Learning-based Multi-Drone Network Edge Orchestration for Video Analytics

Chengyi Qu*, Rounak Singh†, Alicia Esquivel-Morel† and Prasad Calyam‡
* Department of Computer Science, Florida Gulf Coast University, USA.
†‡ Department of Electrical Engineering and Computer Science, University of Missouri - Columbia, USA.
Email: * cqu@fgcu.edu, †{rsft6, ace6qv}@mail.missouri.edu, ‡calyamp@missouri.edu

***Abstract*—Unmanned aerial vehicles (also known as drones) equipped with high-resolution video cameras have become increasingly popular for applications such as public safety and smart farming. However, inefficient configurations in drone video analytics due to misconfigured edge networks can lead to degraded video quality and inefficient resource utilization. In this paper, we propose a novel scheme for network edge orchestration that utilizes both offline and online learning-based approaches to achieve pertinent selections of network protocols and video properties in multi-drone-based video analytics. Our approach utilizes both supervised and unsupervised machine learning algorithms to make decisions regarding network protocols and video properties during the pre-takeoff stage of the drones (i.e., offline stage). Additionally, our approach incorporates a reinforcement learning-based multi-agent deep Q-network algorithm for drone trajectory optimization during flights (i.e., online stage) and a memory-to-memory multi-hop data forwarding strategy for drone swarm video transmission. Our evaluation results demonstrate that our offline orchestration approach can suitably choose network protocols (i.e., among TCP/HTTP, UDP/RTP, QUIC), while our unsupervised learning approach outperforms existing methods and achieves efficient offloading while improving network performance (i.e., throughput and round-trip time) by at least 25%, with satisfactory video quality. Furthermore, we demonstrate through trace-based and real-field experiment testbeds how our online orchestration in terms of decision-making and data forwarding strategies achieves 91% of the oracle baseline network throughput performance with comparable video quality. Overall, our approach offers a promising solution for optimizing drone video analytics and enhancing the overall performance of drone-swarm-based applications.**

***Index Terms*—Multi-access edge computing, Multi-drone networks, Reinforcement learning, Network protocols, Video analytics, Network characterization, Machine Learning**

I. Introduction

There is a rapid evolution in systems of unmanned aerial vehicles (a.k.a. drones) with edge-server architectures fueled by innovations in multi-access edge computing that are vital for applications in e.g., public safety and smart farming [1]. Most drone platforms can be equipped with high-resolution video cameras that can help visualize and monitor target status via object recognition, motion detection or tracking. Thus, it is essential to provide capabilities for video processing through edge network orchestration for application setups with multiple drones and edge resources [2].

However, issues related to multi-drone video analytics using edge computing and network control are understudied. Based on literature surveys [3], [4], prior works only address primitive mechanisms to orchestrate selection of network protocols and video properties in multi-drone video analytics. Further, there are significant challenges due to different features in multi-drone control such as mobility models, limitations in edge computation and communication resources [5]. Inefficient parameter selection for video processing and edge network misconfigurations can result in video impairments, reduced resolution of transmitted videos, and loss of points-of-interest in multi-drone video analytics applications.

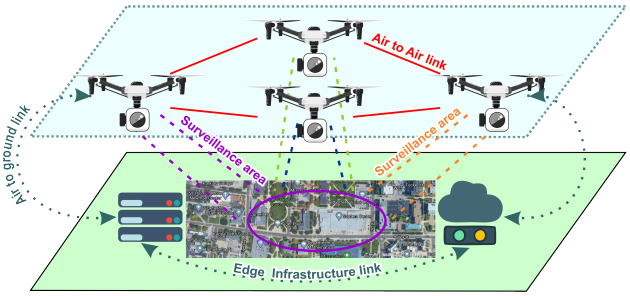

Fig. 1: Overview of multi-drone video analytics setup based on air-to-air and air-to-ground links with edge servers.

To better understand the requirements of multi-drone video analytics with edge network orchestration, let us consider the system setup shown in Figure 1. The setup features a drone-edge network with air-to-air and air-to-ground wireless links that utilize an edge server infrastructure on the ground to enable applications to monitor a surveillance area. System operation requires drones to cooperatively work with each other when recording surveillance area scene videos at different angles. This will require selection of a network protocol (i.e., amongst TCP/HTTP, UDP/RTP and QUIC [6]) in the drone-to-drone, drone-to-edge server communications. Also, in such cases, if the connection among one of the drones to an edge server is interrupted, the video properties (i.e., video codecs and resolutions) need to be adapted. Specifically, adaptation will be required in both the drones' pre-takeoff stage (*offline*) or during drones' flight (*online*) to cope with any limitations in the multi-drone and edge server resources [7], while satisfying application user experience expectations.

In this paper, we present a novel scheme for offline/online learning-based multi-drone video analytics through edge network orchestration to achieve pertinent selection of both network protocols and video properties. Depending on the

drone flight context, the scheme provides either *offline* (i.e., supervised-learning-based or unsupervised-learning-based) or *online* (i.e., reinforcement-learning-based) orchestration with: (i) network protocol selection (i.e., amongst TCP/HTTP, UDP/RTP, QUIC) for various network conditions and drone mobility models, and (ii) video properties (i.e., codec, resolution) selection for video transmission on wireless drone/edge network links. More specifically, using various machine learning models, our approach can predict network conditions either in the pre-takeoff stage (*offline*) or during (*online*) application use for pertinent network protocol and video properties selection in drone-to-drone and drone-to-edge setups.

Summary of the novel contributions of our work is as follows: First, our learning-based multi-drone video analytics with edge network orchestration is based on network conditions analysis (considering metrics of throughput and RTT) and video quality analysis (considering metrics of Peak Signal-To-Noise Ratio i.e., PSNR and video impairment percentage) in various drone video analytics scenarios. The orchestration utilizes function-centric computing in the video analytics where we decouple the application video analytics pipeline into isolated computer vision functions that can be executed either on the drone(s) or on the edge server locations. Our network protocol involves handling network impairments affecting the switching between high resolution/low resolution video capture, or the change of video codec selection for delivering effective scene surveillance.

In addition, to ensure the robustness and practicality of our learning-based approach, we adopt a diverse range of system setup configurations, drawing from a trace-based simulator dataset [8] in our multi-drone video analytics performance studies. These datasets comprise of drone traces derived from various mobility models and application scenarios, sourced from both real-world drone experiments and simulations. Leveraging these traces, we employ supervised learning strategies offline to intelligently curate potential video capture and network protocol setups based on anticipated network characteristics and video quality requirements. Our trace-based performance evaluation experiments with the supervised-learning approach showcase how the delivery of video quality via optimized network control aligns with real-world measurements, demonstrating the accuracy of our machine learning models. Throughout the training phase of the supervised-learning offline approach, we utilize four distinct machine learning models: Kernel-Ridge Regression (KRR), SVR-RBF (Support Vector Regression with Radial Basis Function kernel), Gaussian-Process Regression (GPR), and Random Forest Regression (RFR). This strategic employment of machine learning techniques enables us to streamline our approach, effectively reducing the overhead associated with the use of numerous redundant schemes.

Further, we account for cases where the drone traces generated in real-world experiments may not help in pertinent selection of network protocol and video parameters due to e.g., drone operation limitations, out-of-date network infrastructure or camera settings, and drone regulation policy restrictions. In such cases, we propose the use of an unsupervised clustering algorithm to replace the supervised learning during the *offline* stage to generate more pertinent candidate solutions to enhance video quality and network performance. We specifically test three learning-based clustering algorithms i.e., Possiblistic C-Means (PCM) [9], Fuzzy C-Means (FCM) [10] and K-means [11] and analyze their performance for a number of network protocol and video property clusters with matching solutions.

Our learning-based multi-drone network edge orchestration holds potential for various domains and end-applications. Specifically, it can be leveraged for disaster response management, where efficient coordination of multiple drones is essential e.g., for rapid surveillance. Additionally, in smart agriculture applications, our approach can enhance productivity by integrating multi-sensor data from drones for precise monitoring and decision-making. Furthermore, in logistic management for last-mile delivery systems, our methodology can optimize route planning and package delivery through intelligent drone orchestration. These diverse applications underscore the versatility and practicality of our approach in addressing real-world challenges across different domains.

Lastly, it is evident that the *offline* period decision-making process is not sufficiently efficient for drone swarms to effectively navigate dynamically changing environmental barriers, meet QoS requirements, fit multi-drone communications, or respond to emergency situations. To address these challenges, we additionally propose an *online* orchestration solution that utilizes both an *online* multi-agent reinforcement deep Q-network algorithm to facilitate drone trajectory prediction and an *online* memory-to-memory multi-hop data forwarding scheme to ensure sustainable network communication and accurate video data transmission among multi-drone setups.

The remainder of the paper is organized as follows: Section II presents related work. In Section III, we present details of the multi-drone video analytics with edge network orchestration process and provide an overview of our solution approach along with related performance metrics. In Section IV, we present two *offline* learning-based methods for either supervised or unsupervised decision making related to the selection of video properties and network protocol configurations. Section V enhances the learning-based method by utilizing a reinforcement-learning based algorithm for *online* orchestration. In Section VI, we describe our experimental testbed, performance metrics and evaluation results. Section VII concludes the paper.

II. Related Work

Drone-edge networks represent a strategic solution to the communication complexities inherent in multi-drone environments. At the heart of this architecture lies the edge server, a pivotal component also known as an edge computing server, meticulously engineered to revolutionize content and service delivery for end-users. Situated at the periphery of the network, proximal to user access points, the edge server functions as a localized hub for processing and storage, effectively

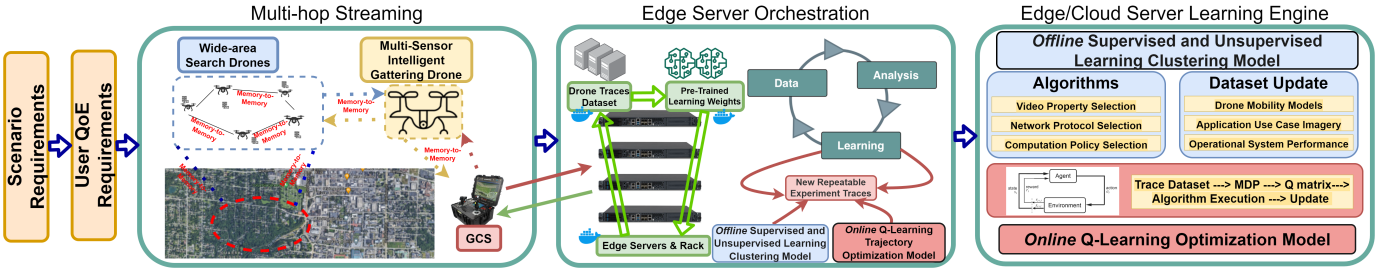

Fig. 2: Multi-drone Multi-hop Video Streaming and Analytics with edge network orchestration process diagram showing how application user requirements are considered in a trace dataset collection based on *offline* and *online* edge server orchestration.

mitigating latency and conserving bandwidth resources [12]. This departure from conventional centralized servers, typically installed within remote data centers, underscores the strategic deployment of edge servers at or near the network's edge. This proximity not only facilitates accelerated response times but also elevates overall user experiences. Recent research, as exemplified in [13], [14], has delved into the pivotal challenge of automating the orchestration of intelligent edge devices through sophisticated edge and cloud microservice platforms. Such endeavors underscore the transformative potential of edge computing in reshaping network dynamics and enhancing operational efficiency. However, their work lacks the implementation of networking aspects in terms of end-to-end orchestration. Specifically, there is a need to address particular problems concerning selection of pertinent network protocols that can help to optimize specific smart edge device parameters for high-resolution video delivery [3], [4].

