DroneCOCoNet: Learning-based Edge Computation Offloading and Control Networking for Drone Video Analytics

Chengyi Qu[a], Prasad Calyam[a], Jeromy Yu[b], Aditya Vandanapu[c], Osunkoya Opeoluwa[a], Ke Gao[a], Songjie Wang[a], Raymond Chastain[a] and Kannappan Palaniappan[a]

[a]*University of Missouri - Columbia, Missouri, 65201, USA.*
[b]*Purdue University, Indiana, 47907, USA.*
[c]*University of Illinois at Chicago, Illinois, 60607, USA.*

ARTICLE INFO

Keywords:
Edge/Fog Computation Offloading
Drone Video Analytics
Mobile Edge Computing
Learning-based Scheme
Data processing in Fog Computing

ABSTRACT

Multi-Unmanned Aerial Vehicle (UAV) systems with high-resolution cameras have been found useful for operations such as smart city and disaster management. These systems feature Flying Ad-Hoc Networks (FANETs) that connect the computation edge with UAVs and a Ground Control Station (GCS) through air-to-ground wireless network links. Leveraging the edge/fog computation resources effectively with energy-latency-awareness, and handling intermittent failures of FANETs are the major challenges in supporting video processing applications. In this paper, we propose a novel "DroneCOCoNet" framework for drone video analytics that coordinates intelligent processing of large video datasets using edge computation offloading and performs network protocol selection based on resource-awareness. We present two edge computation offloading approaches, i.e., heuristic-based and reinforcement learning-based approaches. These approaches provide intelligent task sharing and co-ordination for dynamic offloading decision-making among UAVs. Our scheme handles the problem of computation offloading tasks in two separate ways: (i) heuristic decision-making process, and (ii) Markov decision process; wherein we aim to minimize the total computation costs as well as latency in the edge/fog resources while minimizing video processing times to meet application requirements. Our experimental results show that our heuristic-based offloading decision-making scheme enables lower scheduling time and energy consumption for low drone-to-ground server ratios. In comparison, our dynamic reinforcement learning-based decision-making approach increases the accuracy and saves overall time periodically. Notably, these results also hold in various other multi-UAV scenarios involving largely different numbers of detected objects in e.g., smart farming, transportation traffic flow monitoring and disaster response.

1. Introduction

In the last few decades, Unmanned Aerial Vehicles (UAVs), also known as drones, have been extensively used in different scenarios in urban and rural area control such as disaster response, surveillance of smart city, crime fighting and smart farming. Most commercially used drones are equipped with high-resolution cameras that are used to visualize and monitor target status, e.g., object recognition, counting and tracking purposes.

State-of-the-art video analytics applications are increasingly using drone video data collection/processing that requires high-performance computing resources and real-time network communications. The problems of insufficient computing resources and network bandwidth at the edge can be addressed by using computation offloading to a Ground Control Station (GCS). As shown in Figure 1, we consider an experimental setup comprising of a fleet of drones connected via the Flying Ad hoc Networks (FANETs) [1]. A multi-UAV system can operate in a centralized or decentralized manner. In a decentralized system, the UAVs need to explicitly cooperate at different levels to exchange information, share tasks and make collective decisions in order to achieve the system goals under limited edge/fog resources. Under this setting, even if the connection from one of the drones to the GCS is interrupted, inter-drone communication is still possible, as outlined in [14].

However, edge computation offloading cannot always be used due to the dynamic nature of wireless channels and the energy consumption constraints on the drones for processing high-resolution images. In edge networks, a variety of environmental conditions may affect the video streaming from drones to the GCS. This can in turn jeopardize the

cqy78@mail.missouri.edu (C. Qu); calyamp@missouri.edu (P. Calyam); yu618@purdue.edu (J. Yu); avanda7@uic.edu (A. Vandanapu); osoykp@mail.missouri.edu (O. Opeoluwa); kegao@mail.missouri.edu (K. Gao); wangso@missouri.edu (S. Wang); rlc5m8@mail.missouri.edu (R. Chastain); pal@missouri.edu (K. Palaniappan)
ORCID(s):

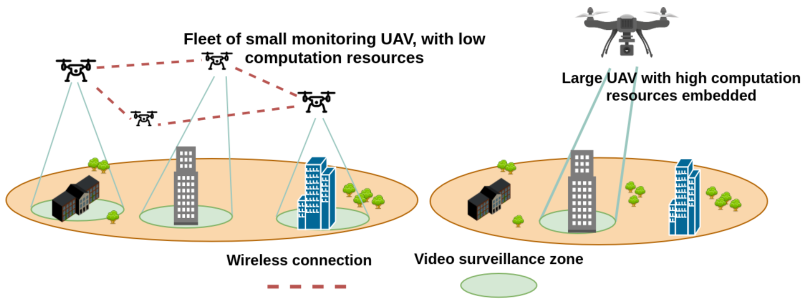

Fleet of small monitoring UAV, with low computation resources
Large UAV with high computation resources embedded
Wireless connection
Video surveillance zone

Figure 1: Search and Intelligence application scenario using multiple drones with diverse video cameras and sensing capabilities.

performance of video analytics with increased end-to-end delay, frame blurring, stalling and distortion rates. Network performance orchestration and drone-edge computation offloading strategies are always considered together to make sure the video analytics application performs in a stable and integrated manner to meet application requirements. Selection of a suitable network protocol is a key feature for resource orchestration along with an appropriate location selection for the computation tasks offloading in a multi-drone-edge network. However, based on our literature survey and observations from [17] and [27], there is a clear lack of mechanisms to efficiently coordinate the networking protocols selection in conjunction with drone control orchestration and edge/fog computation offloading during drone video analytics.

In this paper, we present a novel *learning-based dynamic computation offloading, control networking scheme* viz., “DroneCOCoNet”, which accomplishes: a) intelligent reinforcement learning-based processing in edge computation offloading strategies, and b) resource-aware network protocol selections based on application requirements. The intelligent processing considers the trade-offs in processing time vs. tracking/accuracy rate using a Function Centric Computing (FCC) architecture [5]. With the use of the FCC architecture, a given drone video analytics pipeline can be decomposed into separate functions for distributed processing at the edge GCS or on-board the drones using a suitable resource allocation scheme. The resource-aware scheme selection for a given drone video analytics application setting is primarily based on a set of options involving a combination of video compression and video streaming protocols. We summarize the two major novel contributions in our DroneCOCoNet work as follows:

***CO*: Learning-based Edge Computation Offloading.** We implemented a learning-based offloading scheduler based on data collected in the field to predict computational energy consumption on the drones. Supervised learning approaches, such as Kernel-Ridge regression, support vector machine, Gaussian process regression, and random forest regression, are used to predict energy consumption, processing time and transmission latency within a multi-UAV system. We measure the performance of these approaches on predicting the energy consumption and latency in real multi-UAV environments, and test various FANET setups with three mobility models, i.e., random way point, Gauss-Markov and Mission-plan based model [7]. In addition to supervised learning, we design a dynamic solution to determine how and when drones should offload tasks to improve accuracy while minimizing the cost in cloud communication. We formulate offloading decisions within a set of sequential decision making algorithms, and propose a novel solution using reinforcement learning (RL).

***CoNet*: Control Networking Decision Making.** To solve the initial problems in providing high/optimal system performance in various wireless video streaming scenarios, we propose a resource-aware protocol selection algorithm, which chooses the most suitable protocol combination at the beginning of drone video streaming and analytics processes. We investigate the features of various candidate protocol combinations on video data transmission and compression codec strategies, given details on how users may use the algorithm to make decisions when setting up their drone data processing applications. To facilitate the transferring of data for parallel execution, we implement a novel client/server application named QUICer. QUICer operates using HTTP/3 over the popular QUIC transport protocol [31], which allows faster data transfer and is used in many emerging web applications. QUICer consists of servers and clients, both of which support multiple workers.

The rest of the paper is organized as follows: In Section 2, we discuss related works. In Section 3, we give an overview of the DroneCOCoNet system. Section 4 presents *CO* on using supervised learning for heuristic computation offloading decisions and its extension that involves reinforcement learning. Section 5 introduce the *CoNet* contributions that include *QUICer* and its integration in an advanced drone video analytics application. Section 6 discusses the performance evaluation of the DroneCoCONet system. Section 7 concludes the paper.

2. Related Work

FANET Management. Unmanned Aerial Vehicle Networks [15] are autonomous networks formed by self-organizing aerial vehicles like drones or other aircraft. The UAV network is known as FANET (Flying Ad-Hoc Networks), and is a subset of the MANET (Mobile Ad-Hoc Network) [7]. Research on these networks has steadily increased over the recent years, especially in next-generation civil applications like consumer product delivery, autonomous and mobile environmental monitoring, search, rescue, and disaster management. FANETs support distributed wireless networks, which solve the problem of communication range restriction by allowing communication among the UAVs without the need for an infrastructure [1]. There is no doubt that with the proper collaboration and coordination of multiple UAVS in the FANET, the system would far exceed the capacity and capabilities of single UAV systems. Although there are several advantages of using FANETs over the single UAV-to-Ground Control Station (GCS) system architectures, several challenges emerge when implementing FANETs. Notable challenges particularly rise in systems with different requirements, such as total processing time, energy consumption, efficient communication and coordination between UAVs-UAVs and UAVs-GCSs based on mobility [23]. However, the challenges that accompany many emerging UAV applications cannot be solved alone by static optimization models, especially if they are applied in complex and adverse environments [24]. Our proposed DroneCOCoNet overcomes these challenges by selecting the best suitable schemes for computation and communication among various UAV networks, allowing saving on network bandwidth, energy consumption, and edge/fog computation resources during the execution.

Edge Computation Offloading in Multi-UAV Systems. Computation Offloading is the act of transferring computationally intensive tasks or functions to other platforms. In our previous work, we implemented Function-Centric Computing (FCC) and computation offloading policies in drone video analytics. The FCC paradigm involves decomposing applications into microservices that can be individually deployed onto edge resources [5]. Advantages of this architecture are that specific functions in a larger application that are computation-intensive and would cause high energy consumption can be offloaded. In our proposed system, the drones have the option to transfer tasks to another drone or the GCS. Low power devices trying to execute computation-intensive tasks will often deplete energy faster than what is expected by the user. Energy-awareness is especially important for drones since the development of battery capacity has stagnated, and we want to have the drones flying in the air for as long as possible for data collection [8]. Computation offloading overcomes such limitations of low-power devices such as drones [22]. With computation offloading, energy consumption of the drones can be minimized as shown in our prior work [36], without compromising the user experience quality at the application level. In our proposed scheme, we leverage this framework in the scheduling of the tasks in the multi-drone scenarios that we have considered for use of function-centric computing. Specifically, the novelty of our approach can be seen in the method used for the functions to be offloaded from the drones to the edge as a way to minimize energy consumption of the drones as long as the delay constraint is met. In addition, our method allows drones to offload functions to other drones to evenly distribute the energy consumption.

