benchmarks by achieving better normalized UAV propulsion energy (1.02) and distance covered (0.59) compared to fixed-speed benchmarks at 2 m/s and 6 m/s (both fixed at 0.167) (Table V). This is due to IntUNe average speed of 7.11 m/s, close to the optimal 8 m/s, enabling efficient energy use and faster coverage despite reduced LoS. To maintain accuracy at this higher speed, IntUNe optimizes its IPS rate, with a mean of 31.71 img/s, leading to higher mean UAV processing energy consumption than the benchmark. This trade-off is justified as the increased IPS rate enhances accuracy while achieving near-optimal speed. UAV processing energy depends on onboard image processing (when offloading is not selected) and image transmission (when offloading is chosen). Regarding latency

TABLE IV
PERFORMANCE COMPARISON OF INTUNE (DEFAULT AND FIXED-SPEED) WITH BENCHMARK1 [3] AND BENCHMARK2 [10] IN THE SUBURBAN SCENARIO

Metric	**IntUNe (Default)**	**IntUNe (v=6 **m/s**)**	**Benchmark1 (v=2 **m/s**)**	**Benchmark1 (v=6 **m/s**)**	**Benchmark2 (v=2 **m/s**)**	**Benchmark2 (v=6 **m/s**)**
Average reward	-1.105	-1.177	-1.516	-0.914	-1.475	**-0.614**
Reward variance	0.739	0.095	1.246	**0.000**	**0.000**	0.003
Mean normalized UAV propulsion energy	1.272	**1.034**	1.562	**1.034**	1.562	**1.034**
Mean normalized UAV processing energy	0.184	0.648	0.121	0.110	**0.080**	**0.080**
Mean normalized distance covered	0.350	**0.500**	0.167	**0.500**	0.167	**0.500**
Offloading ratio	0.830	0.286	0.880	0.890	**0.937**	**0.937**
Average UAV speed (m/s)	4.2	**6.0**	2.0	**6.0**	2.0	**6.0**
Average IPS rate	4.38	**18.67**	4.00	4.00	4.00	4.00
Accuracy constraint violations (%)	**0.72%**	1.55%	9.280%	41.550%	7.620%	44.190%
Latency constraint violations (%)	5.28%	**4.64%**	5.480%	5.560%	11.450%	11.450%
Mean achieved accuracy (%)	35.81%	42.69%	38.32%	32.24%	**44.19%**	39.23%
Mean achieved latency (s)	0.085	0.626	0.086	0.089	**0.005**	**0.005**
RMSE of achieved accuracy (%)	**3.89%**	11.72%	7.02%	5.31%	12.71%	8.09%
RMSE of achieved latency (s)	1.940	**1.580**	1.950	1.950	1.995	1.995

TABLE V
PERFORMANCE COMPARISON OF INTUNE (DEFAULT AND FIXED-SPEED) WITH BENCHMARK1 [3] AND BENCHMARK2 [10] IN THE URBAN SCENARIO

Metric	**IntUNe (Default)**	**IntUNe (v=6 **m/s**)**	**Benchmark1 (v=2 **m/s**)**	**Benchmark1 (v=6 **m/s**)**	**Benchmark2 (v=2 **m/s**)**	**Benchmark2 (v=6 **m/s**)**
Average reward	-1.070	-0.981	-2.350	-0.994	-1.713	**-0.852**
Reward variance	0.358	0.021	0.865	0.125	**0.001**	0.011
Mean normalized UAV propulsion energy	**1.020**	1.034	1.562	1.034	1.562	1.034
Mean normalized UAV processing energy	0.556	0.447	0.950	0.410	**0.318**	**0.318**
Mean normalized distance covered	**0.59**0	0.500	0.167	0.500	0.167	0.500
Offloading ratio	0.460	0.198	0.036	0.600	**0.601**	**0.601**
Average UAV speed (m/s)	**7.1**	6.00	2.0	6.0	2.0	6.0
Average IPS rate	**31.710**	14.740	4	4	4	4
Accuracy constraint violations (%)	**1.17%**	2.04%	8.25%	59.69%	30.69%	53.91%
Latency constraint violations (%)	26.60%	3.85%	**1.46%**	64.29%	38.51%	38.51%
Mean achieved accuracy (%)	40.96%	41.25%	38.62%	9.90%	**44.19%**	39.23%
Mean achieved latency (s)	0.350	0.522	**0.003**	0.012	0.007	0.007
RMSE of achieved accuracy (%)	10.47%	10.65%	**7.46%**	27.51%	12.71%	8.09%
RMSE of achieved latency (s)	1.670	**1.640**	1.990	1.990	1.993	1.993