We also leverage the multi-access edge computing paradigm in our drone-edge setups, where we optimize the user experience in terms of video quality in the multi-drone video analytics related application scenarios. Uniquely, our work uses learning-based strategies network edge orchestration by considering both network protocol and video properties. We employ machine learning techniques primarily classified into three categories: supervised, unsupervised, and reinforcement learning. Supervised learning utilizes labeled data samples to establish relationships between input and output spaces, while unsupervised learning operates on unlabeled data for pattern discovery. Our research focuses on employing supervised learning-based methods to intelligently configure network protocols and video capture setups in multi-drone video analytics applications. Similarly, other studies, such as [15], have utilized supervised learning to classify video traffic types, distinguishing between real-time streaming and on-demand videos. Moreover, [16] introduces a novel unsupervised K-means clustering algorithm capable of automatically determining the optimal number of clusters, without requiring initialization or parameter selection. Additionally, [17], [18] explored end-to-end machine learning frameworks, demonstrating their effectiveness in enhancing network quality of service. In alignment with these methodologies employed in prior works to similar problems, our research applies unsupervised learning clustering algorithms to aid users in multi-drone video analytics in selecting appropriate video properties and network protocols.

Machine learning can also be applied for the trajectory design and power control of multi-drone assisted wireless networks. Authors in [19] provide an approach to study trajectory design and analyze network configurations using Reinforcement Learning (RL) and a echo-state network. Their work focused on the pre-deployment of drones by using user location information from social media data in their solution. Other works such as [20] [21] focus on formulation of the trajectory as a Markov decision process (MDP). RL approaches can also include several other applications that focus on wireless power transfer between drones (in the air) and energy receivers (on the ground) to design the sub-optimal trajectories with lower complexities, compared with conventional power transfer systems [22]. Authors in [23] apply a deep Q-network (DQN) for optimization of drone systems navigation. The drones learn based on the received signal strength information for navigation with the aid of Q-learning. Our approach builds on this prior work in [23] and our novelty is in the use of network signal strength along with video codec information for drones to make dynamic decisions to stay in the optimal trajectory that helps in transmission of high-resolution video to meet user expectations.

As more than one drone generate data at rates on the order of tens of megabytes per second [24], real-time analysis of streaming data has emerged as a solution to cope with this new paradigm [25]–[28]. Furthermore, rapid analysis of generated data may permit real-time feedback and experiment steering. However, this often requires computational capabilities greater than those available at a single experimental facility—or may require the use of specialized computer systems [29]. To overcome the communication barriers on long distance multi-hop drone video data transmission, there is a need to adopt a memory-to-memory data forwarding strategy [30]. Our solution on transport layer data forwarding strategy is based on the ideas on the Science DMZ [31] which applied on scientific equipment and HPC (e.g., supercomputer, cloud computing resources) links.

III. Edge Orchestration Solution Overview

In this section, we provide a background on the process involved in the multi-drone video analytics with edge network orchestration in terms of offline and online algorithms. Figure 2 illustrates the multi-drone video analytics with edge

network orchestration process steps. We fist collect the traces from various drone scenarios, and provide user QoE requirements categories on the collections. After that, we provide two categories of orchestration models on either pre-tekeoff stage and during flights, i.e., offline and online model.

A. Trace Data Collection and Dataset Components

In this paper, we utilize two main resources for data collection: the VisDrone 2019 dataset [8] and a trace-based simulator, viz., DyCOCo [32]. The VisDrone dataset provides video/image analytics application-related data, consisting of a diverse set of videos captured by various types of drones. We integrate this dataset into our trace-based simulator DyCOCo to simulate drone flight traces based on the VisDrone dataset output. Within our simulator, we can simulate various aspects of drone operation, including platform control, camera/video control, and navigation control. Platform control is particularly crucial as it governs changes in the position, pose, and height of the multi-drone configuration, as well as adjustments to transport/application layer protocols for drone-to-drone transmission and data encryption. The real-world traces involve a multi-drone configuration of 8 drones embedded with high-definition video cameras that cooperate together along with an edge server network for video capture across a surveillance scene. The control of search and intelligence drones involves data points with various drone positions, speeds and camera angles. In addition, data points related to drone video analytics and drone-edge network performance are in the dataset.

In scenarios involving multiple drones and edge servers, accurately estimating network conditions is crucial for making informed orchestration decisions during drone flights. However, this can be a challenging task. Therefore, it is essential to select appropriate network protocols and video properties at the outset of each drone flight to ensure that application requirements are met. Table I provides an overview of the application requirement parameters used in a multi-drone setup, which are organized into two broad categories: (i) Analytics Layer, and (ii) System Layer. To train our machine learning models, we have compiled a database of 400 real-world traces collected from DyCOCo experiments and VisDrone. The traces used for training include three types of data: (i) *preliminary application data*, which consists of the transport protocols, application protocols, and video properties used in each flight; (ii) *real-time/live application data*, which includes captured network packets, video streams, and flight traces; and (iii) *post-application data*, which comprises measurements of the overall network status and video quality during the flight.

In the post-application data, video quality scores are rated subjectively by users and mapped to one of the three categories: high, medium, or low. To determine the standard for assigning a high score, we use the parameter provided by the user in the preliminary application data. If the average user scores are higher than this value, we classify the traces as a 'high case'. If the average scores match the user-provided value, we classify the traces as a 'medium case'. Similarly, if the average scores are lower than expected, we classify the traces as a 'low case'. This approach enables us to evaluate the performance of each trace objectively and accurately and helps us to use this information to train our machine-learning models effectively.

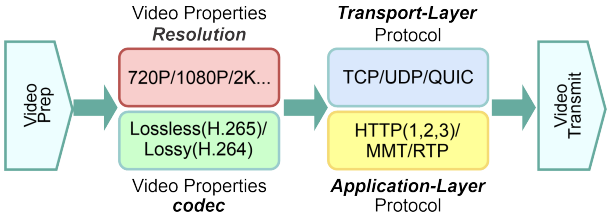

Fig. 3: Network protocol and video properties selection process applied after the video preparation from the drone camera, and before the transmission process from drone to ground.

Figure 3 shows the available options of network protocols and video properties that can be chosen in our approach. We label the video properties and network protocols shown in Figure 3 as our desired output values.

B. Edge Server Orchestration

As shown in Figure 2, our edge server orchestration for drone control is responsible for processing videos from the multi-drone deployments. Edge server(s) are tasked to observe the impact of the initial video codec selection based on the quality of the video captured. The observation is to verify whether video quality meets application user requirements i.e., if the high-quality video delivery is occurring regardless of the mobility model of the multi-drone configuration and any impairments resulting from network limitations. Thus, edge servers analyze the video properties and if necessary, adapt the future video properties' settings as well as network protocol configuration strategies. To make the multi-drone video analytics flexible, scalable and reusable, edge server orchestration utilizes function-centric computing, where the decoupling of the video analytics pipeline is performed into isolated computer vision functions that are packaged as Docker containers [33]. Pre-trained learning weights for the edge server orchestration are derived from the drone traces dataset detailed above in Section III-A, and are stored in the Docker containers. A ground operator can use the edge server to either choose our pre-trained learning weights on the existing trace dataset or utilize the learning-based algorithm itself to train and generate more accurate network protocol and video property configurations relevant to the application user requirements.

Our proposed learning-based *offline* and *online* algorithms are implemented in the edge server orchestration as part of a learning engine that serves as a key orchestration component. More specifically, as shown in Figure 2, the learning engine is composed of: (i) *offline* supervised learning and unsupervised learning clustering models that use our algorithms and updated datasets, and (ii) *online* DQN trajectory optimization model. Both algorithms leverage the trace-based simulator dataset in the learning engine, and information related to drone mobility models, application imagery and operational system perfor-

TABLE I: Descriptions of parameters used for client benchmark analysis.

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	[0 Mbps - 50 Mbps), [50 Mbps - 100 Mbps), [100 Mbps - 500 Mbps), [500 Mbps - 2 Gbps), [2 Gbps+)
	P5	On-Flight CPU Level	1-5	[0 MHz - 75 MHz), [75 MHz - 250 MHz), [250 MHz - 750 MHz), [750 MHz - 1.5 GHz), [1.5 GHz+]

mance are utitlized in the context of multi-drone video analytics. Larger trace data fosters accurate decisions to improve the edge server orchestration performance, and also guide configurations of upcoming experiments. In order to facilitate communication on A2G link, we also introduced memory-to-memory data forwarding for enhancing the performance of multi-hop streaming.

IV. *Offline* Learning-based Video Properties and Network Protocols Selection

In this section, we detail our *offline* supervised learning and unsupervised learning algorithms for multi-drone network edge orchestration.

A. Supervised Learning Algorithm

Both training phase and testing phase are designed and described in the following. In general, we use 70% of the dataset for training and the rest 30% of the dataset is used for testing, with 10-fold validation method.

1) Training Phase: The training phase is a step that is used to narrow down the searchable space of traces by filtering our 400 real-world traces database. Although the settings of each of the traces can be quite varied, they have redundant data that impacts the learning process. Consequently, it is possible to find multiple network protocol configurations that satisfy the needs of the application (e.g., traces with multiple video resolution settings (720p and 480p) will use the same protocol settings (e.g., TCP) under ideal network conditions). Hence, drone operators and experimenters can obtain more number of choice suggestions with the same application requirements.

With the collected data sets, we used machine learning to predict the network protocol and video properties used in the 'high case' in the post-application measurements. We have nearly 85% of data in the dataset categorized as a 'high case'. Those traces vary by application/transport protocol settings and video resolution/codec selections. We use the supervised learning approach to achieve prediction and classification. In the training phase, four machine learning models from the Sci-Kit Learn toolkit [34] 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). In essence, machine learning based categorization allows us to reduce the overhead of relying on the use of hundreds of redundant schemes. We are able to select the most optimal choice in terms of network protocol and video properties, and configure them as preliminary application settings for the drone flight paths. In terms of the training size of the dataset, we used 70% of the data for training. Additionally, we applied ten-fold cross-validation to ensure more robust and reliable training results. By doing so, we ensure that the models are not only evaluated on a single train-test split but are also validated across multiple subsets of the data, providing a more comprehensive measure of their performance. The results from the testing phase are then compared to determine which algorithm delivers the best predictive accuracy and generalization capability for our specific dataset.

2) Testing Phase: To test the accuracy of the machine learning model predictions, we use a 95% confidence interval for correctly categorized data. Additionally, to evaluate overall video quality, we employ PSNR as a performance metric, as detailed in Section VI-B. The evaluation phase results are presented in Section VI-C, where we discuss overall training accuracy and other metrics, such as prediction time.

In terms of the hyperparameters of four supervised learning algorithms—Kernel-Ridge Regression (KRR), SVR-RBF (Radial Basis Function kernel SVM), Gaussian-Process Regression (GPR), and Random Forest Regression (RFR), we utlize the defaulty values on Sci-Kit Learn toolkit to apply and evaluate the preformance of the datasets. In KRR, key hyperparameters include the regularization parameter (α) with a default value of 1.0, and the kernel parameters, such as the bandwidth (δ) for the RBF kernel, which is often set to 1.0 by default. For SVR-RBF, critical hyperparameters are the regularization parameter (C) with a default of 1.0, the kernel coefficient (γ) which defaults to 'scale' ($1/(n\_features * X.var())$) and the epsilon (ϵ) with a default value of 0.1. GPR involves hyperparameters such as the length-scale (l) with a default value of 1.0, the noise level often set to 1e-10, and the choice of kernel function, with the RBF kernel as a common default. Lastly, RFR relies on hyperparameters like the number of trees in the forest ($n\_estimators$) with a default of 100, the maximum depth of each tree (max_depth) which is typically set to None, allowing trees to expand until all leaves are pure or contain less than the minimum samples required to

split, and the minimum number of samples required to split a node ($min\_samples\_split$) with a default of 2. Tuning these hyperparameters effectively is crucial for optimizing the performance of each regression model.