It is important to figure out how and when drones as well as other IoT devices should offload sensing tasks, especially if they are uncertain, to improve accuracy while minimizing the cost in cloud communication [27]. Many computation offloading strategies have been proposed for multi-access edge computation in terms of network packets transmission and system responsiveness through dynamic task partitioning between cloud data centers and edge devices [16]. In [6], the authors formulate offloading as a sequential decision making problem in the context of IoT devices, and propose a solution using deep reinforcement learning. Algorithms for deployment optimization of multi-drones in a FANET were developed in [33] by focusing on the amount and location of the drones. Assignment of tasks among local or edge devices while minimizing energy consumption and latency is also a trending problem in joint computation offloading and resource allocation. Authors in [8] formulate the problem as a new online Reinforcement Learning problem while considering both delay and device computation constraints. They also proposed a new strategy based on a Q-Learning algorithm, named QL-Joint Task Assignment and Resource Allocation (QL-JTAR) as a solution. This work is particularly relevant because we consider it as a baseline approach in the performance comparison of our heuristic solution. Besides these two approaches, authors in [35] also used various reinforcement learning approaches on dynamic decision making. Authors in [32] summarize and compare advantages/disadvantages of reinforcement learning-based mechanisms, supervised learning-based mechanisms, and unsupervised learning-based mechanisms in computation offloading methods. They considered the essential features such as performance metrics, case studies, utilized techniques, and evaluation tools in their comparison studies. Other solution strategies can be seen in [21], where the authors use a game theory approach to deal with delay sensitive applications in MEC. The novelty of our work in comparison to these prior works is in the use of reinforcement learning for making decisions with various reward conditions. We query the edge cloud server status during decision making for improving analytics accuracy, while also handling the constraints such as latency and power. We also test our proposed reinforcement learning strategy in real experiment testbeds.

Networking Protocol Selection in FANETs. The recognition of imagery taken from over-the-head cameras poses many challenging tasks compared with imagery taken by a fixed ground level camera because of low or high resolution imagery patterns [3, 4]. Herein, a major challenging task involves handling the large intra-class variation in activities including variations in resolution scale, target (e.g., visual appearance, speed of motion) and environment (e.g., lighting condition, occlusion) [34]. The meaningful structures in a video are extracted through unsupervised learning of temporal clusters and are associated with the metadata. A flexible dual TCP-UDP streaming protocol (FDSP) is used for high-quality video streaming [12, 11]. This FDSP approach results in lower packet loss compared to UDP-based streaming. The FDSP is particularly relevant to our work because it features a flexible dual TCP-UDP streaming protocol that requires application awareness of the critical video streaming content. The application awareness is used during the streaming for the TCP protocol configuration. However, FDSP may not fully satisfy our requirements in the video analytics procedure, which is the reason we do not use FDSP as a comparison method (See Section 6.4 for detail.) Similarly, we also consider application awareness in the hierarchical FANET environment for a different case i.e., to select the QUICer protocol configuration.

The performance of QUIC vs. TCP for achieving a better performance in video streaming was studied in [31]. Unlike the conclusions in [31], we have not found a case for limited bitrate conditions during our whole flight experiments. Specifically, we have not faced the situation where QUIC performance drops below the necessary frames per second (FPS) for the video analytics pipeline we use in our application. In our experiments, we assume that the GCS can handle the distance of drone flight paths that do not significantly degrade signal conditions and can also address low battery situations, and hence we are able to avoid drone connectivity issues that involve low bitrate transmission cases. Our approach involves transport protocol experiments involving changes in video resolution in order to avoid video impairments. In our proposed scheme, we specifically use HTTP/3 and QUIC protocols to generate a video transport pipeline and corresponding network monitoring to help with development of machine learning based prediction methods. We also use a reinforcement learning approach for dynamic decision-making to enhance the performance in the context of network speed prediction.

Despite several advances in recent years, FANET networks still have limitations that need to be overcome for their successful operation and wide adoption. The main limitation is energy consumption [18, 19, 25] because it limits the flight time of the drones. Other limitations include the speed of connection and the range of the signal transmission due to the mobility of drones, and their limited storage capacity.

3. DroneCOCoNet Framework

In this section, we first describe the vision of the DroneCOCoNet framework and discuss the key challenges that we address in a system deployment. Following this, we list the various modules in our system and explain the interactions among them, and present various use case scenarios of our system. Lastly, we discuss practical considerations for the deployment of the DroneCOCoNet in a multi-drone edge/fog computing environment.

3.1. Framework Overview for *Drone* Video Analytics

Figure 2 illustrates a detailed view to integrate our DroneCOCoNet system in a multi-drone-edge-server scenario, which involves communication and computation offloading between drone and the GCS connected to the edge server. A hierarchical FANET with both low capability search drones and high capability intelligence gathering drones is considered in our multi-drone communication setup. To make sure the drone video analytics application performs in a stable and reliable manner, we considered both the problems of network protocol selection as well as the drone-edge computation task offloading. In the other words, our overall methodology aims to solve two salient aspects of system resource orchestration: computation offloading (CO), and control networking (CoNet). Three main logical modules are included in our hierarchical FANET system: (i) the application data collection module, (ii) the resource-aware protocol selection module, and (iii) the learning-based offloading module. The first two modules represent the control networking part of the system (described briefly in Section 3.2 and detailed in Section 5), and the last module represents the edge computation offloading part (introduced first in Section 3.3 and detailed later in Section 4). The control networking part provides an environment for data transfers corresponding to various multi-drone-edge-server scenarios. After setting up the essential hardware, software, and network connections, we use a novel learning-based

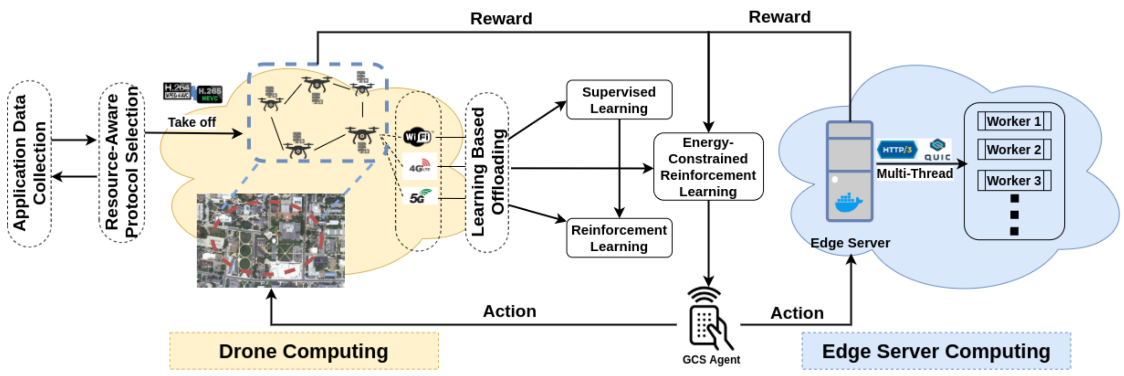


Figure 2: DroneCOCoNet system architecture showing the integration of data collection and processing through integration of two salient aspects of system resource orchestration involving a learning-based computation offloading and network protocol selection.

computation offloading algorithm module that we developed in this work to enhance the resource usage and system performance, such as e.g., energy consumption, analytics time and data processing accuracy.

3.2. Computation Offloading (*CO*)

To enhance system performance in the drone application, we use edge computation offloading to release the computation workload on drone and speed-up the video analytics. In real scenarios, in the case of uncertain wireless environments, it is difficult to predict the network conditions to give a precise decision on 'where' and 'when' to offload the computation-intensive jobs. However, using a machine learning approach, we could learn from previous flight traces with the network settings and video transmission properties to label the status and categorize the conditions. Moreover, a static condition may not be ideal for the entire flight due to the complex changes occuring in a dynamic field environment. Thus, we prefer to use a more dynamic approach for making decisions i.e., we use a reinforcement learning approach for learning from the dynamically changing environments by issuing certain rewards. In addition, integrating multi-thread capability could also enhance the processing speed of large data sets by running multiple independent tasks at the same time.

3.3. Control Networking (*CoNet*)

Considering various use cases of a multi-drone system, we select a few scenarios for using video-camera embedded drones as intermediate or front-end devices in applications such as e.g., traffic management, disaster management, and smart farming. In most of these scenarios, multi-drones cooperate together by recording and processing videos locally. However, drones mostly have low computational resources on-board to avoid large payload that limits the flight time. To cope with this limitation, they offload video data to the edge computing resources through the GCS. There can be several options for video transmission through the wireless channel. We could choose from different transport layer protocols and application layer protocols to enhance the speed and accuracy in video transmission. For instance in live streaming, we can choose from UDP, TCP, or QUIC as a transport layer protocol, and from RTP, HTTP/1, HTTP/2 and HTTP/3 as the application layer protocol. In addition, due to the limitation on the bandwidth of the wireless channel at the network edge, a compression of the drone videos is highly desirable before data transmissions. Although a more advanced video encoding/decoding algorithm could provide higher quality and/or save storage space, such an algorithm could be a bottleneck for supporting time-sensitive applications due to its longer encoding/decoding time. Thus, a control scheme to select network protocols and configure video data compression algorithms is essential for setting up environment variables before running the multi-drone-edge-server system applications.

4. Computation Offloading in Drone Video Analytics

In this section, we discuss computation offloading by using machine learning based prediction and task scheduling. We present two learning based approaches, i.e., (i) the supervised learning based algorithm, and (ii) the reinforcement learning based algorithm and detail the implementation of these approaches.

4.1. Supervised Learning Based Job Scheduling

4.1.1. Problem Background

We designed experiments (See Section 6) to acquire flight traces with various network setting and video transmission properties. Table 1 shows the features that were collected during our experiments. These features are divided

Table 1
Collected features from our drone experiments dataset.