constraint violations, IntUNe has a higher violation rate than Benchmark1 at 2 m/s due to a significantly higher offloading rate (0.46 vs. 0.036). However, IntUNe 's latency violations remain substantially lower than those of the benchmarks. It also achieves a mean accuracy of 40.96%, exceeding the required 32.67%, highlighting IntUNe 's adaptive processing suited to NLoS urban conditions. Although IntUNe 's RMSE in accuracy is higher than Benchmark1 at 2 m/s and Benchmark2 at 6 m/s, it is much lower than Benchmark1 at 6 m/s and close to Benchmark2 at 2 m/s, indicating consistent accuracy under challenging conditions. Also, while IntUNe 's mean latency is higher than the benchmarks, it stays within acceptable limits, and its latency RMSE is slightly better, confirming more stable latency despite urban challenges.

In the fixed-speed case (v=6 m/s), normalized propulsion energy (1.034) and distance covered (0.5) remain constant, while normalized processing energy drops to 0.447 due to a lower IPS rate (14.74 img/s) and reduced offloading ratio, which lowers latency violations to 3.85% but slightly increases accuracy violations to 2.04% compared to default IntUNe. This variant achieves the highest mean accuracy (41.25%), surpassing the target 32.67%, though its accuracy RMSE (10.65%) is marginally higher than default IntUNe (10.47%).

For Benchmark1 at v=2 m/s, the fixed IPS rate (4 img/s) limits accuracy, resulting in 8.25% accuracy violations despite low latency violations (1.46%) and low offloading (0.036). At v=6 m/s, accuracy violations increase sharply to 59.69%, and latency violations to 64.29% due to low IPS rate. Benchmark2 maintains a similar offloading ratio (0.6) at both speeds but incurs high accuracy violations (30.69% at 2 m/s, 53.91% at 6 m/s) and high latency violations (38.51%) in both cases because of aggressive offloading during channel fades.

Discussion on performance gains. The significant performance gap between IntUNe and the considered benchmarks [4], [11] is due to the multi-dimensional adaptability of our approach. Specifically, Benchmark1 focuses on offloading decisions under fixed camera and speed settings; consequently, it cannot compensate for high-speed motion blur, leading to the high accuracy violations observed in Table V. Benchmark2, while efficient in delay-aware offloading, lacks awareness of image semantics. It thus treats all frames equally, often offloading low-criticality data even when the mmWave link is deteriorating, which causes the observed spikes in latency.

IntUNe instead leverages the coupling between UAV speed and IPS to dynamically adjust the ensemble size. In case of high-criticality scenes, the UAV reduces speed to minimize blur and increases the IPS to provide more temporal context to the CV model, effectively trading off accuracy for temporary higher processing power. This trade-off is ignored by [4] and [11], which rely on static acquisition parameters.

Summary. IntUNe substantially reduces UAV processing energy through intelligent adaptation of IPS and offloading strategies compared to benchmarks. Specifically, in the suburban scenario, it achieves 18.61% savings in UAV propul-

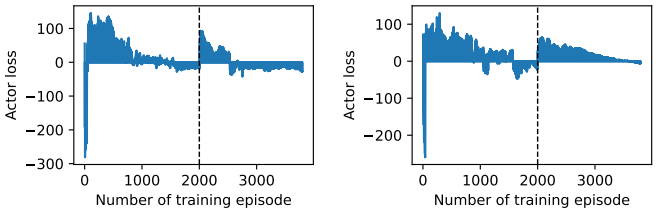

Fig. 12. Performance evolution of IntUNe during environmental transitions: from Suburban to Mixed scenario (left), and from Urban to Mixed scenario (right). The vertical dashed line at episode 2,000 indicates the moment of the switch.