3) Discussion: Although we have labeled data for learning as categories, the category of the dataset may not be the best choice overall for satisfactory application user experience. Here are two examples which can be considered as a risk in using our supervised learning algorithm in practice:

Example 1: Some drone management systems may only support UDP, which may not be the most suitable solution.

Example 2: During the processing, the protocol could be changed in the application to improve the application performance. Thus, a multi-drone and edge server system many not have a fixed configuration during an experiment. Our supervised learning output only provides consistent network protocol and video properties choices which may not suitable for some advanced situations where dynamic decisions could provide better performance.

To overcome both questions, we take the following actions: (i) we unlabel the dataset in terms of protocol selection and video properties choices, only focusing on the video quality and the streaming data in order to do clustering on those datasets, and (ii) we do not determine a precise network protocol/video property choice for each dataset i.e., each element is assigned to all of the available clusters with a different membership degree for each cluster; once the system setup requires a dual protocol selection or a drone video resolution switching strategy, we transition the decision making in the application to an *online* procedure described in Section V.

B. Unsupervised Learning Algorithm

Given that we unlabeled the trace data on network protocols and video properties, we utilize the unsupervised learning algorithm to aid the decision making in the orchestration process. The orchestration needs to analyze the unlabeled data to find the optimal choice of the network protocol or video properties for a given multi-drone-edge-server application context. Clustering algorithm provides either fuzzy or precise bounds to categorize the unlabeled data into different categories that map to the potential network protocol and video properties. As part of the solution approach, we introduce three different unsupervised learning algorithms, FCM, PCM and K-means, which belong to two categories on clustering: either fuzzy or not. FCM is one of the most widely used fuzzy clustering algorithms expressed as shown in Equation 1.

$$FCM(U,X,V) = \sum_{i=1}^{n}\sum_{j=1}^{m} u_{ij}^{k} \cdot \|x_j - v_i\|, 1 < k < \infty \quad (1)$$

where $X = x_1, ..., x_m$ and $V = v_1, ..., v_n$ are the feature data and cluster centroids; $U = u_{ij}^{k}$ is a fuzzy partition matrix composed of the membership degree of pattern x_k for a given cluster i; m is the total number of patterns in a given data set and n is the number of clusters; k is a factor which defines the fuzziness degree of the partition. In such a FCM algorithm, the main constraint is that the sum of each column in membership matrix U is equal to 1. Although FCM is not effective in finding complex cluster shapes other than the spherical shape, we still consider FCM as one of the comparison methods to relatively detect the noise from clustering. Our orchestration can benefit from FCM on an unbalanced dataset such as e.g., the one relating to network protocols. The data imbalance arises from the fact that we generate more TCP and UDP based data rather than data on QUIC in our collected dataset.

FCM algorithm produces the memberships of the data points that are related to the distance of that data point from the centers of the clusters. Thus, if a data point is equidistant from the clusters, then it will have the same membership value in each cluster. In order to prevent such outliers from being accounted in, another clustering technique was introduced by Krishnapuram and Keller, named PCM. In contrast to FCM algorithm, membership value generated by PCM algorithm can be interpreted as "degree of belongings or compatibility or typicality". Typicality degrees are defined to build prototypes that characterize data subcategories Typicality values with respect to one cluster do not depend on any of the prototypes of other clusters. Equation 2 shows the relationships -

$$PCM(U,X,V) = \sum_{i=1}^{n}\sum_{j=1}^{m} u_{ij}^{k} d_{ij}^{2} + \sum_{i=1}^{n} \mu_i \sum_{j=1}^{m} (1-u_{ij})^{k} \quad (2)$$

where k is the factor that defines the degree of the partition, and d_{ij}^{2} is the point distances. PCM is different from FCM because of the μ variable addition, which is called the "scale" parameter that is estimated from the data to prevent outliers.

Algorithm 1: Unsupervised Learning Algorithm

Input: $P\{1-5\}, videoCategory, protocolCategory, videoSource, X = x_1 - x_m, V = v_1 - v_m, U, D$
Result: $cluster\_choice$

```
if videoCategory || protocolCategory then
    preCluster ← K-MEANS(P{1 − 5});
end if
else
    Fuzzy ←
      min{FCM{X, V}, PCM{U, D}} ←avg[P{1 − 5}];
    preCluster ← FUZZY(P{1 − 5});
end if
foreach preCluster do
    while videoSource do
        PSNR ← videoSource;
    end while
    throughput[] ← avg{videoSource} ;
end foreach
cluster_choice ←
  min{max{throughput[]}, max{videoQuality[]}}
```

Algorithm 1 provides the logic of our unsupervised learning clustering algorithm selection. The logic of our unsupervised clustering algorithm selection is as follows: Initially, all three unsupervised learning algorithms are considered. If user provides the number of the clusters, the K-means algorithm will be used for obtaining results. If the cluster number is not restricted, we will use recursion utilizing either PCM or FCM to select the optimal number of clusters based on the network

performance and video quality metrics. By evaluating the network performance (average throughput) and video metrics (minimum PSNR), we choose the lower limit of the number of clusters among all outputs. In addition, the strategy to select fuzzy or not is given by the category requirements detail. If either of the category is given, a classic k-means will be used for clustering. If the bound on the category is fuzzy, we will choose either the FCM or PCM, depending on which ever has the smaller amount of cluster numbers.

To conclude, we can observe that both the supervised and unsupervised learning algorithms provide *offline* results which only provide guidance in the pre-takeoff stage on multi-drone video analytics with edge network orchestration. In the case where we deal with intermittent network failures or other environment limitations, we transition the decision making in the application to an *online* procedure described in Section V.

V. *Online* Learning-based Drone Trajectory Optimization and Video Data Forwarding

In this section, we first describe the motivations for our *online* orchestration. Following this, we describe the reinforcement learning-based drone trajectory optimization algorithm, and then we present our memory-to-memory data forwarding strategy among drone-swarm setups on video transmission purposes.

A. Orchestration Motivation for Online Learning

Unsupervised learning could receive cluster combination options *offline* i.e., in the pre-takeoff stage of drones. In many drone swarm applications [35], [36] it may be necessary to adjust network protocols or video properties during mid-flight operation to improve user experience. This requires analyzing drone trajectories and video quality to determine the most appropriate network protocols and video codecs. To address this challenge, we propose a multi-agent Deep Q-Network (DQN) algorithm for intelligent path planning of drones. By analyzing trajectories and video quality, our algorithm can select the most appropriate network protocols and video codecs to ensure that the user experience expectations are met. Overall, our proposed approach can help to optimize drone swarm operations and improve user satisfaction.

Moreover, when it comes to drone swarm communication, it requires a high-speed sustainable data forwarding link to achieve user expectations. To enable the use of various transport protocols in real-world drone swarm applications, a highly efficient communication data forwarding strategy [37], [38] is necessary to extend the A2G (air-to-ground) communications. Unlike prior works that focus only on L2 approaches [39]–[42], we propose a memory-to-memory data forwarding approach that focuses on L4 transport protocols that can be used in multi-hop communications to facilitate information exchange between nodes that are not directly connected. In this approach, an intermediary node (relay) is used to forward data between two long-distance nodes which cannot communicate directly, or overcome various communication barriers such as distance, interference, or node mobility. By routing data through multiple intermediate nodes, the memory-to-memory data forwarding approach can improve the reliability and efficiency of communication in multi-hop networks, making it a useful approach for drone swarm applications.

B. Reinforcement learning based drone trajectory optimization

1) Multi-Agent Deep Q-Network: To achieve intelligent trajectory learning, we propose a multi-agent Deep Q-Network [43], [44] based method which aims to establish an optimal policy for drones' path selection as per changes in network performance and video quality. The path selection aids the drones to learn and make necessary sequence of decisions under uncertainty in drone-edge network conditions. The learning involved in path selection by the drones can be represented as a Markov Decision Process [45] (MDP) which forms the basis for the DQN algorithm.

A similar approach to formulate an MDP for intelligent trajectory design for drones is shown in [46]. Figure 2 shows (see bottom portion related to $online$ DQN) the basic steps for formulating an MDP, and the subsequent use of DQN (detailed in following sub-section) to obtain the drone trajectory update guidance.

The multi-agent DQN uses Boltzmann's Q policy [47] by considering a continuous state space that allows the drones to explore and exploit [48] the learning environment to the largest extent possible along with a sequential memory called replay buffer to store the state-action pairs along with rewards for reinforcement learning based simulations denoted by $(s^o, a^u, r, s^{o'})$. The output is the drone trajectory update guidance that is used to keep the drones as much as possible in their optimal trajectories. The intelligent trajectory learning detailed in the following sub-section renders network performance in terms of throughput and the video quality scores (i.e., rewards) obtained in the learning process.

2) Intelligent Trajectory Learning: The agents are operating in the environment are denoted by $D_{(n)}$, where n represents the number of drones, and the time intervals in which they carry out surveillance are considered to be irregular. We refer to each of these time steps as episodes. During each episode, the agents hover over specific regions in discrete time steps denoted by Δt and observe a global state given by $s_t^o = (S_0, S_1, S_2, S_3)$ that represents *'Random Area', 'Secure Area', 'Precarious Area'* and *'Terminal State'*, respectively and perform independent actions $a_t^u = A_0, A_1$ and A_2 that represent *'Hover'*, *'Move rapidly'*, and *'Land to recharge'*, respectively and receive global reward r_{net}. Let C be the cost associated with the network protocol and video properties selection, with i being the task of switching the video codec along with resolution i.e., H.265 HEVC and H.264 AVC between 720p, 1080p and 2K. The immediate costs of actions $a_{t(i)}^u$ are given by -

$$\psi(s_t^o, a_{t(i)}^u) = \begin{cases} C_{switch}(s_t^o, a_{t(i)}^u), if a_{t(i)}^u \geq 1 \\ C_{stable}(s_t^o, a_{t(i)}^u), if a_{t(i)}^u = 0 \end{cases} \quad (3)$$

Where C_{switch} represents the cost of switching resolution of the video stream due to increased traffic in a particular state, and C_{stable} is the cost related to the change in network performance in the stable state. The total long term cost is the expected sum of all the components' immediate costs, given by -

$$\psi^{\pi}(s) = \sum_{i \in \xi} \psi(s_t^o, a_{t(i)}^u) \tag{4}$$

Let d be the distance that the drones travel in the secure area S_1, λ is the network wavelength and β is the bandwidth of the network and frequency lies in the range (2.4 GHz to 5.8 GHz). It is assumed that the best network conditions are available when drones hover in S_1, and transmit high resolution video. As the drones keep taking actions to reach the secure state, there is a possibility of a large number of drones accumulating in the same space creating traffic and β is over-utilized. This results in frame stalling, distortion and blurring of the video. To effectively use β, the agent has to remain in the S_1 and simultaneously continue to configure the network protocol. The reward associated with the reliable connection establishment after entering a new state at cost C_{stable} is given by -

$$R(s_t^o, a_t^u)_{network} = \frac{\psi^{\pi}(s^o)}{\lambda} log_{10}(\beta) \tag{5}$$