Transmission Dataset	Processing Dataset
Analytics Layer Attributes:	
Video Resolution (width × height)	Video Resolution (width × height)
System Layer Attributes:	
Wireless Data Rate (Mb/s)	CPU Cloud Speed (MHz)
Number of Spatial Streams	RAM (GB)
Code rate (Mb/s)	Free Memory (MB)
Number of bits per symbol	Hard Drive Capacity (MB)
Mobility Models:	
Gauss-Markov, Random Way Point, Mission Plan based	
Result Parameters:	
Transmission Energy (J)	Processing Energy (J)

Table 2
Machine Learning model average training time (± RMSE) based on transmission and procesing dataset.

Model Type	Transmission (time)	Processing (time)
KRR	0.120±0.00362	**0.004±0.00272**
SVR-RBF	**0.099±0.01620**	0.068±0.00252
GPR	2.170±0.00100	1.962±0.00000
RFR	0.205±0.05400	0.156±0.05400

into three categories: (i) analytics layer attributes, (ii) system layer attributes and (iii) mobility models. In addition, we obtained the transmission energy results based on the calculations given in [8] and calculated the consumed energy during processing the data set using the given hardware settings.

With the collected data sets, we used machine learning to predict the time needed to complete each of the jobs. The reason why we compare the prediction time to complete each job is because we found that: (i) the ML models provide at least 95% accuracy on prediction given a time series of measurements and thus can be used in computation offloading decisions, and (ii) we run an on-line learning procedure on a given multi-drone scenario to ensure that the finish time (that is one of the most important features) is minimum to not disrupt the drone flight tasks. Four machine learning models from the Sci-Kit Learn Tool [26] were used: (i) Kernel-Ridge Regression (KRR), (ii) SVR-RBF (Radial Basis Function kernel SVM), (iii) Gaussian-Process Regression (GPR), and (iv) Random Forest Regression (RFR). The choice of selection of the supervised ML tools is motivated by the following reasons: First, in real-world scenarios, due to the uncertain wireless environment conditions, it is difficult to predict the network performance to give a precise decision on 'where' and 'when' to offload the computation-intensive jobs. However, using a machine learning approach (i.e., supervised learning), we could learn from previous flight traces with the network settings and video transmission properties to label the status and categorize the conditions. Second, the different machine learning methods are chosen based on the regression methods in machine learning approaches. We have chosen to focus on regression since most of the previous works (e.g., [27, 32]) involve measuring the performance of models using metrics such as the transmission time or processing time or the energy consumption. In these machine learning models, we were primarily concerned about the training time and RMSE (Root Mean Square Error) metrics, as shown in Table 2. For transmission time, we found that the machine learning model that had the shortest training time was the SVR-RBF model. For the processing time, the best performance was seen in the Kernel-Ridge Regression. Leveraging the predicted information from the machine learning models, we can obtain information on energy consumption by each job, network condition on each time-step, and use such information to allocate the resources from drones and edge servers.

Regarding to the model training, validation and testing procedure, we assume above that all of these activities will happen only on the GCS. The model training as part of the computation offloading algorithms is occurring in two different solutions: (i) the heuristic based on the machine learning results, and (ii) on-time reinforcement learning algorithm. In addition, we do consider the case where the UAV does not have enough battery in prediction. In such a case, we assume that the GCS will detect such a situation and not assign any computation offloading tasks for that UAV. In the event of a condition when UAV runs out of battery power during the flight, we assume that the UAV will be made to ground itself to be recharged before it is added back to the hierarchical FANET setup.

In the following sub-section, we introduce a heuristic algorithm, which uses a job-shop scheduling method based on our prior work [36] to allocate computation resources from drones and edge servers.

4.1.2. Heuristic Job Scheduling Algorithm

The results of supervised learning were used to predict the network conditions, with the aim to allocate the resources from drones and servers. With regards to 'where' to execute the jobs, we use this scheduling algorithm to allocate the jobs and minimize the final energy consumption. For this purpose, we designed a heuristic solution, with which we always choose the best place (i.e., 'where') to execute the job, and allows parallel execution as soon as there is no conflicts between any parallel jobs. After the heuristic algorithm, each job gets the location and order of execution. With regards to 'when' to execute the job, it is the same as the job-shop scheduling problem. There are many ways to solve the job-shop scheduling problem. In this work, we used existing tools [10] to calculate the scheduling sequence and the final execution time, as shown in Algorithm 1.

In this heuristic approach, we introduce a threshold for helping the system determine where to offload the jobs based on the remaining ideal flight time and the remaining battery life of the drone fleets. This algorithm's result is used to compare among the Local Only, Random, and Offload Only approaches; detailed evaluation results and observations are provided in Section 6.2. The basic logic of the algorithm is: when the total execution energy reaches the threshold value, it is not necessary to run the job-shop scheduling algorithm again in the next period. In this case, we choose a more advanced approach i.e., the learning based approach as detailed in Section 4.2.2. The algorithm used in the heuristic decision making approach with supervised learning based computation offloading is detailed in Algorithm 1.

Algorithm 1: Heuristic Algorithm with Supervised Learning based Computation Offloading

```
Input: [$jobSet$]:= total set of jobs and energy properties on a multi-drone analytics application; [$envSet$]:= set of all essential properties on drone system
       (battery, camera, and etc); $TE$:= predicted transmission energy of a job; $PE$:= predicted processing energy of a job
Output: [$COlocations$]:= a set of locations on which job running on where, [$totalEnergy$]:= total energy spend on an application
Function Main([$jobSet$],[$envSet$]):
    while $remainTime(envSet.battery)! = 0$ do
        foreach $job \in [jobSet]$ do
            $COlocations.job = $ Heuristic($job$,[$envSet$])
            $remainTime(envSet.battery) - -$
        end
    end
    [$totalEnergy$] $==$ $JopShopAlgorithm([jobSet], [COlocations])$
return
Function Heuristic($job$,[$envSet$]):
    if $job.energy < min(remainEnergy(envSet.battery), remainEnergy(envSet.flightTime))$ then
        if $job.TE >= job.PE$ then
            $COlocations.job.position == GCS$
        end
        else
            $COlocations.job.position == Drone$
        end
    end
    else
        $COlocations.job.position == Drone$
    end
    return $COlocations.job$
```

4.2. Reinforcement Learning based Job Scheduling

4.2.1. Problem Background

Since the heuristic approach is fixed during the entire flight of drones, it does not perform better than real-time decision making algorithms. Here we provide a problem statement to abstract the computation offloading problem as a Markov-decision process. The optimization problem is defined as: *find the optimal offloading control approach that maximizes expected cumulative rewards R*. Since in last subsection we already considered energy, which influences computation offloading, the Reward function for this subsection will only include accuracy and latency.

To overcome the disadvantage of the heuristic approach, we define the edge computation offloading problem as a finite-time Markov decision process (MDP) with regards to the total drone flight process.

$$M_{offload} = (S_{offload}, A_{offload}, P_{offload}, R_{offload}, T) \tag{1}$$

where $M_{offload}$ is the state space, $A_{offload}$ is the action space, $P_{offload}$ is the probability function that indicates the probability that action a in state s at time t will lead to state s' at time t+1. $R_{offload}$ is a reward function, and T is the problem horizon.

We consider the offloading decision problem to choose amongst options based on prediction to use for downstream tasks at time t. The offloading system action can either choose to use past predictions to avoid performing computation on the new input, or incur the computation or network cost to use either the on-device computation model or the cloud computation model. Three discrete actions of the offloading system (as shown in the $A_{offload}$ in the MDP problem) that we consider are listed in the following equation:

$$a^t_{offload} = \begin{cases} 0, & \hat{y}^t_{keep\_the\_same\_state} = f_{previous}(x^t) \\ 1, & \hat{y}^t_{drone\_prediction} = f_{drone}(x^t) \\ 2, & \hat{y}^t_{cloud\_prediction} = f_{cloud}(x^t) \end{cases} \tag{2}$$

We define the state in the offload MDP to include the information needed to choose between the actions. The choice depends on the current sensory input, the stored previous predictions, and the remaining query budget. The state in the offloading MDP is displayed in the following equation:

$$s^t_{offload} = \left[\underbrace{\phi\left(x^t\right)}_{features}, \underbrace{f\left(x^{\tau_{drone}}\right)}_{past\,drone}, \underbrace{f\left(x^{\tau_{cloud}}\right)}_{past\,cloud} \right] \tag{3}$$

The input is high-dimensional, and could yield an extremely large state-space. The specific choice is dependent on the expense associated with utilizing the chosen features, as well as the standard encodings or feature mappings.

Our objective using the MDP is to achieve a higher prediction accuracy compared to heuristic solutions, while minimizing both drone computation and network utilization. We can naturally express the goal of high prediction accuracy by adding a penalty proportional to the loss function, under which the cloud computing and drone computing models are evaluated. To model the cost of network utilization and computation, we add specific action costs. The reward function is given by adding two different constraints, model error and compute latency. Those two constraints are described inside the heuristic algorithm approach text portions. α and β add up as 1, which will control and balance the weights of each of these constraints. The weights in our experiments are set to 0.5 each to simplify the evaluation results. This ultimately gives us the reward function:

$$R^t_{offload}(s^t, a^t) = -\alpha_{accuracy} \underbrace{\imath(y^t, \hat{y}^t)}_{model\,error} - \beta_{cost} \underbrace{(cost(a^t))}_{compute\,latency} \tag{4}$$

In addition, since the MDP problem can be applied during the whole system operation, we also considered adding the total energy consumption as a third metric. The reason we did not consider energy in the heuristic algorithm is because of the work in [2], where the power of rotating rotors is considered to be 20 times the power of computing. Hence, in the context of the time for prediction, the transmission and processing are important on the heuristic side. However, considering the whole system, we can use energy as an additional metric to upgrade the system performance and improve the decision making accuracy. Similar to the first reward function, the updated reward function is given as follows:

$$-\alpha_{accuracy} \underbrace{\imath(y^t, \hat{y}^t)}_{model\,error} - \beta_{cost} \underbrace{(cost(a^t))}_{compute\,latency} - \epsilon_{energy} \underbrace{(\epsilon(e^t))}_{energy} \tag{5}$$