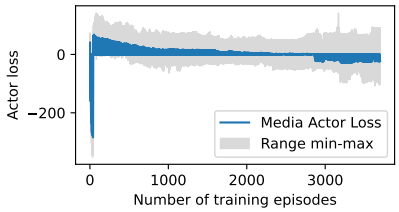

Fig. 13. Convergence of the IntUNe's actor loss across 128 randomized environment instances. The solid line represents the mean loss, while the shaded region indicates the range between the minimum (best-case) and maximum (worst-case) loss values observed.

sion energy and covers 109.58% more distance compared to both Benchmark1 and Benchmark2 operating at 2 m/s. While IntUNe incurs higher processing energy in this scenario, it reduces accuracy constraint violations by 92.24% and 90.54% compared to Benchmark1 and Benchmark2, respectively. In the urban scenario, IntUNe achieves even higher propulsion energy savings of 34.67% and extends the distance covered by 252.99% compared to both benchmarks at 2 m/s. Also, it reduces accuracy violations by 85.82% relative to Benchmark1 and by 96.19% relative to Benchmark2. Although IntUNe consumes more processing energy than Benchmark2 in this setting, the significant improvements in coverage and accuracy constraint adherence demonstrate its effectiveness. Further, to counteract the harsher urban environment's increased LoS–NLoS transitions, IntUNe significantly increases IPS rate from 4.38 img/s in suburban to 31.71 img/s in urban scenarios (Table IV and Table V respectively), a 524% increase, enabling it to sustain high accuracy. This higher IPS rate is aligned with recent real-world UAV deployments capable of comparable processing throughput [34]. Moreover, both Benchmark1 and Benchmark2 operating at 6 m/s exhibit accuracy violation rates exceeding 40% in suburban and 50% in urban scenarios, rendering these schemes impractical at higher speeds.

E. Adaptability and Robustness Analysis

To further validate the capabilities of IntUNe, we investigate its adaptability to sudden environmental changes and its robustness across a wide range of channel conditions. This analysis moves beyond the static Suburban and Urban scenarios discussed previously, demonstrating the agent's ability to generalize its learning to unseen environments.

Adaptability to dynamic environment switches. We first evaluate the ability of IntUNe to adapt its policy online when the operational environment changes abruptly during a mission. We define a "Mixed" scenario, representing an intermediate environment between the suburban and urban settings, characterized by the following mmWave channel parameters: $P_{LoS} = 0.674$, $q = 0.6$, and $p = 0.81$.

In this experiment, we initialize the training in a source environment (either suburban or urban). At episode $2,000$ of the training phase, we instantly switch the environment parameters to those of the Mixed scenario. Fig. 12 illustrates the evolution of the actor loss during this transition.

As shown in Fig. 12 (left), the agent initially operates in the favorable suburban environment. Upon switching to the Mixed scenario at episode $2,000$, which features a lower probability of LoS and higher blockage frequency, we observe a transient adaptation phase. Immediately after the switch, the loss spikes, as the actions previously learned violate the constraints of the new environment. After about 500 episodes since the switch, the agents' loss converges, showing that the agents have learned the new environment. Similarly, Fig. 12 (right) depicts the transition from the harsh urban environment to the relatively more stable Mixed scenario. In this case, the adaptation is characterized by a steady and slower (compared to Fig. 12 (left)) decrease in the actor loss as the agent exploits the improved channel statistics. Importantly, in both scenarios, IntUNe demonstrates its ability to adapt to sudden environmental changes during execution.