The reward associated with the agent hovering over the secure state that allows for highest quality video resolution at cost C_{switch} is given by -

$$R(s_t^o, a_t^u)_{video} = \alpha \left[1 - [\tfrac{d}{d_{max}}]^{0.4}\right] . \psi^{\pi}(s) \tag{6}$$

where α is the security parameter associated with S_1. The global reward function of a state-action pair of an episode is given as the sum of the two intermediate rewards.

$$r_{net} = R(s_t^o, a_t^u)_{video} + R(s_t^o, a_t^u)_{network} \tag{7}$$

The trajectory learning of the drones occurs by maximizing the gain G_t along their path which is a function of expected cumulative discounted rewards.

$$G_t = E[\sum_{n=0}^{\infty} \gamma^n r_{net}(s_t^o, a_t^u)] \tag{8}$$

where γ is the discount factor $(0 \leq \gamma \leq 1)$. Each action change may only produce a small reward. Thus, we require the value of the discount factor γ to be such that it maximizes the cumulative reward. The multi-agent DQN uses an estimator neural network (parameterized by θ) and a target neural network (parameterized by $\theta^{'}$) along with the replay memory to approximate the action-value function. The DQN is trained using the loss function-

$$L(\theta) = E[r_{net} + \gamma max Q(s_t^{o'}, a_t^{u'}; \theta^{'}) - Q(s_t^o, a_t^u); \theta] \tag{9}$$

The estimator and target neural networks have pre-defined weights. The estimator takes state space values (s_t^o) of a drone as input, and generates action value function $Q(s_t^o, a_t^u)$. The target network's weights are updated at specific predefined intervals so that they match with weights of the estimator network to produce maximum value of the action-value function and add stability to the performance. While back-propagating, the weights of both networks are updated in an iterative fashion and the output values come close to the optimal value. All experiences $(s^o, a^u, r, s^{o'})$ are stored in the replay buffer and are sampled uniformly as training examples. This process makes sure that the there is no correlation between the training examples which may lead the policy to reach a local minima, and is followed until optimal Q function $Q_{\pi}^*(s_t^o, a_t^u)$ is obtained. Once the optimal value is reached, our DQN algorithm converges. The optimal Q function is given as -

$$Q_{\pi}^*(s_t^o, a_t^u) = E[\sum_k \gamma^k (r_{net} | s^o, a^u)] \tag{10}$$

From our empirical observations, our proposed DQN best converges at $\gamma = 0.8$. The optimal policy which maps the state-space and actions, $\pi_t^* : S_t \rightarrow A_t$ is given as -

$$\pi_t^* = argmin_{\pi} \sum_{i \in \xi} \psi^{\pi}(s_t^o, a_{t(i)}^u) \tag{11}$$

The optimal policy governs the convergence of the multi-agent DQN algorithm and leads the agents (drones) in independent intelligent path, orchestrating network and video analytics during their flight operations. We remark that the online decision making delay can be ignored compared to the whole multi-drone mission period in applications because the actual delay value depends on the learning period and the convergence efficiency.

The steps for the multi-agent DQN based trajectory learning are given in Algorithm 2. We further extend DQN to Double DQN and Dueling DQN using the same custom learning environment. The Double DQN (DDQN) decouples the task of selection and estimation of actions by using two estimator neural networks. Thus, if there are any overestimations, they are removed by decoupling the action value function into two separate functions. Furthermore, the Dueling DQN has a single Q-Network architecture but instead of having a single stream of convolutional layers, the Dueling DQN has a structure involving two sequences of fully connected layers [49]. Due to the dueling architecture of the DQN, the state values and action values are separated into two streams which share a common neural network for feature extraction. Both these streams get combined to produce a Q-function estimate.

C. Memory-to-memory Data Forwarding for Multi-hop Streaming

In order to facilitate end-to-end communication on a multi-hop A2G link, we propose three memory-to-memory data forwarding approaches for each corresponding transport protocol. In this experiment setup, note that the 'end-to-end' refers to the complete process of communication from the sender to the receiver, encompassing all intermediate steps and components involved in transmitting and receiving the message. For example, the receiver uses offline datasets to build a predictive model based on past observations or historical trends. Once the initial setup is complete, online communication begins. This involves the continuous exchange of data between the sender

Algorithm 2: Multi-agent DQN Algorithm with Policy to Achieve Optimal Trajectory

$D_{(n)}$ = Agents, where $n \neq 0$
$r_t = r_{net}$
Output: policy π_t^*
initialize replay memory, action-value function $Q(s_t^o, a_t^u)$ and weights of estimator network and target network.
Let $\Delta t = t, Q_t(s^o, a^u) = 0$ for all s and a;
while $s_t^o = S_3$ **do**
  $Q(s_t^o, a_t^u) = (s_t^o, a_t^u)$;
  $s_t^o = S_0(or)S_1(or)S_2$;
  Let $D(1)_t, D(2)_t, D(3)_t, ...D(n)_t$ perform a_t^u independently ;
  receive r_{net};
  store the experience $< (s_t^o, a_t^u, r_{net}, s_t^{o\prime}) >$
  sample random transitions $< s_t, a_t, r_t, s_t^{'} >$,
  calculate loss for each transition $L(\theta) = E[r_{net} + \gamma max Q(s_t^{o^{'}}, a_t^{u^{'}}; \theta^{'}) - Q(s_t^o, a_t^u); \theta]$;
  Update weights of estimator and target network;
  if *Positive* r_{net} **then**
    $a_t = A_0$;
    Update Policy $\pi_t : s_t \rightarrow a_t$;
    update weights;
  else
    keep training;
  end if
end while
$a_t = A_2$;
update $Q(s_t^0, a_t^u) to Q_\pi^*(s_t^o, a_t^u)$;
update policy π_t^*;

and receiver in real-time. More specifically, when it comes to multi-hop communication, there is a risk of increased latency and decreased reliability due to the involvement of multiple intermediate nodes. To address this issue, we implement a memory-to-memory communication approach inspired by the SciStream project [50]. Unlike the SciStream approach that only focuses on high-capacity scientific data streaming between scientific instruments and high-performance computing (HPC), our approach establishes bridging and end-to-end authentication between only on drone swarm-based A2A and A2G links, which aims to enable efficient drone video data streaming for drone swarm video analytics applications. By using memory-to-memory data streaming, we can avoid useless input/output operations in a multi-hop relay mode and improve the reliability as well as the efficiency of communication in multi-hop networks. Due to the various implementations of each transport protocol, we propose our novel memory-to-memory data forwarding approach with three separate approaches for each transport protocol: TCP, UDP, and QUIC.

1) TCP Data Forwarding: We first introduce the TCP data forwarding method between drones. In this port-forwarding strategy, a Packet Forwarding Drone (PFD) listens on a local

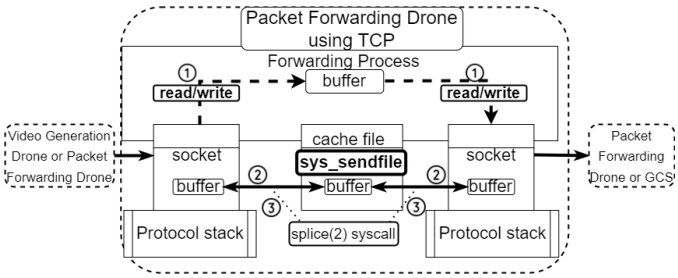

Fig. 4: Data forwarding drone utilizing TCP-based socket with either: option-1) buffer, option-2) *sendfile* function on Linux core, and option-3) *splice* function on Linux core

TCP port and forwards all incoming data to a remote host. This remote host can either be another PFD or a consumer application. The core functionality of this operation relies on traditional socket programming functions such as $listen$, $bind$, and $send$. To effectively manage and process large volumes of data, our approach employs user space buffers for data queuing, as illustrated in Figure 4 option-1. However, despite its potential to accommodate various scientific streaming applications, a notable drawback of this method is the introduction of significant jitter. The increased jitter can be primarily attributed to the fluctuation in context switch and data copy latencies between the kernel and user space. Unfortunately, this jitter can have adverse effects on applications that require near real-time streaming capabilities.

Within the context of data center networking, encountering jitter during the transmission of high-capacity data is deemed acceptable. However, the dynamics change when dealing with drone flights, as they often venture beyond signal ranges. Here, achieving consistent and uninterrupted end-to-end data transmission takes precedence. In our pursuit of mitigating jitter, we embarked on a comprehensive exploration of kernel operations and functions to minimize context switches. It is important to note that our investigation is based on the assumption that all embedded systems on the drones are Linux-compatible. In Linux system, we identified two specialized syscalls namely, $sendfile$ and $splice(2)$, each designed to tackle this challenge. The $sendfile$ syscall, originally designed for rapid file transfers from disk to socket, has the potential to be harnessed in creating a proxy equipped with a cache file positioned between sockets (refer to Figure 4, option-2). However, this approach retains the necessity for copy operations between sockets, exerting additional strain on the proxy as the load from streaming applications intensifies. This often results in persistent buffering within the user space, consequently leading to repetitive copying from core to user space. Alternatively, the Linux $splice$ function (depicted in Figure 4, option-3) circumvents the need for data copying from kernel to user space. Nonetheless, this approach incurs the cost of multiple syscalls for each forwarded packet, which can escalate operational overhead. Implementing this method as a proxy, especially within a confined operating system environment, is intricately tied to the specific operating system version

and settings. In our pursuit of a solution, we embraced Docker containers to encapsulate dependencies, kernel versions, and the proxy service itself. It is clear that each approach has its own set of limitations. For instance, option-1 is universally applicable across drones regardless of their operating systems, yet it introduces the formidable challenge of jitter, a critical hurdle in drone communication. On the other hand, both options 2 and 3 effectively address the jitter concern, but their efficacy hinges greatly on system configurations and versions, particularly in the case of option-3. In summation, the choice among these approaches is contingent upon the system version compatibility and the administrative privilege acquisition. If these conditions are met, option-3 emerges as the optimal preference for TCP packet forwarding.

2) UDP Data Forwarding: To enhance the jitter performance of scientific streaming applications during UDP video transmission, we present an alternative design for a UDP proxy. This design aims to reduce context switching between the kernel and user space for individual Packet Forwarding Drones (PFDs), potentially leading to improved performance. It is noteworthy to highlight that UDP has taken a prominent role in contemporary drone video transmission methodologies, as evidenced by recent studies [51]. Similar to the preceding TCP data forwarding strategy, we operate under the assumption that the core of the transmission system facilitates modifications to the Linux core code. Our proposed solution involves leveraging the capabilities of Linux eBPF (Berkeley Packet Filter) technology [52] to create a UDP proxy that exclusively processes all packet operations within the kernel space. This approach utilizes eBPF's ability to run isolated programs in the kernel space, eliminating the need for modifications to the kernel's source code.

Modifying the core code of eBPF is necessary to establish the logic for forwarding UDP packets. In the conventional socket programming model embedded within the Linux core logic, a socket establishes a communication link between the kernel IP stack and a task context in the user space. This setup facilitates the transfer of data from the kernel space to tasks associated with the bound socket. To streamline this process, the introduction of *sockmap* [53] by eBPF proves instrumental. This entails utilizing a technique known as the *Stream Parser* to route packets to other eBPF tasks. These tasks execute streaming redirection operations exclusively within the kernel space, leading to a marked acceleration in data transmission compared to the original socket programming.

In Figure 5, the fundamental logic underlying the application of *sockmap* within a Packet Forwarding Drone (PFD) is depicted. In contrast to the previous practice of directly loading processes into the user space or interacting with a cache file, the UDP socket descriptors within a Global Network (GN) are now integrated into a $SOCKMAP$, specifically the $BPF\_MAP\_TYPE\_SOCKMAP$, which is a type of eBPF map structure. Following this insertion, the forwarding procedure is managed by eBPF, ensuring that all processing occurs exclusively within the kernel space, thus eliminating the involvement of user space.