4.2.2. RL-based Job Scheduling Algorithm

Having formally defined the drone offloading scenario as an MDP, we can quantify the performance of an offloading approach in terms of the expected total reward it obtains. The problem can be expressed as in the following:

Given a drone computation model f_{drone}, cloud computation model f_{cloud}, a budget of over a finite horizon of T steps, and an offloading MDP $M_{offload}$, find optimal offloading control approach $\pi_{offload} : S_{offload} \rightarrow A_{offload}$ that maximizes expected cumulative reward $R_{offload}$:

$$\pi_{offload} \in argmax E(\sum_{T} R_{offload}(s_{offload}, a_{offload})) \tag{6}$$

Although we framed this problem as an MDP, it's not easy to apply conventional tools, such as dynamic programming for solving the MDP. This is because, many of the aspects of this problem are hard to analytically characterize,

Table 3
A subset of candidate schemes for prerequisites on various scenarios

Requirements Scheme	Application Protocol	Transport Protocol	Video Codec	Video Resolution	Description
H1_TCP_H264_720P	HTTP/1.1	TCP	H.264	720p	Slow, lossless video quality. Good for small analytics tasks.
H2_TCP_H264_720P	HTTP/2	TCP	MPEG-4	720p	Fast scheme, high security. Lossy video quality. Good for everyday survillance system.
H3_QC_H264_1080P	HTTP/3	QUIC	H.264	1080p	Very high security and fast scheme, lossless video quality. Good for crime detection and tracing.
RTP_UDP_MP4_1080P	RTP	UDP	MPEG-4	1080p	Low security, very fast scheme, lossy video quality. Good for traffic management.
H3_QC_H264_2K	HTTP/3	QUIC	H.264	2K	Very high security, lossless video quality and high resolution. Good for disaster scene response.
H1_TCP_MP4_2K	HTTP/1.1	TCP	MPEG-4	2K	Slow, low security scheme but with high video quality. Good for smart agriculture/farming.

notably the dynamics of the sensory input stream. This motivates our investigation of approximate solution techniques such as the model-free reinforcement learning [13] approach. There are several advantages for using reinforcement learning. First, model-free policy search methods such as RL avoid the need to model the dynamics of the system, especially the complex evolution of the drone's incoming sensory inputs. The model-free approach is capable of learning optimal offloading policies based solely on the features included in the state, and avoids the need to predict incoming images. Moreover, the use of a recurrent policy allows better estimation of latent variables defining the possible non-Markovian context of the incoming images. In addition, RL enables simple methods to handle stochastic rewards, which may arise in offloading due to various costs associated with network conditions or load on cloud compute servers. Finally, an RL approach allows inexpensive evaluation of the policy, as it is not necessary to evaluate dynamics and perform optimization-based action selection as seen in, e.g., model predictive control.

In our solution, we use the Q-Learning reinforcement learning algorithm, in order to learn a policy and guide the drones on actions to take to minimize energy consumption and system latency. Since this algorithm does not require a model, it can handle problems with stochastic transitions and rewards without requiring adaptations. Detailed implementation of the RL approach can be found in the GitHub repository [29].

5. Control Networking in Drone Video Analytics

As previously stated, achieving optimal system performance in multi-drone applications in the wireless video streaming scenario requires high bandwidth and low transmission delay. This is a challenging task considering wireless transmission factors such as low bandwidth in lossy wireless channels, delay, lack of coverage and congested networks. Our hierarchical FANET setup can overcome those challenges by supposing that the search drones will act as a sensor network that communicates with the GCS. Upon detection, drones with higher capability will be used to survey the area. These intelligence gathering drones therafter are supposed to capture high quality video in a stable network connection environment. In the following, we describe a reliable and seamless connection and load-balanced decision making scheme in drone video analytics.

5.1. Protocols Background

For choosing amongst the various transmission protocols and video codec types at the beginning of the application, our approach considers various options to switch configurations based on the application context and requirements. The candidate choices represent one option in each of the four categories, including: (i) Application-layer Protocol (choice between HTTP/1.1, HTTP/2, and HTTP/3), (ii) Transport-layer Protocol (choice between TCP, UDP and QUIC), (iii) Video Codec (choice between H.265 HEVC and H.264 AVC), and (iv) Video Resolution (choice between 720p, 1080p and 2K). Through combination, the possible candidate protocol choices for the requirement schemes can be a large set. Table 3 shows a subset of the candidate schemes and provides a brief description for these schemes. The motivation of the choice for each category is given as follows:

Application-layer Protocol. The protocol selection allows a choice between HTTP and RTP. The difference in performance and bandwidth requirements of various application-layer protocols can get noticeably high when considering the impact of the ideal encryption, video quality and authentication schemes. Instead of web-based video data transmissions, other wireless video streaming protocols such as RTP are also widely used in Multi-access Edge Computing focusing on accessing player buffer status and relevant video content information, such as frames priorities and coding dependencies.

Transport-layer protocol. The protocol selection allows a choice between TCP, UDP and QUIC. Delivery of packets in the same order is guaranteed with the TCP protocol. Due to the secured connection, stalling is witnessed in the video streaming from the source (i.e., drone) to the destination (i.e., the GCS). In contrast, UDP is a connection-less and unreliable protocol. During the transmission of video by UDP, image or video blurring is observed due to the loss of data packets/information. Compared to TCP and UDP, the QUIC protocol delivers data in a reliable, secure and fast manner. Our objective is to analyze the performance differences among the QUIC, UDP,and TCP protocols and create an application that meets the necessary frame rate requirement in drone video analytics. More discussion on the transport protocol selection is provided in Section 6.4.

Video Codecs. Video compression is also an important consideration to reduce the capacity of video content transmission in a wireless network setting. A variety of video compression formats can be implemented on IoT devices, including drones. In our experiments, we consider two categories of video codecs - lossless and lossy. In general, a lossless video codec will take large amounts of space for storage and could easily fill up the channel bandwidth. On the contrary, a lossy video codec will save space but may loose some information, which may not be acceptable in application scenarios, e.g., smart agriculture. We prefer to use H.265 HEVC lossless codec and use H.264 AVC lossy codec as the main choices in our experiments.

Video Resolution. Lastly, the choice of video resolution is important to ensure integrity of the data being sent from the source (i.e., drone) to the destination (i.e., the GCS). The available options are 720p, 1080p and 2K, which are the commonly used video resolutions in recent times in applications. The reason why we did not prefer to use 4K is that - currently not all video cameras on commercial drones support 4K video capture. Overall, through permutation and combination, the possible choices for the satisfying requirements can be a large collection of scheme options. Table 3 shows a subset of candidate schemes along with a brief application related description for each of those schemes.

5.2. Use Cases for Network Protocol Selection

Recall that our scheme allows choosing of the network transmission protocols and video codecs as per the users' requirements. However, in a real-world application, the decision-making process is quite difficult. This can be attributed to the the fact that users might not be familiar with the encryption algorithms, and the inherent advantages or disadvantages in choosing amongst the various network protocols. We suppose that the drone application users lack of the knowledge to choose from the optimal network properties and video properties. Hence, we provide a subjective use case requirement table and provide discrete levels for them to choose from. The dataset of the network and video properties selection is based on the drone flight traces we generate from drone flight experiments in our previous work [28]. Consequently, users need to be presented with relevant information of suitable candidate options to choose the best option. To assist with this process, we propose a "Resource-aware Protocol Selection Algorithm" as one of our DroneCOCoNet components. We identified ≈90 possible scheme choices, from which the Resource-aware Protocol Selection Algorithm will choose when setting-up a protocol configuration for a given Drone-GCS application. After the user provides the choice of the application requirements, we run the ML algorithm to select the network and video properties choice in the application configuration.

As part of the solution approach, we use both single thread and multi-thread programs to categorize and simplify the real-world drone video analytics applications. The number of threads is determined based on the ratio of drones and the GCS nodes available in the field. We categorize the drone to GCS ratio in three modes: one-to-one, one-to-many and many-to-one. In our case, if on average an application has more drones than GCS nodes, it is considered as a many-to-one mode, and vice versa. Both of these situations are used in the context multi-thread programs and involve parallel execution. Surprisingly, if the drone itself is programmable, we could also provide a multi-thread program solution on the one-to-one mode. In general, a single thread program approach is simpler and easier to design and setup. In our QUICer (see Section 5.4), we will mainly focus on multi-thread program solutions.

For our system to choose from the available options, it is essential to collect a user's requirement for the various parameters in the system. Table 4 shows the parameters used for client benchmark analysis in our scheme, which we categorize into three different layers, i.e., (i) the analytics layer, (ii) the system layer, and (iii) the energy layer; each of these layers contains multiple parameters. The values of these parameters represent the user's desired level for a given feature, with a higher level meaning that the user prefers to have the feature, and a lower level signifies that a particular feature is not necessary. The 7 metrics under the 'Parameter' column in Table 4 can be used to form the cluster of devices. By calculating the mean values of each variable for each cluster, we can classify them into intermediate categories. In other words, for each desired requests from users, we will have one solution for them, which is categorized by the rate generated during our experiments and learning results. The detailed categorized selection and decision making results can be seen in our prior work in [28]. We use ML models to process real-world drone traces that include various mobility models, geospatial link information and on-time network status obtained

Table 4
User Provided Parameter Values and Description

Category	Parameter		Values	Description
Analytics Layer	P1	Real-Time Analytics	1 3 5	No Real-time Analytics Needed Accept Slightly Response Delay Fast Response, High Real-time Needed
	P2	Video Quality	1 3 5	Survillance Used Low Quality Video Entertainment Level Used Video High Quality Scientific Used Video
System Layer	P3	Parallel Level	1,3,5	Drone to Server Ratio: One-to-One, Many-to- One, One-to-Many
	P4	Bandwidth Usage	1-5	[0Mbps - 50Mbps), [50Mbps - 100Mbps), [100Mbps - 500Mbps), [500Mbps - 2Gbps), [2Gbps+)
	P5	On-Flight CPU Level	1-5	[0MHz - 75 MHz), [75MHz - 250 MHz), [250MHz - 750MHz), [750MHz - 1.5GHz), [1.5GHz+)
Energy Layer	Total Battery Capacity		Actual Number	
	Ideal Flight Time		Actual Number	

from real-world data-gathering efforts. We remark that the learning results details are explained later in Section 6.4.