Generalization across randomized environments. To demonstrate the robustness of IntUNe and ensure that the convergence reported in Sec. VI-D is not limited to specific simulation instances, we conducted an extensive generalization analysis. We trained 128 distinct models, where each model operates in a unique environment with channel parameters drawn from a uniform distribution. Specifically, P_{LoS} and q were sampled from the intervals $[0.39, 0.95]$ and $[0.4, 0.8]$, respectively, covering the entire spectrum of environments, from dense urban to open suburban areas. Note that, because of the different values of P_{LoS}, this analysis may also reflect operation conditions where different values of UAV flight altitudes are considered.

Fig. 13 presents the convergence of the actor loss across these 128 instances. The plot highlights the mean convergence trend, as well as the envelope defined by the best-case (minimum loss) and worst-case (maximum loss) scenarios. Despite the significant variability in channel conditions, IntUNe consistently converges across all instances. The worst-case scenario corresponds to environments with low P_{LoS} and high q, where the agent requires longer exploration to optimize the entropy-regularized objective. Nevertheless, the narrow spread between the minimum and maximum values at the end of training confirms the reliability and generalization capability of IntUNe.

VII. Conclusion

We addressed the challenges of optimizing edge-assisted UAV surveillance systems by proposing a comprehensive approach that jointly optimizes UAV speed, IPS rate, offloading schemes, and transmission rates. Our method integrates image criticality and UAV-to-edge mmWave channel statistics to enhance performance by reducing UAV energy consumption and

increasing the distance. In light of the (proven) NP-hardness of the optimization problem, we designed the IntUNe low-complexity solution, which efficiently and effectively solves the problem we posed. Extensive numerical analysis shows that IntUNe significantly improves both energy efficiency and inference accuracy compared to state-of-the-art benchmarks. Also, by considering images criticality and leveraging contextual analysis of image ensembles, IntUNe ensures timely and accurate object detection, which is crucial for effective surveillance operations, outperforming its competitive alternatives. Even in worst case urban scenarios, with low probability of LoS, IntUNe achieves 34.67% increased saving in UAV propulsion energy and extends coverage by an extra 252.99%, while also reducing inference accuracy constraint violations by up to 96.19%. For future work, we aim to extend IntUNe to cope with multi-server scenarios, handover management, and worst-case interference conditions, including jamming, and assess its robustness to such scenarios.

REFERENCES

[1] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-based IoT platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, 2017.

[2] S. A. Huda and S. Moh, "Survey on computation offloading in UAV-Enabled mobile edge computing," *Journal of Network and Computer Applications*, vol. 201, p. 103341, 2022.

[3] D. Callegaro and M. Levorato, "Optimal edge computing for infrastructure-assisted UAV systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1782–1792, 2021.

[4] P. Sujit, S. Saripalli, and J. B. Sousa, "Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicles," *IEEE Control Systems Magazine*, vol. 34, no. 1, pp. 42–59, 2014.

[5] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 408–417.

[6] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[7] S. A. Huda and S. Moh, "Deep reinforcement learning-based computation offloading in UAV swarm-enabled edge computing for surveillance applications," *IEEE Access*, vol. 11, pp. 68 269–68 285, 2023.

[8] Z. Ran, Y. Chuanwei, J. Hai *et al.*, "Parallel key frame extraction for surveillance video service in a smart city," *Plos One*, vol. 10, no. 8, pp. 16–56, 2015.

[9] S. Hayat, R. Jung, H. Hellwagner, C. Bettstetter, D. Emini, and D. Schnieders, "Edge computing in 5G for drone navigation: What to offload?" *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2571–2578, 2021.

[10] Y. Wang, M. Kong, G. Zhang, W. Wang, T. Nakachi, and J. Liou, "Adaptive task offloading for mobile edge computing with forecast information," *IEEE Transactions on Vehicular Technology*, 2024.

[11] D. Cavaliere, S. Senatore, and V. Loia, "Proactive UAVs for cognitive contextual awareness," *IEEE Systems Journal*, vol. 13, no. 3, pp. 3568–3579, 2018.