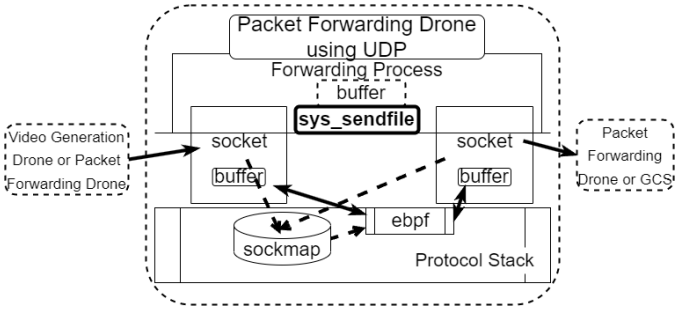

Fig. 5: Data forwarding process utilizing sockmap on eBPF where UDP is used as a transport protocol.

3) Implementation of a QUIC Proxy for Secure Streaming: QUIC [6] is a new transport protocol standardized by the IETF and developed on top of UDP with the aim of being encrypted by default. This protocol lays the foundation for HTTP/3 [54], another IETF-developed application-layer protocol operating atop QUIC. Enabling QUIC proxying can be achieved through two distinct methods, namely L4 and L7 approaches [55]. L7 proxy solutions, exemplified by MASQUE [56], extend the capabilities of HTTP CONNECT methods to facilitate the utilization of HTTP/3 as a tunneling protocol between clients and proxies. The MASQUE working group within the IETF has developed HTTP CAPSULE methods [57], which enable the adoption of HTTP/3 as a tunneling mechanism between two nodes. The emergence of MASQUE has introduced a novel communication pattern, wherein consumer applications can solicit producer application services from proxy nodes. This paradigm, reminiscent of a VPN, enables MASQUE to obscure the consumer application's address from the producer application. In adherence to the HTTP CAPSULE framework, the destination drone initially seeks the destination location in the HTTP header, containing IP addresses and ports. Subsequently, each PFD consults the proxy table to map the intended destination, or forwards the request to the subsequent PFD if necessary. However, this methodology proves unsuitable for multi-drone communication, as the connection establishment does not necessitate mutual recognition between the destination drone and the proxy drone. Furthermore, incorporating MASQUE into the role of packet forwarding drones for QUIC proxying runs counter to the original design, given the MASQUE's L7 nature. Such an adaptation may squander resources on the communication link, undermining efficiency.

To this end, in this subsection we present our QUIC proxy solution that remains at the transport layer (L4) using the QUIC stream mode. Our QUIC proxy implementation establishes two distinct QUIC connections: one for proxy binding and listening, and the other for connecting to the designated target. Within our QUIC proxy setup, data streaming over the initial connection undergoes buffering at the system level within the proxy node itself. This buffered data is subsequently forwarded to the preassigned target through the secondary connection. However, it's essential to note that QUIC transfers across high-bandwidth links may face limitations imposed by

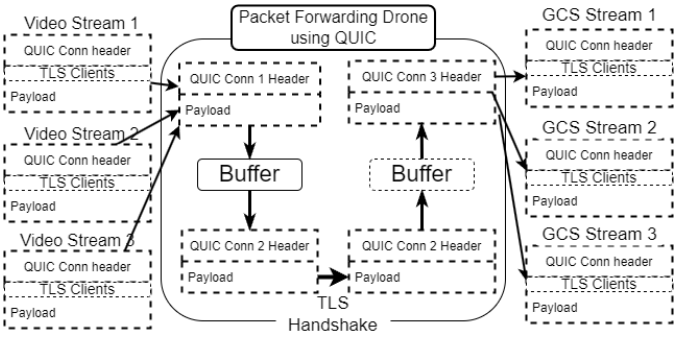

Fig. 6: Data forwarding process utilizing QUIC proxy with default encryption settings. The 'TLS clients' segment can be defined as with encryption or without by each drone instead of corresponding application nodes.

the default UDP receive buffer size. This buffer retains packets received by the kernel but not yet accessed by the application. For applications that demand substantial throughput, an inadequately sized buffer may result in considerable packet loss or even connection disruptions. To mitigate this challenge, it becomes imperative to adjust the UDP buffer size within the operating system before deploying the QUIC proxy. Figure 6 shows the packet forwarding data flow of our QUIC proxy implementation.

In conclusion, when it comes to using TCP, UDP, and QUIC in a drone swarm scenario, we have the appropriate packet forwarding solution on L4 for each protocol. This approach is necessary when multi-hop end-to-end communication is required due to either the need for extended communication range or the when there is blockage of direct communication.

VI. Performance Evaluation

In this section, we present the performance evaluation of various machine learning models in the orchestration of network protocols and video parmeters selection in multi-drone video analytics. We first describe our experiment setup and data collection. Following this, we detail results for our supervised, unsupervised and reinforcement learning based models. Finally, we present a discussion of the salient findings.

A. Experiment Setup

For evaluation of our edge network orchestration algorithms, we use data collection from a hierarchical drone configuration (detailed in Section III) that is controlled within a drone-edge network with varying considerations such as: different mobility models and number of drones in a fleet. We specifically use the Gauss-Markov Mobility Model to simulate the drone behavior in multi-drone configurations as detailed in [58]. The Gauss-Markov Mobility model can easily and inherently eliminate abrupt stops and sharp turns by allowing past velocities/directions to influence future velocities/directions [59].

To streamline our experimentation, we have segregated the data into two distinct sets: (i) drone video data and (ii) drone trace data. In the realm of drone video data, a comprehensive compilation of 60 video clips has been amassed. For real-world experimentation, we procured 30 video clips derived from three distinct drone models—DJI Phantom 3, Mavic 3, and the Parrot Drone. These clips were meticulously collected in adherence to the Gauss-Markov Mobility Model. The transmission of these video clips transpired via IEEE 802.11 networks to the edge server. Additionally, we factored in drone video data hailing from our trace-based simulation dataset, encompassing an additional 30 video clips. Within this dataset, we deliberately selected video clips captured concurrently in identical locations, albeit subjected to diverse mobility models, including Gauss-Markov, Random, and Mission-based.

TABLE II: Environment Settings used in Experiments

Application Settings		Network Settings	
Number of drones:	10-30	App. Bit rate:	6 Mbps
Flight area:	10-15 miles	Tx power:	32-48 dBm
Transmission range:	50-250 m	Tx/Rx gain:	3 dB
Simulation time:	1000-3000 s	WIFI protocol	802.11 n/ac
Avg. drone speed:	10 - 35 mph	Modulation:	OFDM
Prop. Model:	TWO RAY	Data rate:	65 Mbps

From the drone video and drone trace datasets, we gathered the original video property settings in terms of video codec type, and video resolution into our learning-based database. Traces are formatted and stored in the form of CSV files. Within these traces, we collect data on the network protocols, i.e., TCP, UDP, and QUIC. Our dataset features three different video resolutions (1344x756, 1902x1071, 2688x1322). For each video resolution, we have 20 video clips each with 50 seconds duration and having a frame rate of 30 frames/second. In terms of video codecs, we have videos with both H.265 and H.264 codecs. We labeled the data with the video properties and network protocols as the output, and the rest of the settings as the input. The video captured by drone swarms contains various scenes, i.e., metropolitan area, urban area, city park, and crop area [8]. In the video analytics, we process each live stream through an image processing pipeline that comprises of a pre-trained model designed for pedestrian detection running on Tensorflow, and an object detection function to detect number of pedestrians in each frame.

For the network performance evaluation, we created a computing environment with a desktop serving an edge server. Nvidia Jetson Nano was used as an embedded drone device to process the network performance and video properties. Other related application and network settings on the trace-based simulation can be found in Table II. Regarding the choice of routing protocol, in our previous research [60], we explored the application of various cutting-edge routing protocols within the same drone-edge computation topology. To streamline our experiments, we will continue to use the routing protocol proposed in [60], which is a learning-based location-aided routing protocol.

In the context of multi-drone communication scenarios, regulatory constraints on real-world drone operations led us to employ a wired experimental testbed for simulating multi-drone interactions. Our approach leverages the Terabit network

supercore and a coast-to-coast 100 Gbps core provided in a FABRIC testbed [61] we have used. The presented diagram in Figure 7 offers a visualization of our experimental setup, orchestrated using FABRIC's GUI. Our methodology entails sourcing FABRIC resources from two distinct sites—Salt Lake City (SALT) and Utah (UTAH)—and interconnecting them through a WAN link and a dedicated control network. At each site, a minimum of two compute nodes are acquired, and they are interlinked through LAN connections, while the two nodes establish WAN connectivity. Furthermore, an out-of-band network, referred to as `control_net`, facilitates the execution of our control protocol. Importantly, the third compute node at the Utah site is designated for executing proxy scripts, functioning as a relay drone. The chosen operating system is Linux Ubuntu 20.04, paired with Linux source code version 5.4.0-97 to enable eBPF support. Data emulation is managed using Python 3.8, with ZeroMQ enabling the implementation of a Python-based Pub/Sub streaming pipeline. To address the network congestion issue, we adopt the HTCP [62] as the congestion control mechanism, meticulously tuning TCP settings to optimize performance for WAN scenarios. Our QUIC proxy implementation rests on GoLang version 1.18.3, complemented by quic-go version 0.28.0.

B. Network Performance and Video Quality Measurement

Unstable network quality or intermittent outages can frequently occur in the drone-edge network, which could significantly influence the drone video analytics tasks at the application level (due to the impact on computation offloading), and at the drone guidance level (due to impact on operational commands from the edge server/ground control station). To characterize the network performance in a drone-edge network, we use a network recovery time that estimates the network quality before we make *offline* decisions on video transmission parameters or operational command control settings. In addition, throughput and round-trip-time (RTT) are also considered as evaluation metrics.

To measure the video quality, we consider both objective and subjective metrics. First, for the objective metric, we use the Peak Signal-To-Noise Ratio (PSNR, expressed in decibels (dB)) to measure the quality of the video streamed by the drones, as shown in Equation 13. We remark that PSNR is a widely used metric to estimate the quality of images that undergo degradation when being transmitted on a communication link. PSNR is estimated by using mean square error (MSE), and the image quality at the destination of a communication link can be estimated as follows:

$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N} \tag{12}$$

where M, N define the size of the image, and m, n are pixels.

$$Threshold = PSNR = 20log_{10}\frac{V_{Peak}}{MSE} \tag{13}$$

Here, we establish that effective rectification of video impairments hinges on the Peak Signal-to-Noise Ratio (PSNR) falling below or equal to a predefined threshold ($\leq$30%). Within this threshold, the strategy involves directing the drone to stream higher-resolution video through the controller, effectively mitigating the impairments. Conversely, if the PSNR exceeds the threshold ($>$30%), the controller engages nearby drones within the drone-edge network to capture the requisite high-quality footage. This iterative process continues until the desired high-resolution video quality is achieved, forming an integral part of the video farming procedure.

Note that the PNSR values in the range of 15 dB to 25 dB are considered to be at a minimally acceptable level and indicate poor network connectivity between the sender and receiver of the video streams. PSNR values above 25 dB up to 40 dB are deemed favorable for a satisfactory user experience, with values surpassing 32 dB considered optimal for meeting most users' expectations regarding video quality. Therefore, we posit that transmitted video quality can be effectively rectified through camera control when the PSNR falls below or equals a fixed threshold, such as $\leq$30 dB. Within this range, the drone controller can mitigate video impairments by instructing the drone to stream higher-resolution video. Conversely, if the PSNR surpasses the threshold, the controller orchestrates nearby drones to capture high-quality video within the drone-edge network. This iterative process ensures the attainment of the desired high-resolution video quality, integral to the seamless operation of the drone-edge network for video streaming.