5.3. Resource-aware Protocol Selection Algorithm

Once user requirements are collected, the protocol selection algorithm shown in Algorithm 2 is invoked to choose a scheme from the candidate scheme catalog. Three major threshold variables have been set up to help make decisions, mapped on a scale of 1 to 5. The threshold value to decide whether the application requires real-time analytics can be set using the variable *real_time_threshold*. For instance, a threshold of Level 3 implies that this specific application requires a drone with video analytics computing resources to transmit videos and receive instructions asynchronously. Similarly, *video_quality_threshold* is used to identify the video codec and video resolution so that the combination could satisfy the user's accuracy requirement *accuracy_threshold*. All the decisions and threshold variables are created based on the analytics layer requests/settings.

Algorithm 2: Resource-aware Protocol Selection Algorithm

```
Input: $Video$:= video data to be packaged; [$systemSet$]:= set of system layer properties, with weight; [$analyzSet$]:= set of analytics layer properties, with
       weight; [$envSet$]:= set of all essential properties on drone system (battery, camera, and etc)
Output: $P$:= protocol to be used for transmission; $VC$:=video codec to be used for compression;$VR$:= video capture resolution(default setting)
Function Main($Video$, [$systemSet$],[$analyzSet$]):
    $Data \leftarrow Video$
    if $systemSet[parallelLevel] >= 3$ then
        $Thread()$
        VideoDecision([$analyzSet$])
        $P.trans == QUIC$ && $P.app == HTTP3$
    end
    else
        VideoDecision([$analyzSet$])
        if $analyzSet.real\_time <= real\_time\_threshold$ then
            $P.trans == TCP$
            $P.app == [envSet].auth == true?(HTTP2 : HTTP1)$
        else if $envSet.scenario <= accuracy\_threshold$ then
            $P.trans == UDP \&\& P.app == RTP$
        else
            $P.trans == QUIC \&\& P.app == HTTP3$
        end
    end
    $exe(Video)$
return
Function VideoDecision([$analyzSet$]):
    if $analyzSet.videoQuality <= video\_quality\_threshold$ then
        $VC == H.264$
        $VR == envSet.camera < 1080P?(720p : 1080P)$
    end
    else
        $VC == H.265 \&\& VR == 2K$
    end
    return $VC, VR$
```

As shown in Algorithm 2, the inputs include: video data to be encoded and transferred, $Video$, the system layer variables, [$systemSet$], the analytics layer variables, [$analyzSet$], and the environment properties of drone systems, [$envSet$]. [$systemSet$] and [$analyzSet$] are given as system settings and user requirements for a specific application scenario. [$envSet$] is given by the drone system and the application context. In our case, [$envSet$] includes the camera

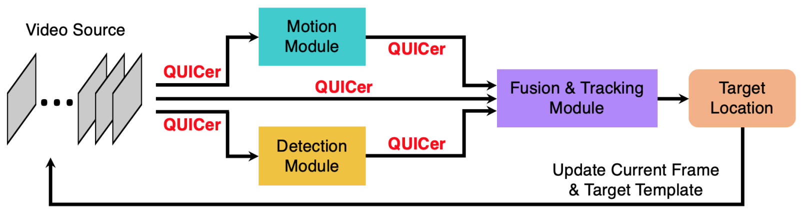


Figure 3: Visual object tracking pipeline that includes the modules integrated with QUICer for high-performance object tracking.

information, battery information and scenario description. We may not get the exact accuracy requirements from the [$envSet$], since it is dependent on the video analytics algorithm. However, we could get a preliminary decision on which a transmission protocol can be used to avoid the transmission being a bottleneck during the video analytics pipeline execution. The output of the algorithm provides the specific application and transport layer protocols to be used for drone video transmission, video codec selection, and the recommended video resolution.

5.4. QUICer: Application Pipeline Data Movement Acceleration

In this section, we introduce QUICer, a new video transmission tool that we developed to integrate HTTP/3 and QUIC protocols with a multi-thread program capability. To show application integration ability of QUICer, we also present an advanced video analytics application pipeline that uses the QUICer tool for video data transmission.

5.4.1. QUICer Implementation

In most cases, simply making decisions on video properties and networking protocols cannot ensure optimized system performance. One solution is to use a multi-thread program to exponentially speed up the video analytics process. To achieve parallelism in data transmission within a drone video analytics pipeline, we propose a novel python-based tool viz., QUICer that builds upon the popular QUIC transport protocol. QUICer operates by using HTTP/3 over QUIC, thus enabling faster transfer rates, and consists of a client/server program, which support multiple workers to allow multi-threading capability in data transmission over a network path between the drones and GCS.

5.4.2. Application Pipeline Integration

We integrate QUICer into a visual object tracking pipeline, which consists of 3 application modules to detect and return the location of a specified class of objects (e.g., cars) in a video data source. Each module includes HTTP3/QUIC client/server processes that accept data from upstream processes and direct output to downstream processes.

Video Source. The video source for our pipeline is a video feed from a multi-drone client. This video is simultaneously copied to all other devices for processing. To achieve a better result, we used the H.265 codec to compress the video source and set the video resolution at 2K that is commonly used in drone video systems.

Motion Module. The motion module performs image-based motion detection on the input video. For each frame in the video, we established a local 3D spatial-temporal volume and computed the information about temporal gradient changes, which can be used to distinguish between moving and stationary regions. Subsequently, a binary mask showing the moving objects was generated and sent to the Fusion & Tracking module. Motion detection can greatly improve object tracking performance especially for the hover mode of UAVs where the video is stabilized.

Detection Module. In this module, we employed YOLO [30], a deep learning-based object detector using a convolutional neural network, for object detection to enhance the tracking performance. We used the weights that were pre-trained on ImageNet [9]. Note that in the practice of our drone video analytics experiments, we use VisDrone [37] dataset as our object detection testing dataset. This is because the VisDrone dataset provides bird-view images that are different from the multi-angle images in the ImageNet dataset. The detection module can increase the tracking accuracy by refining the object bounding box. We also use the detection module to restart the tracking process when the tracker loses the target.

Fusion & Tracking Module. In this module, we operate the visual object tracking on a selected target. We used the CSRT [20] as a single object tracker that relies on distinctive object appearance for tracking. CSRT is able to handle challenging scenarios such as object deformation, fast camera motion, low video frame rate, and occlusion. We utilize the information provided by both motion module and detection module to assist the tracker. We have developed a rule-based fusion model that can effectively fuse the motion, detection, and appearance cues and produce optimal tracking results.

Table 5
Extended metrics related to the real-flight experiment dataset.

Name	Content
Attitude Data	Mainly includes the navigation information including location, speed, attitude, angular speed, etc. and sensors information including accelerators, gyroscope, magnetometer, barometer, etc.
OSD Data	Mainly includes the flight status information of the drone, such as GPS signal strength, flight status, and Return to Home status, etc.
Remote Controller Data	Mainly includes the remote controller's information, such as flight mode, and controller stick's movements, etc.
Battery Data	Mainly includes the battery's information, including voltage, current, temperature, capacity, etc. Only consider battery cost on computing
Obstacle Avoidance Data	Mainly includes the Obstacle Avoidance information of the vision sensors, ultrasonic sensors and infrared sensors.

6. Performance Evaluation

In this section, we first introduce the evaluation setup and data sets collected from a real drone use case scenario. Then we discuss the heuristic based scheduling results on edge computation offloading. In addition, we present the RL approach results compared to the heuristic solution, and provide a comprehensive solution by considering together the accuracy, latency and energy metrics. Lastly, we present the control networking experiments, where we compare QUICer with TCP and UDP connections and show the advantages of using our QUICer approach.

6.1. Evaluation Setup

For evaluation of our DroneCOCoNet framework, we built a hierarchical FANET environment, where we consider various mobility models and increased drone to GCS server ratios. To simplify our experiments, we collected 20 video clips with 40 seconds and 25 FPS. 10 of the video clips are with low density of objects, and the other 10 video clips are with high density of objects. For each of the 10 video clips, we use 70% of the frames for testing and 30% of the frames for training, with 10 fold validations. Those videos are collected from 3 various version of drones (DJI Phantom 3, DJI Mavic 2 and Parrot Drone), each with 1000 image frames as the same settings in VisDrone dataset [37]. These 20 video clips represent 3 mobility models, i.e., random way point, Gauss-Markov and mission plan based, with different setups shown in Table 3. In addition to using the parameters collected in Section 4, we also query cost and mean classification loss during the flight for calculating the reward function in the MDP problem. For the energy evaluation, we measure the total flight time when using a battery, and calculate the energy consumption per second.

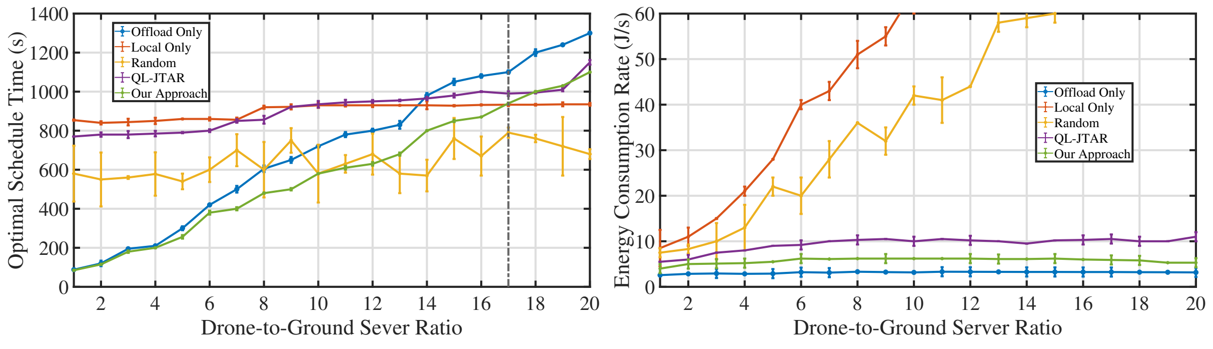
We have noted that the energy consumption in the flight is significantly larger in comparison to the energy consumption for the prediction. Consequently, the main factor in the decision to ground a UAV is based on the flight pattern and related battery charge necessary to complete a computation task. Further, in the evaluation section (Section 6.3), we use energy consumption as an add-on to our updated reward function, and assume that the energy consumption is not a major factor in the drone video analytics side but it is a significant factor on the environment side.