[12] M. J. Er, C. Ma, T. Liu, and H. Gong, "Intelligent motion control of unmanned surface vehicles: A critical review," *Ocean Engineering*, vol. 280, p. 114562, 2023.

[13] F. Koohifar, A. Kumbhar, and I. Guvenc, "Receding horizon multi-UAV cooperative tracking of moving RF source," *IEEE Communications Letters*, vol. 21, no. 6, pp. 1433–1436, 2016.

[14] L. Zhang, R. Tan, M. Ai, H. Xiang, C. Peng, and Z. Zeng, "DSUTO: Differential rate SAC-based UAV-assisted task offloading algorithm in collaborative edge computing," in *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2023, pp. 2328–2335.

[15] L. Zhang, R. Tan, Y. Zhang, J. Peng, J. Liu, and K. Li, "UAV-assisted dependency-aware computation offloading in device–edge–cloud collaborative computing based on improved actor–critic drl," *Journal of Systems Architecture*, vol. 154, p. 103215, 2024.

[16] I. Ahmad, "Intelligent task offloading decisions for enhanced swarm-based UAV surveillance," *IEEE Transactions on Aerospace and Electronic Systems*, 2025.

[17] N. Varshney, C. Puligheddu, C. F. Chiasserini, C. Casetti, and S. De, "Intelligent Execution of Computer Vision Tasks in Delay-Constrained UAV-Aided Networks," in *2023 IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 86–91.

[18] W. van Iersel, M. Straatsma, E. Addink, and H. Middelkoop, "Monitoring height and greenness of non-woody floodplain vegetation with UAV time series," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 141, pp. 112–123, 2018.

[19] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, 2014.

[20] ITU-R, "Propagation data and prediction methods required for the design of terrestrial broadband radio access systems operating in a frequency range from 3 ghz to 60 ghz," International Telecommunication Union, Recommendation ITU-R P.1410-6, aug 2023, p Series: Radiowave propagation.

[21] N. Varshney and S. De, "Multi-RF beamforming-based cellular communication over wideband mmWaves," *IEEE Transactions on Communications*, vol. 70, no. 4, pp. 2772–2787, 2022.

[22] F. Sohrabi and W. Yu, "Hybrid analog and digital beamforming for mmWave OFDM large-scale antenna arrays," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 7, pp. 1432–1443, 2017.

[23] K. C. Tan, L. H. Lee, Q. Zhu, and K. Ou, "Heuristic methods for vehicle routing problem with time windows," *Artificial intelligence in Engineering*, vol. 15, no. 3, pp. 281–295, 2001.

[24] C. Wu, X. Yi, Y. Zhu, W. Wang, L. You, and X. Gao, "Channel prediction in high-mobility massive mimo: From spatio-temporal autoregression to deep learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 1915–1930, 2021.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[26] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint arXiv:1910.07207*, 2019.

[27] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. Spaan, "Safety-constrained reinforcement learning with a distributional safety critic," *Machine Learning*, vol. 112, no. 3, pp. 859–887, 2023.

[28] Y. Xu, Z. Zhao, P. Cheng, Z. Chen, M. Ding, B. Vucetic, and Y. Li, "Constrained reinforcement learning for resource allocation in network slicing," *IEEE Communications Letters*, vol. 25, no. 5, pp. 1554–1558, 2021.

[29] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[30] T. Gong, K. Chen, X. Wang, Q. Chu, F. Zhu, D. Lin, N. Yu, and H. Feng, "Temporal roi align for video object recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, 2021, pp. 1442–1450.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[32] DJI, "DJI Inspire 3 datasheet," https://www.dji.com/it/inspire-3/specs, 2023.

[33] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[34] R. N. Tripathi, K. Agarwal, V. Tripathi, R. Badola, and S. A. Hussain, "Conservation in action: Cost-effective UAVs and real-time detection of the globally threatened swamp deer (rucervus duvaucelii)," *Ecological Informatics*, vol. 85, p. 102913, 2025.