The unsupervised learning algorithm we employ incorporates the Mean Opinion Score (MOS) as a key subjective metric for assessing video quality, a widely accepted measure in subjective evaluations involving human subjects. MOS serves to rank the perceptual quality experienced by end-users on a scale from 1 to 5. Within this scale, the range [1, 3) denotes a "Poor" grade, indicating that end-users perceive severe and frequent impairments that render the application unusable. In contrast, the range [3, 4) signifies an "Acceptable" grade, where end-users encounter intermittent impairments but find the application mostly usable. Finally, the range [4, 5] corresponds to a "Good" grade, indicating that end-users perceive minimal or no impairments, thus finding the application consistently usable. By leveraging MOS within these defined ranges, our algorithm effectively gauges the subjective quality of video content, facilitating informed decision-making in optimizing user experience.

TABLE III: The average prediction time taken by various Supervised Learning models ($\pm$ RMSE) based on trace based dataset in terms of network protocol and video properties prediction.

Model Type	Protocol Prediction	Video Properties Prediction
KRR	0.650±0.00181	**0.0015±0.0035**
SVR-RBF	**0.043±0.0081**	0.034±0.00161
GPR	1.080±0.0005	0.836±0.0003
RFR	0.125±0.0370	0.096±0.0200

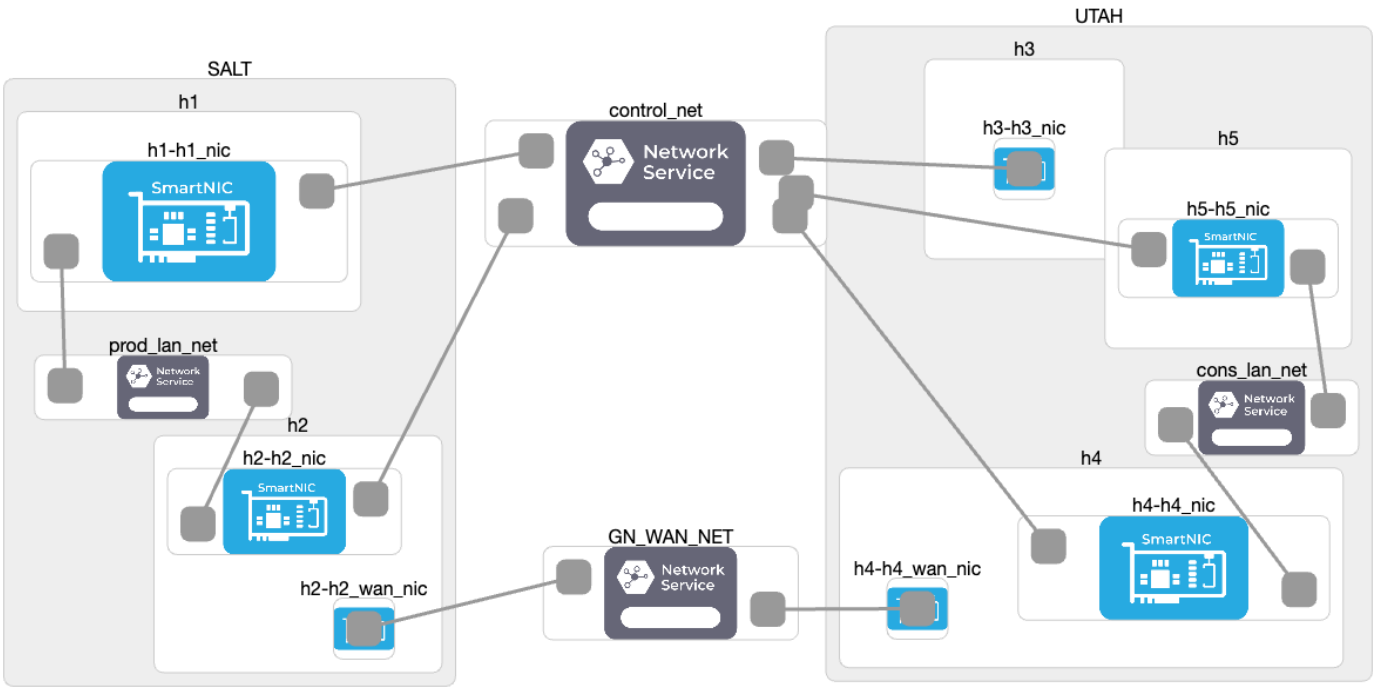

Fig. 7: Example the logical topology of one of our experimental setups (our LAN scenario) as drawn by FABRIC's GUI.

C. *Offline* Supervised Learning Model Evaluation

We selected four different machine learning models, i.e., KRR, SVR-RBF, GPR, and RFR for both the test phase and training phase in our supervised learning method. In these models, initially, we were primarily concerned about the prediction accuracy on networking protocols and video properties. However, all machine learning models achieved similar high accuracy on prediction (0.95 ± 0.036) in both the network protocol and video properties selection cases. Hence, we selected the models based on the shortest training time with relatively smaller RMSE (Root Mean Square Error), as indicated in Table III. For the network protocol prediction, we found that the model that had the shortest training time was the SVR-RBF model. For the video properties prediction, the best performance was seen in the KRR model.

We compared our learning-based trace simulation results with previous policy-based estimation and pure NS-3 simulation traces [63]. The scheme in [63] uses a simple policy-based decision-making algorithm to determine network protocol and video properties. As shown in Table IV, without the supervised machine learning prediction, the accuracy of network protocol and video property selection is relatively low and uncertain. Even for the PSNR testing after reproducing the traces, a policy-based estimation can only achieve a PSNR at around 30 dB, which is is not ideal for user experience expectations of the video quality.

We evaluated the performance of our supervised machine learning approach, against the 300 diverse traces selected from our database as part of the testing set. In each of the traces, the number of objects and their locations are distinct from the traces in the training set. To test the system's ability to adapt to various network bandwidth constraints, we evaluated each trace with 5 network condition settings, ranging from 50 Mbps to 2 Gbps. Table IV provides the supervised machine learning model results on network 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 conclude that: (a) 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; and (b) KRR can generate more reliable results with a stable range of PSNR, although the performance is lower than other machine learning models. As describe in the offline

TABLE IV: Comparison of prediction among policy-based estimate (baseline) and the four supervised machine learning models. PSNR results shown are the $[min, max]$ values among all the experiments with the video trace data.

Model Type	Protocol 95% CI	Video Property 95% CI	PSNR
Policy-based [63]	(0.267, 0.6577)	(0.1904,0.2715)	[26.71, 33.59]
KRR	(0.652, 0.928)	(0.803,0.927)	[32.86, 35.93]
SVR-RBF	(0.779, 0.833)	(0.9034,0.952)	[30.59, 33.93]
GPR	(0.813,0.882)	(0.7523,0.8)	[29.26, 32.86]
RFR	(0.9124,0.96)	(0.9032,0.97)	[22.26, 29.37]

D. *Offline* Unsupervised Learning Model Evaluation

Since both, FCM and PCM are two C-means-based clustering algorithms and with similar clustering logic instead of K-means algorithm, we first evaluated the performance between these two algorithms. Three C-means related evaluation metrics are considered to evaluate the accuracy of PCM and FCM. These metrics are: Dunn Index (D), Davies-Bouldin Index (DB), and Partition Coefficient Index (PC) on Coefficient

TABLE V: Unsupervised C-Means clustering algorithm accuracy comparison

	Algo.	D	DB	PC
Video	FCM	5.89102e-05	1.5634	0.7444
	PCM	1.1012e-04	2.0972e-03	1.3677
Network	FCM	0.005	0.7195	0.7982
	PCM	7.725e-03	0.513	1.5053

Index (PC). DB index considers the dispersion and separation of all clusters, the Dunn Index only considers the worst cases in the clustering i.e., it considers the clusters that are closest together and the single most dispersed cluster. In addition, partition coefficient index only uses the membership matrix to compute the index based on the table and clustering results.

According to the definition, the lower DB, higher PC, and higher Dunn index indicate a better cluster. As we can observe from the Table V, PCM outperforms FCM on both network protocol and video properties categories. Thus, we prefer to choose PCM as our C-means clustering algorithm to compare with the K-means clustering algorithm. The reason why the performance of FCM is lower than PCM could be attributed to the fact that: (i) its noise points or the outliers are also accounted in the membership values, and (ii) it detects spherical clusters effectively, but is not effective in finding other cluster shapes.

To summarize, in the comparison of C-Means and K-Means cluster, we will only choose PCM as our C-Means solution. For network performance evaluation, we use both throughput and round-trip time (RTT) as the metrics, and for the video properties evaluation, we use the PSNR objective metric as well as the MOS subjective metric. For the subjective assessment, we recruited 10 human subjects and asked the participants to provide their MOS rankings on a 1 (Poor) - 5 (Excellent) scale to assess their experience quality with 95% confidence intervals. We have the following observations on both network performance and video quality sides:

1) PCM improves overall network performance on network protocol selection: Figure 8 shows how PCM clustering improves overall network performance in comparison with K-Means approach. By selecting PCM as the clustering strategy, categories which are assigned to use TCP, UDP, and QUIC as the transport protocol on video data transmission could achieve at least 25% of improvement, as well as at least 18% of reduction in delay in terms of RTT. In contrast to PCM clustering, K-Means clustering prediction results in poor overall bandwidth and delay (up to 50% reduction in throughput and RTT).

The reason why the K-Means clustering algorithm failed to choose the proper network protocol is that: (i) for large dataset and multi-dimensional inputs, K-Means can easily be trapped into a local minimum, even in a long term of training, which will in turn categorize the traces into the wrong cluster; and (ii) the dataset labels in terms of network protocols are unbalanced e.g., we only obtained limited amount of drone traces from QUIC protocol compared with TCP and UDP. The unbalanced dataset will influence the K-means centroid and result in an abandonment of the small cluster.

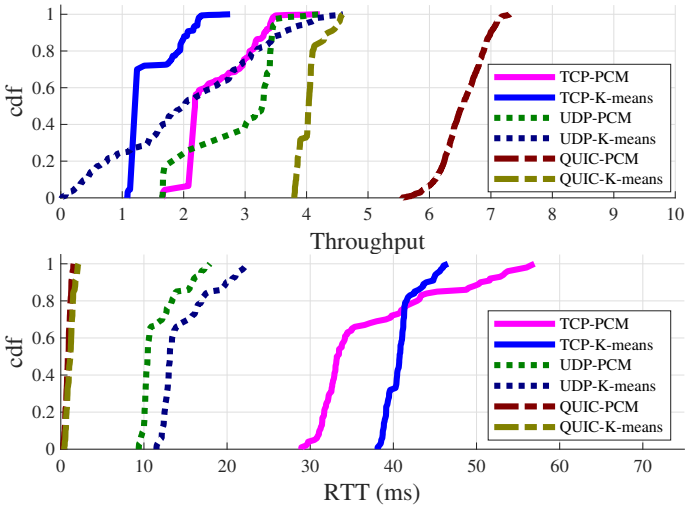

Fig. 8: Cumulative distribution function (cdf) of: network throughput (Mbps), and RTT (ms) - for the PCM and K-means clustering algorithms; each group is categorized according to TCP, UDP and QUIC protocol.