The real drone flying path was tested using three mobility models to predict the network environment, transmission time and energy viz., Random Mobility model, Gauss-Markov Mobility model and Mission-Based Mobility model. The flight time, flight mode, GPS satellites number, speed, battery consumption percentage and voltage were recorded. A realistic drone based data collection and associated formats are shown in Table 5.

In terms of the benchmarks for comparison purposes related to computation offloading, we considered a baseline and other state-of-the-art approaches. First, we consider three simple solutions offloading decisions are made on either a random basis, offload only basis or a drone-only basis. In addition, for comparison of our heuristic based scheduling results, we select a state-of-the-art approach viz., *QL-JTAR* as a side-by-side comparison method. Further, for comparison of our RL approach, we select an *Oracle baseline* as an ideal upper limit. More details on the RL approach related baselines are provided in Section 6.3.

6.2. Heuristic based Scheduling Results

Figure 4 shows the evaluation results with our heuristic solution in a real multi-UAV environment. Based on the results data, we conclude the following observations:



(a) (b)

Figure 4: Offload only, local execution (drone) only, random approach, QL-JTAR learning based approach and our heursitic approach results of: (a) increasing drone to ground ratio (Drone:GCS) and optimal schedule time (s) and (b) Drone:GCS and average energy per drone (Joules)).

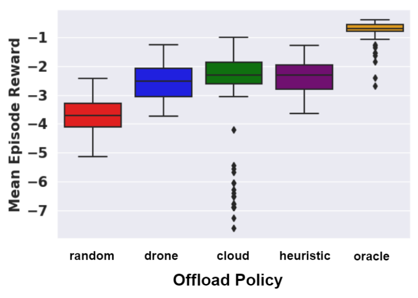
Observation 1: Our heuristic-based offloading scheme outperformed local-execution and Q-learning based QL-JTAR solution on scheduling time for low drone-to-ground-server ratios. As shown in Figure 4a, our approach follows a similar trajectory as the Offload Only, which offloads all of the tasks to the ground (i.e., the GCS connected to the edge server). We found that our heuristic-based offloading scheme resulted in up to 15% better scheduling make-span than the offload-only scheme. Random offloading is sporadic with no definable correlation and contains a far greater degree of variance compared to the other offloading schemes. The optimal scheduling time is very consistent when processing is performed only on the drone locally, even when the drone to ground server ratio changes. This is because of the parallel execution of the tasks done on the drones, with each drone processing its own video stream without ever offloading to the ground. Consequently, there will be no build up of the queue on the ground. In addition, as we can observe from the QL-JTAR solution results, QL-JTAR solution takes ≈800 seconds in all experiments, which is not efficient enough for the low ratio situation. When we consider significantly more drone flight paths on the area, QL-JTAR may perform better than our approach. However, the difficulty on operational drones will exponentially increase when the ratio reaches 10. Hence in our experiments, we only considered low drone-to-ground-server ratio cases.

Observation 2: Our heuristic-based offloading scheme outperformed Random offloading and Local Only execution on energy consumption, and provides reasonable energy savings compared to the QL-JTAR solution. From Figure 4b, we can see that the energy consumption rate grows at a very fast rate as the air to ground server ratio increases in the Local Only and Random schemes. However, our approach consumed energy consistently at a lower rate than the Local Only and random offloading schemes. However the energy consumption was slightly higher than the Offload Only scheme. This is reasonable because the energy consumed by the Offload Only scheme involves just the transmission energy, but not the energy used for video processing. In addition, this difference in energy consumption between our approach and the Offload Only scheme is fairly marginal. Considering that our approach performs up to 15% better than the Offload Only scheme in terms of scheduling time, we conclude that our approach overall performs better when both time and energy are considered.

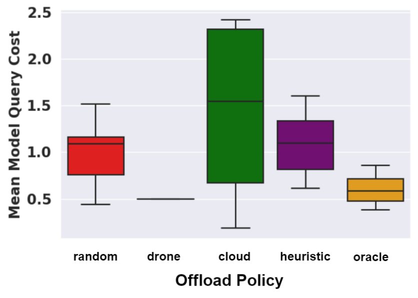
From both the above observations, we can see that our approach consistently outperformed the Local Only scheme and the QL-JTAR approach until the drone-to-ground server ratio reached $\approx 17 : 1$. This is because, as more drones are connected to the server with each recording a live stream, more tasks will naturally be offloaded to the ground GCS. For large drone to ground ratios, the queue on the ground will grow very large, meaning that it will take a long time for all tasks in the queue to be completed. Thus, we designed a dynamic computation offloading approach based on reinforcement learning, which we introduced in Section 4.2 and whose performance evaluation is presented in the following section.

6.3. Reinforcement Learning based Scheduling Results

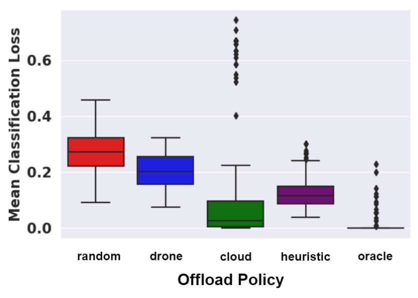
Similar to the comparison methods introduced in the context of the heuristic algorithm evaluation, we again use the random, drone-only and offload-only approaches as our baseline methods. As we can observe from Equation 2, during the procedure for making action decisions on MDP, those three baseline approaches can be achieved by selecting a pre-defined action a^t without learning procedures. In addition, besides those three baseline approaches and the heuristic itself, we also introduce an oracle approach as a upper limit bound which provides a standard for our learning-based algorithm. Consequently, we compare our RL-based approach against the following baselines:



(a)



(b)



(c)

Figure 5: Experiment results under different baselines: random, drone-only, cloud-only, heuristic and oracle showing comparison in terms of metrics: mean episode reward, mean model query cost and mean classification loss.

Random Baseline $\pi^{random}_{offload}$. This simple benchmark chooses a random action at offload 0,1,2 when the cloud query budget is not saturated and thereafter chooses randomly from actions 0 - 2.

Drone-only Baseline $\pi^{drone}_{offload}$. The Drone-only baseline chooses at offload = 1 at every time-step to query the drone computation model and can optionally use past drone predictions at offload = 0 in between.

Cloud-only Baseline $\pi^{cloud}_{offload}$. The cloud-only baseline chooses at offload = 2 uniformly uses the past cloud predictions at offload = 1 in between.

Heuristic Baseline $\pi^{heuristic}_{offload}$. This baseline uses previous approach on applying supervised learning results on decision making at the beginning of the experiment. It will choose a heuristic for selecting the most optimal solution at the beginning of each flight without changing the decision through the flight. Depending upon the characteristics of each trace in the traces dataset, the heuristic solution on model error could achieve different results.

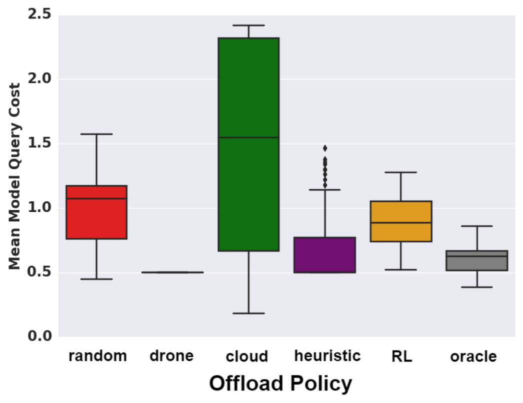
Oracle, Human-selected Baseline $\pi^{oracle}_{offload}$. This baseline uses human-selected approach, which achieves 100% accuracy on object/pedestrian recognition and registration, and is used only for comparison purposes. Here, we ignore the calculation time on reinforcement learning decision making and always use the ideal location for computation. On the other hand, oracle baseline cannot be achieved in the real-world experiments since the location of the object and the registration are given in advance. In general, the results will vary for a given trace dataset in terms of model error.

To summarize, offload-only and drone-only approaches consider the lower bound of metrics such as e.g., compute latency and model error. Whereas, the oracle considers an upper bound of the algorithm, which results in an ideal solution, where the performance is known in advance and the selection is given manually by the users.

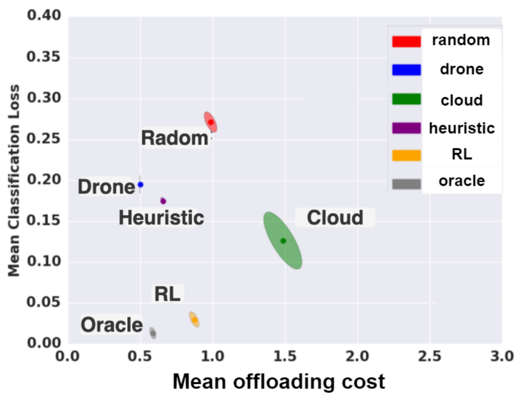
Observation 1: Our heuristic-based offloading scheme Pareto-optimizes the trade-offs compared with drone-only execution and offload-only execution in terms of computation latency and performance in the video analytics application. Figure 5 shows the comparison of baseline approaches with heuristic approach only by calculating their overall rewards generated during the whole flight period in the testing trace dataset. For example, Figure 5a represents the mean episode reward R with the components of *compute latency* (represented as query cost in y-axis in Figure 5b) and *model error* (represented as classification loss in Figure 5c). In Figure 5a, the heuristic baseline performed slightly greater than the cloud-only and the drone-only baseline. This is because the heuristic approach predicted the environment in advance and scheduled the jobs to save the resource to the maximum extent. However, as a trade-off, the heuristic approach has lower performance compared to the drone-only baseline with regards to the query cost (see Figure 5b) and the cloud-only baseline in accuracy (see Figure 5c). This is because of its scheduling time performance and the presence of a fixed job queue, regardless of the long waiting time involved with complex jobs.

To summarize, without the learning procedure, our heuristic baseline Pareto-optimizes the trade-offs between computation cost and video analytics performance compared with drone-only, cloud-only baselines. To achieve better results, we run experiments using our RL approach, which provides better performance than all the other lower bound comparison methods.