2) PCM improves overall video quality performance after transmission: Table VI shows both objective (i.e., PSNR) and subjective (MOS) measurements related to the performance of PCM and K-Means algorithms for clustering the video properties. $Q1$ to $Q5$ in the table represent 5 survey questions that were used to obtain related MOS rankings from participants: *video quality, video smoothness, no blur effects, no frame freezes,* and *no tiling effects*. The baseline represents the original video captured from the camera on the drone. Since there is no transmission process, the PSNR does not apply to the baseline videos. We can observe that the PCM can achieve acceptable video quality comparable to the MOS rankings on the original video, which is in contrast to K-Means clustering with low MOS rankings for all 5 survey questions. Also, the PSNR results show a similar difference as well. Overall, PCM clustering can improve both overall network performance and video quality by making near-optimized decisions.

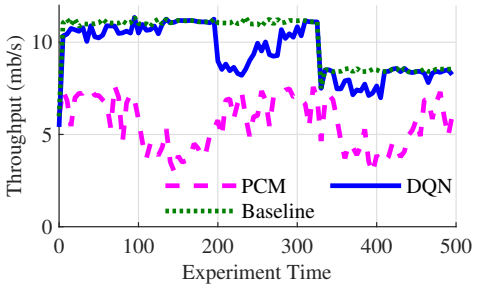

Fig. 9: Throughput performance comparison of PCM, DQN (with LR = 0.001) and Baseline in a trace-based experiment.

*E. **Online** RL-based Model Evaluation*

Figure 9 illustrates the network performance difference between PCM and DQN as the learning procedure. In this experiment, we use the trace-based simulation dataset. The benefit of using simulation instead of the real-world experiment is that

TABLE VI: MOS measurement and PSNR results on K-Means, PCM and Baseline.

Approach	$Q1$	$Q2$	$Q3$	$Q4$	$Q5$	**Overall**	**PSNR**
K-Means	1.65±0.3	1.59±0.4	2.61±0.3	2.47±1.2	2.26±0.5	2.12±0.4	[27.35,33.90]
PCM	3.27±0.1	3.70±0.1	4.11±0.0	4.23±0.3	4.17±0.2	3.90±0.15	[32.32,39.00]
Baseline	4.64±0.3	4.89±0.1	4.73±0.	4.91±0.1	4.82±0.2	4.79±0.2	N/A

simulation could provide any potential combinations without limitations. Baseline data represents the oracle situation when the drone is aware of the network situation in advance, which is an unrealistic assumption in real-world operations.

As we can observe from Figure 9, it is clear that - with the DQN model, trace-based experiments can achieve better throughput performance than PCM comparable with the oracle baseline. Specifically, according to the results of our presented experiments, by utilizing DQN as the *online* orchestration algorithm, we can achieve at least 91% of throughput performance of the oracle baseline approach. In the same case, the PCM can only achieve around half of the throughput performance. The reason for the sudden drops in DQN throughput is that DQN model may provide a flawed prediction when the drone flies across area boundaries. However, DQN can recover from these negative reward conditions in a short time ($\sim$20s) and resume the high throughput. It is worth noting that *online* DQN learning exhibits similar results to *offline* unsupervised learning (i.e., PCM) in terms of video properties performance. This observation shows that - dynamically changing video properties according to the drone's trajectory does not increase the video quality after transmission. Another significant observation is that - we expected a better network performance that can improve the video quality after the transmission process, but we did not get the expected experimental results. This may be because of other factors i.e., the environment, noise, or delay caused by frequent video codec changes that influence the improvement of the video quality.

*F. **Online** Multi-hop Data Forwarding Evaluation*

Figures 10 and 11 illustrate the network performance in terms of goodput and delay for multi-hop data forwarding scenarios. In these experiments, multiple slices on FABRIC are used to simulate multiple drone nodes, which are essential for multi-hop data forwarding. The original throughput between nodes is limited to 100 Gbps. Figure 10 presents the goodput performance results for TCP, UDP, and QUIC proxies across four different video resolutions (720p, 1080p, 2K, and 4K). To fully utilize the 100 Gbps capacity, a maximum of three video streams are run simultaneously. Each video stream application requires at least two drones (simulation nodes) as relay nodes. It is evident that increasing the number of drones used as relay nodes leads to decreased streaming goodput. In the best-case scenario, utilizing a TCP proxy to stream a single 720p video with four relay nodes results in a maximum achievable throughput of approximately 82 Gbps. Among all video resolutions, no significant difference is observed between TCP and UDP proxies, attributed to the memory-to-memory packet forwarding design, which separates the user space and kernel space on the proxy node. However, since the QUIC connection requires default encryption on packets, it fails to achieve similar results to TCP and UDP proxies as the video streaming capacity increases. Combining the information from Figures 10 and 11, it is evident that TCP outperforms other protocols in terms of both goodput and communication delays due to its simpler kernel-only packet forwarding design. However, even though the QUIC proxy requires packet transmission in the user space, the results show that it can still achieve at least 80% of the performance in terms of goodput without compromising network delay.

G. Overall Results Discussion

In this section, we conclude how different learning approaches used in multi-drone video analytics with edge network orchestration have advantages and disadvantages. Based on the conclusions, we provide guidance to drone system operators on the potential learning-based approach choices that are suitable in terms of optimizing the selection of network protocols and video properties.

First, multi-drone applications users demand different network performance and video quality requirements. If the user requests are related to video quality experience, it is suitable to use supervised learning with trained weights for decision making of network protocols and video properties. Secondly, if no preferred category is needed or supervised learning outputs decision cannot achieve the goals, unsupervised learning algorithms can take the charge of clustering the drone setup environment into pre-trained categories. As shown from the experiments on our drone video data and drone trace data, our unsupervised learning approach could achieve at least $\approx$32 dB value (good video quality as perceived by users) for PSNR after transmission. Nonetheless, some advanced network protocols such as QUIC are not commonly used in commercial drones, which might limit the choices.

Thus, unsupervised learning is not the primary choice for simple use cases such as: small area surveillance, traffic management or drone-aided parcel delivery. Computation resources in terms of learning and inference can be embedded on the drone using edge devices such as Nvidia Jetson [64]. In such cases, reinforcement learning procedures can be applied to dynamically optimize the drone trajectories and make effective predictions to aid decisions for selection of network protocols and video properties.

In addition, when considering multi-hop communication that utilizes multiple drones as relay nodes, we can make the following observations: 1) Regardless of the transport protocol used (TCP, UDP, or QUIC), our memory-to-memory packet forwarding proxy can achieve comparable results in terms of overall goodput. 2) When taking communication delay into account, kernel-based forwarding approaches (such as TCP and UDP) tend to outperform user-space-based forwarding approaches (such as QUIC). 3) It is important to note that

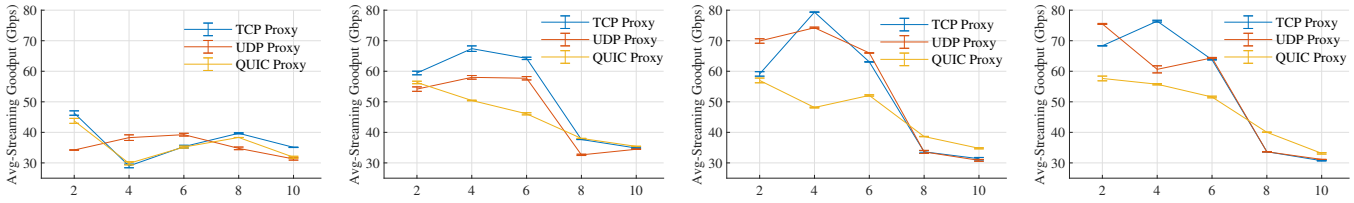

Fig. 10: Goodput performance of three proxy solutions (TCP, UDP, and QUIC) over various numbers of drones as relay nodes in terms of various video resolutions of a) 720p, b) 1080p, c) 2K and d) 4K as transmission sources.

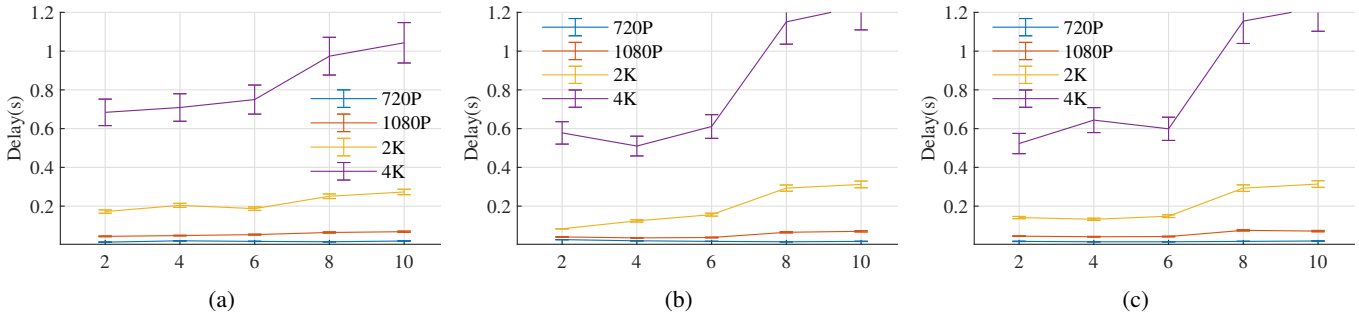

Fig. 11: Delay performance on various resolution video transmission under the various proxy types, i.e., a) TCP proxy, b) UDP proxy, and c) QUIC proxy.

video resolution only impacts the transmission delay and not the goodput itself. In the real-world experiments, we conclude that multi-hop communication could benefit the performance of video streaming and analytics, which in further enhance the orchestration process.

VII. Conclusion

In this paper, we presented a novel scheme for learning-based multi-drone video analytics with edge network orchestration that considers both network protocol and video property selection. Three different categories (i.e., supervised, unsupervised, and RL) of learning based algorithms were proposed and validated to facilitate decision making for pertinent selection of network protocol and video properties in both an *offline* manner (i.e., pre-takeoff stage of drones) and in an *online* manner (i.e., during drone(s) flight).

Through evaluation experiments, we showed that our proposed Possibilistic C-means (PCM) learning approach in the *offline* setting achieves efficient offloading, while also improving the network performance (i.e., throughput and round-trip time) for by least 25% compared with supervised approaches with acceptable video quality (i.e., PSNR $>$ 32). Also, based on experiments on a trace-based simulator dataset, we showed that DQN that utilizes deep reinforcement learning to predict trajectory in the *online* setting can allow for dynamic decision-making, achieving $\approx$91% of the oracle baseline network throughput performance with comparable video quality. This is as in the case seen in the unsupervised clustering algorithm's performance. These results show that our scheme can be used to handle significant challenges due to various features involved in multi-drone control such as mobility models, limitations in edge computation resources as well as multiple choices in communication strategies with trade-offs. Thus, our scheme is relevant for different multi-drone video analytics applications in e.g., disaster management, smart city traffic management, and precision agriculture.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Award Number: CNS-1647182. 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.

References

[1] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (fanets): A survey," *Elsevier Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[2] C. Rametta and G. Schembra, "Designing a softwarized network deployed on a fleet of drones for rural zone monitoring," *Future Internet*, vol. 9, p. 8, 2017.

[3] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[4] A. Hamm, A. Willner, and I. Schieferdecker, "Edge computing: A comprehensive survey of current initiatives and a roadmap for a sustainable edge computing development," 2019.

[5] M. Khan, I. Qureshi, and F. Khanzada, "A hybrid communication scheme for efficient and low-cost deployment of future flying ad-hoc network (fanet)," vol. 3, p. 22, 02 2019.

[6] M. Seufert, R. Schatz, N. Wehner, and P. Casas, "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)*, Feb 2019, pp. 7–12.

[7] A. Guillen-Perez and M.-D. Cano, "Flying ad hoc networks: A new domain for network communications," *Sensors*, vol. 18, no. 10, p. 3571, 2018.

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

[9] R. Krishnapuram and J. M. Keller, "The possibilistic c-means algorithm: insights and recommendations," *IEEE transactions on Fuzzy Systems*, vol. 4, no. 3, pp. 385–393, 1996.

[10] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers Geosciences*, vol. 10, no. 2, pp. 191–203, 1984. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0098300484900207

[11] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[12] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and ai for uavs: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 435–15 459, 2022.

[13] Fernandez, I. Vidal, and Valera, "Enabling the orchestration of iot slices through edge and cloud microservice platforms," *Sensors*, vol. 19, p. 2980, 07 2019.

[14] T. Cerquitelli, M. Meo, M. Curado, L. Skorin-Kapov, and E. E. Tsiropoulou, "Machine learning empowered computer networks," p. 109807, 2023.

[15] E. Grabs, E. Petersons, A. Ipatovs, and D. Chulkovs, "Supervised machine learning based classification of video traffic types," in *2020 24th International Conference Electronics*, 2020, pp. 1–4.

[16] K. P. Sinaga and M. Yang, "Unsupervised k-means clustering algorithm," *IEEE Access*, vol. 8, pp. 80 716–80 727, 2020.

[17] T. Zhang and S. Mao, "Machine learning for end-to-end congestion control," *IEEE Communications Magazine*, vol. 58, pp. 52–57, 06 2020.

[18] G. Zhu, J. Zan, Y. Yang, and X. Qi, "A supervised learning based qos assurance architecture for 5g networks," *IEEE Access*, vol. 7, pp. 43 598–43 606, 2019.

[19] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory design and power control for multi-uav assisted wireless networks: A machine learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7957–7969, 2019.

[20] S. Yin, S. Zhao, Y. Zhao, and F. R. Yu, "Intelligent trajectory design in uav-aided communications with reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8227–8231, 2019.

[21] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative internet of uavs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 6807–6821, 2020.

[22] S. Ku, S. Jung, and C. Lee, "Uav trajectory design based on reinforcement learning for wireless power transfer," in *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, 2019, pp. 1–3.

[23] H. Huang, Y. Yang, H. Wang, Z. Ding, H. Sari, and F. Adachi, "Deep reinforcement learning for uav navigation through massive mimo technique," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1117–1121, 2020.

[24] DOE-SC, "The Report of the BES Advisory Subcommittee on Future X-ray Light Sources," https://science.osti.gov/-/media/bes/besac/pdf/Reports/Future_Light_Sources_report_BESAC_approved_72513.pdf, 2013, accessed: 2021-09-06.

[25] NSF's Big Ideas, "Harnessing data for 21st century science and engineering," https://www.nsf.gov/news/special_reports/big_ideas/harnessing.jsp, 2017, Accessed: 2020-01-06.

[26] Advanced Scientific Computing Research and Basic Energy Sciences, U.S. Department of Energy, "Exascale Requirements Review," https://www.osti.gov/servlets/purl/1341721, 2015, Accessed: 2020-01-06.

[27] A. L. Kinney and D. M. Tilbury, "Dear Colleague Letter: Data-Driven Discovery Science in Chemistry (D3SC)," https://www.nsf.gov/pubs/2018/nsf18075/nsf18075.pdf, 2018, Accessed: 2021-03-01.

[28] NSF, "Transforming science through cyberinfrastructure: NSF's blueprint for a national cyberinfrastructure ecosystem for science and engineering in the 21st century," https://www.nsf.gov/cise/oac/vision/blueprint-2019/nsf-aci-blueprint-v10-508.pdf, 2018, Accessed: 2020-03-01.

[29] R. Kettimuthu, Z. Liu, D. Wheeler, I. Foster, K. Heitmann, and F. Cappello, "Transferring a petabyte in a day," *4th International Workshop on Innovating the Network for Data Intensive Science (INDIS) 2017*, pp. 1–11, Nov. 2017.

[30] J. Chung, W. Zacherek, A. Wisniewski, Z. Liu, T. Bicer, R. Kettimuthu, and I. Foster, "Scistream: Architecture and toolkit for data streaming between federated science instruments," in *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 185–198. [Online]. Available: https://doi.org/10.1145/3502181.3531475

[31] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The Science DMZ: A network design pattern for data-intensive science," *Scientific Programming*, vol. 22, no. 2, pp. 173–185, 2014.

[32] C. Qu, S. Wang, and P. Calyam, "Dycoco: A dynamic computation offloading and control framework for drone video analytics," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, 2019, pp. 1–2.

[33] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[34] "scikit-learn 0.21.2 documentation." [Online]. Available: https://scikit-learn.org/stable, Accessed April 2021.

[35] H. Hildmann and E. Kovacs, "Using unmanned aerial vehicles (uavs) as mobile sensing platforms (msps) for disaster response, civil security and public safety," *Drones*, vol. 3, no. 3, p. 59, 2019.

[36] "Delivery drones: Understanding the challenges for drone delivery," accessed on Dec 2021. [Online]. Available: https://dronerush.com/google-ups-delivery-drones-19058/

[37] M. Biomo *et al.*, "Routing in unmanned aerial ad hoc networks: A recovery strategy for greedy geographic forwarding failure," in *2014 Wireless Comm. and Networking Conf.* IEEE, 2014, pp. 2236–2241.

[38] H. Yang and Z. Liu, "An optimization routing protocol for fanets," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 120, Accessed April 2021 2019. [Online]. Available: https://doi.org/10.1186/s13638-019-1442-0

[39] V. Sharma, R. Kumar, and N. Kumar, "Dptr: Distributed priority tree-based routing protocol for fanets," *Computer Communications*, vol. 122, pp. 129 – 151, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366416303267

[40] A. Nayyar, "Flying adhoc network (fanets): Simulation based performance comparison of routing protocols: Aodv, dsdv, dsr, olsr, aomdv and hwmp," in *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*. IEEE, 2018, pp. 1–9.

[41] M. Y. Arafat and S. Moh, "Localization and clustering based on swarm intelligence in uav networks for emergency communications," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[42] A. Khan, F. Aftab, and Z. Zhang, "Bicsf: Bio-inspired clustering scheme for fanets," *IEEE Access*, vol. 7, pp. 31 446–31 456, 2019.

[43] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[44] M. Roderick, J. MacGlashan, and S. Tellex, "Implementing the deep q-network," *arXiv preprint arXiv:1711.07478*, 2017.

[45] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2018. [Online]. Available: https://books.google.com/books?id=sWV0DwAAQBAJ

[46] S. Yin, S. Zhao, Y. Zhao, and F. R. Yu, "Intelligent trajectory design in uav-aided communications with reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8227–8231, 2019.

[47] M. Kaisers and K. Tuyls, "Frequency adjusted multi-agent q-learning," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 2010, pp. 309–316.

[48] M. Coggan, "Exploration and exploitation in reinforcement learning," *Research supervised by Prof. Doina Precup, CRA-W DMP Project at McGill University*, 2004.

[49] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[50] J. Chung, W. Zacherek, A. Wisniewski, Z. Liu, T. Bicer, R. Kettimuthu, and I. Foster, "Scistream: Architecture and toolkit for data streaming between federated science instruments," in *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '22. New York, NY, USA:

Association for Computing Machinery, 2022, p. 185–198. [Online]. Available: https://doi.org/10.1145/3502181.3531475
[51] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.
[52] M. A. M. Vieira, M. S. Castanho, R. D. G. Pacífico, E. R. S. Santos, E. P. M. C. Júnior, and L. F. M. Vieira, "Fast packet processing with EBPF and XDP: Concepts, code, challenges, and applications," *ACM Comput. Surv.*, vol. 53, no. 1, feb 2020. [Online]. Available: https://doi.org/10.1145/3371038
[53] J. Sitnicki, "Steering connections to sockets with bpf socket lookup hook," 2020, eBPF Summit. [Online]. Available: https://ebpf.io/summit-2020-slides/eBPF-Summit-2020-Lightning-Jakub-Sitnicki-Steering-connections-to-sockets-with-BPF-socke-lookup-hook.pdf
[54] G. Perna, M. Trevisan, D. Giordano, and I. Drago, "A first look at HTTP/3 adoption and performance," *Computer Communications*, vol. 187, pp. 115–124, 2022.
[55] C. W. Lucus Pardue, "Unlocking QUIC's proxying potential with MASQUE," https://blog.cloudflare.com/unlocking-quic-proxying-potential/, [Online accessed May-2023].
[56] M. Kühlewind, M. Carlander-Reuterfelt, M. Ihlar, and M. Westerlund, "Evaluation of QUIC-based MASQUE proxying," in *Proceedings of the 2021 Workshop on Evolution, Performance and Interoperability of QUIC*, 2021, pp. 29–34.
[57] ietf, "HTTP Datagrams and the Capsule Protocol, RFC 9297," https://datatracker.ietf.org/doc/rfc9297/, [Online accessed May-2023].
[58] K. Kumari, B. Sah, and S. Maakar, "A survey: different mobility model for fanet," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 5, no. 6, 2015.
[59] D. A. Korneev, A. V. Leonov, and G. A. Litvinov, "Estimation of mini-uavs network parameters for search and rescue operation scenario with gauss-markov mobility model," in *2018 IEEE Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, July 2018, pp. 1–7.
[60] C. Qu, F. B. Sorbelli, R. Singh, P. Calyam, and S. K. Das, "Environmentally-aware and energy-efficient multi-drone coordination and networking for disaster response," *IEEE Transactions on Network and Service Management*, 2023.
[61] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "FABRIC: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.
[62] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," in *Proceedings of PFLDnet*, 2004.
[63] R. R. Ramisetty, C. Qu, R. Aktar, S. Wang, P. Calyam, and K. Palaniappan, "Dynamic computation off-loading and control based on occlusion detection in drone video analytics," in *Proceedings of the 21st International Conference on Distributed Computing and Networking*, ser. ICDCN 2020. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3369740.3369793
[64] Nvidia, "Nvidia embedded systems for next-gen autonomous machines." [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/

Chengyi Qu received his Ph.D. degree in the Department of Electrical Engineering and Computer Science at the University of Missouri-Columbia, USA in 2023. He is currently an Assistant Professor in Florida Gulf Coast University, USA. His current research interests include: ad-hoc networks, cloud/edge computing, cybersecurity, and drone video analytics. He is a IEEE member and IEEE ComSoc member.

Rounak Singh received his B.S. degree in Electronics and Communication Engineering from G.I.T.A.M Deemed to be University, Hyderabad, India in 2019. He is currently a graduate student pursuing his M.S degree in Electrical and Computer Engineering at University of Missouri-Columbia, USA. His research interests include: machine learning, deep learning, reinforcement learning, computer vision and cloud computing.

Alicia Esquivel-Morel received her M.S. degree in 2020, and is currently pursuing her Ph.D. in the Department of Electrical Engineering and Computer Science at the University of Missouri-Columbia, USA. Her current research interests include: cloud computing, UAV systems, machine learning, and cybersecurity.

Prasad Calyam received his M.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering at The Ohio State University in 2002 and 2007, respectively. He is currently an Professor in the Department of Electrical Engineering and Computer Science at University of Missouri-Columbia, USA. He directs the Virtualization, Multimedia and Networking (VIMAN) Lab, as well as the Center for Cyber Education, Research and Infrastructure (Mizzou CERI). His current research interests include: distributed and cloud computing, computer networking, and cybersecurity. He is a Senior Member of IEEE.