We evaluated the performance of the RL approach against the above baselines on 10 diverse traces in the testing set. In each of the traces, the number of objects and their locations were distinct from the traces in the training set. To test an offloader's ability to adapt to various network bandwidth constraints, we evaluated each trace with 5 various network condition settings, which map to the $P4$ values range in Table 4. Given that we use different traces, the heuristic and

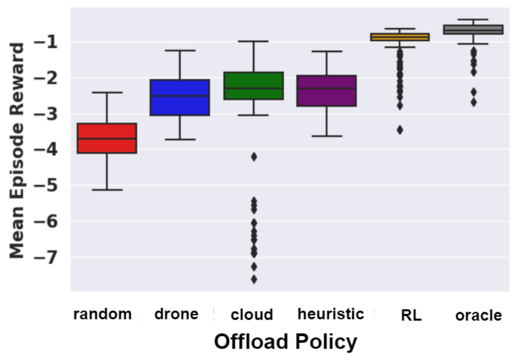


(a)

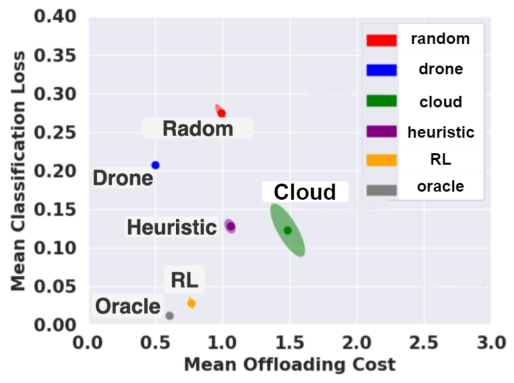


(b)

Figure 6: Experiment results on testing traces using different baselines: random, drone-only, cloud-only, heuristic, oracle and our proposed RL approach. Comparisons are made for *mean model query cost* and *mean classification loss* **without** energy concern.



(a)



(b)

Figure 7: Experiment results using different baselines: random, drone-only, cloud-only, heuristic, oracle and our proposed RL approach. Comparisons are made for *mean episode reward*, *mean model query cost* and *mean classification loss* **with** energy concern.

oracle solutions could get different model errors. However, the other three baselines (Random, cloud-only, drone-only) performance can be seen to be nearly the same since each of them uses the same underlying decision making strategy. In addition, comparing Figures 5b and 6a, we noticed that the heuristic algorithm provides unstable results (i.e., it produces less query cost than training the database). Moreover, we found that the oracle could provide similar results in the low range with regards to the query cost metric.

Observation 2: In the case without energy concerns, our RL approach Pareto-optimizes the trade-offs between computation cost and video analytics performance during the application procedure. In the case with energy concerns, our RL approach outperformed all other non-oracle baseline approaches considering all features in terms of computation cost and video analytics performance during the application procedure. Figure 6 shows test results for our RL approach based on the MDP problem setting in Section 4.2.1. When concerned about both application accuracy and mean offloading cost, our RL approach Pareto-optimizes the trade-offs between performance vs. computation offloading cost during real-time drone video analytics. Note that we only consider the latency when performing image processing, instead of considering the overall latency for both transmission and processing. As we can observe from Figure 6a, in the case without energy concerns, RL approach cannot outperform on *compute latency* although a notable improvement on *model error* is obtained. This is because, in the case without energy concerns, RL approach will put all the efforts on getting the video analytics accuracy which will in turn lead to overall higher energy consumption.

To balance the performance in terms of *compute latency*, *model error* as well as the overall energy consumption, we introduce one more feature i.e., energy consumption to balance the reward function in MDP. We can infer that - the higher the weight on energy consumption, the lower the chance that the RL approach will query for offloading, and lower will be the query cost.

Figure 7a shows the distribution of rewards obtained by the different offloading approaches using the new reward function (Eq. 5), where our RL approach is depicted in the orange boxplot. The mean episode reward is calculated

Table 6
Policy-based estimate (baseline) and machine learning prediction results comparison.

Model Type	Protocol 95% CI	Video Property 95% CI	PSNR
KRR	(0.652, 0.928)	(0.803,0.927)	(39.86, 46.93)
SVR-RBF	(0.779, 0.833)	(0.9034,0.952)	(33.59, 46.93)
GPR	(0.813,0.882)	(0.7523,0.8)	(32.26, 39.86)
RFR	(0.9124,0.96)	(0.9032,0.97)	(32.26, 49.37)

by the reward function in Equation 4. We also break down the mean episode reward into its two components, i.e., (i) prediction accuracy represented by the Mean Classification Loss, and (ii) latency represented by the Mean Offloading Cost. We show the mean performance over all testing clips for each approach in Figure 7b.

Observation 3: Our results show that our RL approach has at least 240% higher median episode reward than other offloading schemes, and is comparable with the lower bound of the oracle solution. Compared to the heuristic solution, RL approach is 1.9x higher on total rewards (see Figure 7a). This is because, the RL approach evaluates the cost dynamically on 'where' to execute the job at each time step by comparing trade-offs on accuracy, latency as well as energy. By comparing Figure 6b and Figure 7b, we can see that when considering the energy parameter, RL approach achieves better results on mean offloading cost. This is because, there is less opportunity on changing executing position in a stable environment. Essentially, the RL approach learns to judiciously query the remote server when the drone model is highly uncertain. This allows it to improve the overall system accuracy and minimize prediction loss. However, the RL approach has lower offloading cost since the bandwidth limits cause the drone-only baseline to periodically, but sparsely, sample the prediction time period. On the contrary, the RL approach has better prediction accuracy than the cloud scheme. This is because, the past prediction can keep the previous result for reference, which consequently keeps the object tracking and recognition to be more accurate and increases the overall processing speed.

To summarize, the experimental results show that our heuristic-based offloading decision-making scheme enables lower scheduling time and energy consumption for low drone-to-ground server ratios. In addition, our dynamic reinforcement learning-based decision-making approach increases the accuracy and saves overall time periodically, as well as provides energy efficiency. We remark that these results can also hold in various multi-UAV scenarios with different numbers of detected objects in e.g., smart farming, transportation traffic flow monitoring and disaster response.

6.4. QUICer Based Control Networking Results

Before we introduce QUICer as a communication strategy, we investigate how machine learning methods [1] could help us categorize the user requirements and provide an appropriate transport protocol choice (choose between TCP and UDP), and video property choice (category of video resolution and video codec). Table 6 provides the machine learning model results on networking protocol and video properties selection. We compared results with real-world experiments for each model and also calculated the 95% CI of accuracy for each model. We firstly conclude that - RFR gives more accuracy on networking protocol and video property selection, however the decision performance is in the unstable range of PSNR within the transmitted video. Secondly, we conclude that the KRR can generate better results with a stable range of PSNR, although the performance is lower than other machine learning models. There exists an advanced protocol which could dynamically decide either to use TCP or UDP during the application execution viz., FDSP. However, our multi-drone video analytics application scenarios do not benefit from the use of FDSP as a communication protocol strategy. There are primarily two reasons: (i) we cannot distinguish (in real-time) the critical part to assign the TCP protocol during the drone video streaming, and (ii) in the case of the limited bitrate condition, during our whole flight experiments, we did not face the situation where QUIC performs with less FPS in the video analytics pipeline than in case of FDSP. To elaborate on the second reason, the only situation where the limited bitrate performance is lower than expected is when the drone flies out of range, which is an undesired situation in our experiments.

QUICer's performance is significantly improved by using multiple connections between the client and server. QUICer supports asymmetric setups, e.g., having 4 server workers and 8 client workers, granting it the ability to adapt to the application. In our current implementation, QUICer requires the client and server to know the number of

[1] *KRR: Kernel-Ridge Regression; SVR-RBF: Radial Basis Function SVM; GPR: Gaussian-Process Regression; RFR: Random Forest Regression*

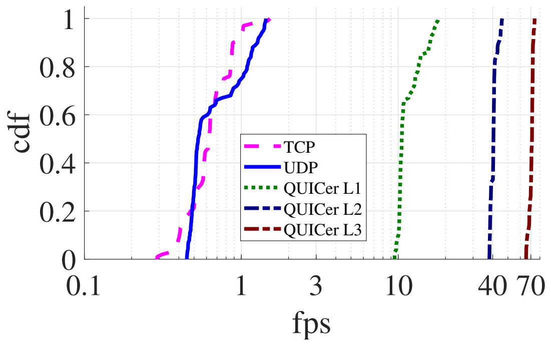


Figure 8: Frame rates CDF achieved by QUICer when using a varying number of simultaneous connections between server and client, compared with frame rate achieved with TCP and UDP performance.

connections prior to launching. However future versions could utilize a brokering endpoint on the server, allowing the client to dynamically adapt to the needs of the server. We remark that QUICer can be used to improve the processing speed of an object pipeline in terms of the communication level. Thus, any video processing pipeline which leverages function centric computing like we do in this work, can utilize QUICer to speed-up the processing tasks.

As shown in the Figure 8, we tested different levels of simultaneous connection between multi-drone and GCS system using our multi-thread program that is part of our QUICer implementation. QUICer L1, L2 and L3 represents 1 connections, 4 connection and 8 connections, respectively. Compared with traditional transport protocols such as TCP and UDP, the QUICer could achieve 12x to 140X improvement in video data transmission. This is primarily due to two reasons: (i) Owing to the one-time hand-shake feature of the QUIC protocol, the video streaming and data transmission experience a speed-up in comparison with TCP connections, and (ii) QUIC with HTTP/3 connection is fault tolerant on data packets, which ensures higher video processing speed and accuracy. Although QUICer is stable, has high-speed and supports package lossless connections, the underlying QUIC with HTTP/3 may not be an ideal solution. Taking QUICer L1 as an example: with a single QUICer connection, we could only get about 12X speed up compared to TCP and UDP protocols. However, if we introduce a multi-thread program implementation, we can easily get up to 140x speed-up with 8 threads, or even higher speed up with more threads. Since a multi-thread program solution is often limited by resources on the edge device, most of our tests have been performed by using at most 8 threads to collect the performance results presented in Figure 8.

To summarize, if device settings on the drone permit QUIC protocol selection, by utilizing our QUICer as a multi-threaded communication strategy, the multi-UAV video analytics can achieve notable speed-up on video transmission in drone to ground transmissions. However, if no parallel execution is considered for transmission or if there are other limitations of device settings on the drone, our approach provides other methods. Specifically, we can provide features of various candidate protocol combinations on video data transmission and compression codec strategies in specific cases e.g., in the case where we know how users will utilize the drone video analytics to make decisions when setting up their drone data processing applications.

7. Conclusion

In this work, we presented DroneCOCoNet, a framework to realize learning-based computation offloading and control networking strategies during drone video analytics. We detailed two general mathematical formulas for the drone computation offloading problem, one that adopts a heuristic algorithm, and another that uses a reinforcement learning algorithm. Our approach to formulate the edge computation offloading problem as a Markov Decision Problem is both general and powerful compared to supervised learning methods. We demonstrated that reinforcement learning can be used in the DroneCOCoNet framework effectively, which outperforms common heuristic approaches as well as other approaches. We also designed and detailed our design of QUICer, a novel application to maximize data transmission using parallel execution within a given drone video analytics pipeline. Our evaluation using real-world experiments showed that our QUICer approach significantly increased the system performance compared to traditional network protocols such as TCP and UDP. We also obtained insights that indicate that our RL approach is an effective way to optimize edge computation offloading under constrained resources while supporting applications such as border security, disaster response, smart farming and smart city surveillance.

Our future work includes evaluating the DroneCOCoNet using simulators in addition to real experiments to introduce more parameters, generate more comprehensive data for various drone scenarios. Such work can help further evaluate both computation offloading and networking solution approaches as a global optimization problem in the orchestration of MEC resources. Our QUICer configurations can be replaced with TCP configurations dynamically when faced with low bitrate conditions that can be caused by e.g., drones flying over longer distance paths that cause significant degradation of signal conditions. On the evaluation side, we supposed that a real world experiment in a small testbed would further strengthen our validation experiments. However, our methods used in the simulation experiments are realistic and are intended to be used as part of large real-world testbeds that are challenging i.e., they are time consuming and expensive to build in practice and are also limited by current government regulations on drone flights for recreational or research purposes.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Award Number: CNS-1647182 and the Army Research Lab under Award Numbers: W911NF1820285 and W911NF1910181. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the Army Research Lab.

We sincerely thank Dr. Derek Anderson, Dr. Hadi Ali Akbarpour and Dr. Ekincan Ufuktepe for providing expert consultation to the team. In addition, we would also like to thank Alicia Esquivel Morel and Wenbin Guo for helping us testing and flying the drone fleets.

References

[1] Bekmezci, I., Sen, I., Erkalkan, E., 2015. Flying ad hoc networks (fanet) test bed implementation, in: 2015 7th International Conference on Recent Advances in Space Technologies (RAST), IEEE. pp. 665–668.

[2] Boroujerdian, B., Genc, H., Krishnan, S., Faust, A., Reddi, V.J., 2018. Why compute matters for uav energy efficiency? 2nd International Symposium on Aerial Robotics URL: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/162705e556829ec12b4e6e5261167bdf2de21d65.pdf.

[3] Calyam, P., Chandrasekaran, P., Trueb, G., Howes, N., Ramnath, R., Yu, D., Liu, Y., Xiong, L., Yang, D., 2012. Multi-resolution multimedia qoe models for iptv applications. Int. J. Digital Multimedia Broadcasting 2012, 904072:1–904072:13.

[4] Calyam, P., Haffner, M., Ekici, E., Lee, C.G., 2007. Measuring interaction qoe in internet videoconferencing, in: IEEE/IFIP Management of Multimedia and Mobile Networks and Services (MMNS), IEEE.

[5] Chemodanov, D., Qu, C., Opeoluwa, O., Wang, S., Calyam, P., 2019. Policy-based function-centric computation offloading for real-time drone video analytics, in: 2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), pp. 1–6. doi:10.1109/LANMAN.2019.8847112.

[6] Chinchali, S., Sharma, A., Harrison, J., Elhafsi, A., Kang, D., Pergament, E., Cidon, E., Katti, S., Pavone, M., 2019. Network offloading policies for cloud robotics: a learning-based approach. CoRR abs/1902.05703. URL: http://arxiv.org/abs/1902.05703, arXiv:1902.05703.

[7] Chriki, A., Touati, H., Snoussi, H., Kamoun, F., 2019. Fanet: Communication, mobility models and security issues. Computer Networks 163, 106877. URL: http://www.sciencedirect.com/science/article/pii/S1389128618309034, doi:https://doi.org/10.1016/j.comnet.2019.106877.

[8] Dab, B., Aitsaadi, N., Langar, R., 2019. Q-learning algorithm for joint computation offloading and resource allocation in edge cloud, in: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 45–52.

[9] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. ImageNet: A large-scale hierarchical image database, in: IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255.

[10] Developers, G., . The job shop problem | or-tools | google developers. URL: https://developers.google.com/optimization/scheduling,AccessedAugust2019.

[11] Gatimu, K., Dhamodaran, A., Johnson, T.T., Lee, B., 2020. Experimental study of qoe improvements towards adaptive hd video streaming using flexible dual tcp-udp streaming protocol. Multimedia Systems 26, 479–493.

[12] Gatimu, K., et al., 2018. Experimental study of low-latency hd vod streaming using flexible dual tcp-udp streaming protocol, in: 2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC), pp. 1–6. doi:10.1109/CCNC.2018.8319234.

[13] Gosavi, A., 2009. Reinforcement learning: A tutorial survey and recent advances. INFORMS J. on Computing 21, 178–192. URL: https://doi.org/10.1287/ijoc.1080.0305, doi:10.1287/ijoc.1080.0305.

[14] Guillen-Perez, A., Cano, M.D., 2018. Flying ad hoc networks: A new domain for network communications. Sensors 18, 3571. doi:10.3390/s18103571.

[15] Hayat, S., Yanmaz, E., Muzaffar, R., 2016. Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. IEEE Communications Surveys Tutorials 18, 2624–2661. doi:10.1109/COMST.2016.2560343.

[16] Jiang, C., Cheng, X., Gao, H., Zhou, X., Wan, J., 2019. Toward computation offloading in edge computing: A survey. IEEE Access 7, 131543–131558. doi:10.1109/ACCESS.2019.2938660.

[17] Jung, W., Yim, J., Ko, Y., Singh, S., 2017. Acods: adaptive computation offloading for drone surveillance system, in: 2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), pp. 1–6. doi:10.1109/MedHocNet.2017.8001647.

[18] Kerrache, C.A., Barka, E., Lagraa, N., Lakas, A., 2017. Reputation-aware energy-efficient solution for fanet monitoring, in: 2017 10th IFIP Wireless and Mobile Networking Conference (WMNC), pp. 1–6. doi:10.1109/WMNC.2017.8248851.

[19] Li, S., Kim, J.G., Han, D.H., Lee, K.S., 2019. A survey of energy-efficient communication protocols with QoS guarantees in wireless multimedia sensor networks. Sensors 19, 199. URL: https://doi.org/10.3390/s19010199, doi:10.3390/s19010199.

[20] Lukezic, A., Vojir, T., Cehovin Zajc, L., Matas, J., Kristan, M., 2017. Discriminative correlation filter with channel and spatial reliability, in: IEEE Conference on Computer Vision and Pattern Recognition, pp. 6309–6318.

[21] Messous, M., Sedjelmaci, H., Houari, N., Senouci, S., 2017. Computation offloading game for an uav network in mobile edge computing, in: 2017 IEEE International Conference on Communications (ICC), pp. 1–6. doi:`10.1109/ICC.2017.7996483`.

[22] Motlagh, N.H., Bagaa, M., Taleb, T., 2017. Uav-based iot platform: A crowd surveillance use case. IEEE Communications Magazine 55, 128–134. doi:`10.1109/MCOM.2017.1600587CM`.

[23] Motlagh, N.H., Taleb, T., Arouk, O., 2016. Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives. IEEE Internet of Things Journal 3, 899–922.

[24] Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E., 2018. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. Networks doi:`10.1002/net.21818`.

[25] Park, J., Choi, S., Hussen, H.R., Kim, J., 2017. Analysis of dynamic cluster head selection for mission-oriented flying ad hoc network, in: 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 21–23. doi:`10.1109/ICUFN.2017.7993740`.

[26] Pedregosa, F., et al., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[27] Pham, Q., et al., 2019. A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art. CoRR abs/1906.08452. URL: `http://arxiv.org/abs/1906.08452`, `arXiv:1906.08452`.

[28] Qu, C., Morel, A.E., Dahlquist, D., Calyam, P., 2020. Dronenet-sim: A learning-based trace simulation framework for control networking in drone video analytics, in: Proceedings of the 6th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications, Association for Computing Machinery, New York, NY, USA. URL: `https://doi.org/10.1145/3396864.3399705`, doi:`10.1145/3396864.3399705`.

[29] Qu, C., Opeoluwa, O., Gao, K., et al., . Dronecoconet project source code. URL: `https://github.com/CaesarQu/DroneCOCoNet`.

[30] Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You Only Look Once: Unified, real-time object detection, in: IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788.

[31] Seufert, M., et al., 2019. Quicker or not? -an empirical analysis of quic vs tcp for video streaming qoe provisioning, in: 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pp. 7–12. doi:`10.1109/ICIN.2019.8685913`.

[32] Shakarami, A., Ghobaei-Arani, M., Shahidinejad, A., 2020. A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. Computer Networks 182, 107496. doi:`10.1016/j.comnet.2020.107496`.

[33] Wang, Y., Ru, Z.Y., Wang, K., Huang, P.Q., 2020. Joint deployment and task scheduling optimization for large-scale mobile users in multi-uav-enabled mobile edge computing. IEEE Transactions on Cybernetics 50, 3984–3997. doi:`10.1109/TCYB.2019.2935466`.

[34] Xie, L., et al., 2004. Discovering meaningful multimedia patterns with audio-visual concepts and associated text, in: 2004 International Conference on Image Processing, 2004. ICIP '04., pp. 2383–2386 Vol. 4. doi:`10.1109/ICIP.2004.1421580`.

[35] Yang, T., et al., 2018. Deep reinforcement learning based resource allocation in low latency edge computing networks, in: 2018 15th International Symposium on Wireless Communication Systems (ISWCS), pp. 1–5. doi:`10.1109/ISWCS.2018.8491089`.

[36] Yu, J., Vandanapu, A., Qu, C., Wang, S., Calyam, P., 2020. Energy-aware dynamic computation offloading for video analytics in multi-uav systems. ICNC 2020 - International Conference on Computing, Networking and Communications .

[37] Zhu, P., Wen, L., Bian, X., Haibin, L., Hu, Q., 2018. Vision meets drones: A challenge. arXiv preprint arXiv:1804.07437 .