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

Chengyi Qu*, Rounak Singh*, Alicia Esquivel-Morel* and Prasad Calyam†
*† Department of Electrical Engineering and Computer Science, University of Missouri - Columbia, USA.
Email: *{cqy78, rsft6, ace6qv}@mail.missouri.edu, †calyamp@missouri.edu

***Abstract*—Unmanned aerial vehicles (a.k.a. drones) with high-resolution video cameras are useful for applications in e.g., public safety and smart farming. Inefficient configurations in drone video analytics applications due to edge network misconfigurations can result in degraded video quality and inefficient resource utilization. In this paper, we present a novel scheme for offline/online learning-based network edge orchestration to achieve pertinent selection of both network protocols and video properties in multi-drone based video analytics. Our approach features both supervised and unsupervised machine learning algorithms to enable decision making for selection of both network protocols and video properties in the drones' pre-takeoff stage i.e., *offline* stage. In addition, our approach facilitates drone trajectory optimization during drone flights through an *online* reinforcement learning-based multi-agent deep Q-network algorithm. Evaluation results show how our offline orchestration can suitably choose network protocols (i.e., amongst TCP/HTTP, UDP/RTP, QUIC). We also demonstrate how our unsupervised learning approach outperforms existing learning approaches, and achieves efficient offloading while also improving the network performance (i.e., throughput and round-trip time) by least 25% with satisfactory video quality. Lastly, we show via trace-based simulations, how our online orchestration achieves 91% of oracle baseline network throughput performance with comparable video quality.**

***Index Terms*—Multi-access edge computing, Multi-drone networks, Reinforcement learning, Network protocols**

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.

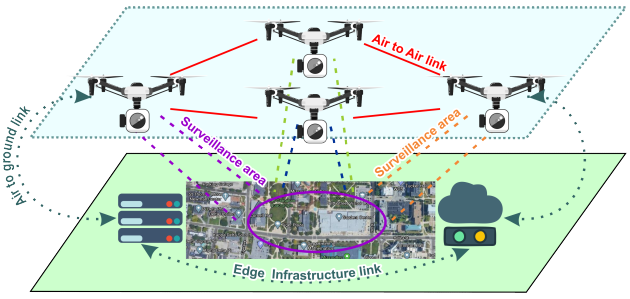

Fig. 1: Overview of multi-drone video analytics setup based on air-to-air and air-to-ground links with edge servers.

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 learn-

ing 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, considering diverse and realistic system setup configurations to develop our learning-based approach, we use a trace-based simulator dataset [8] in our multi-drone video analytics performance studies. We collect drone traces with different mobility models and application scenarios from both real-world drone experiments as well as in simulations. On these traces, we apply supervised learning based strategies *offline* to intelligently select potential video capture and network protocol setup combinations based on expected network characteristics and video quality requirements. Trace-based performance evaluation experiments with our supervised-learning based approach shows how video quality delivery using suitable control networking matches real-world measurements in terms of machine learning model accuracy.

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 are used to overcome the challenges of communication in a multi-drone environment. In [9], authors addressed the problem of enabling the automated orchestration of smart edge devices through edge and cloud microservice platforms. 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]. In our work, we also leverage the multi-access edge computing paradigm in our drone-edge setups, and 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.

Machine learning as a subset of artificial intelligence, provides techniques that typically fall under three main categories: supervised, unsupervised and reinforcement learning. Supervised/unsupervised learning specifically uses a set of labeled/unlabeled data samples to learn and map between the input and output spaces [10], [11]. In our work, we are interested in the investigation of strategies for applying supervised learning-based methods to intelligently provide suitable network protocol and video capture setups in multi-drone video analytics related applications. In addition, our work similarly applies unsupervised learning clustering algorithms to assist multi-drone video analytics users in their decision process to select 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 [12] 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 [13] [14] 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 [15]. Authors in [16] 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 [16] 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.

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.

A. Trace Data Collection and Dataset Components

Figure 2 illustrates the multi-drone video analytics with edge network orchestration process steps. We use an openly-accessible trace-based simulator dataset [8] that considers drone video analytics running on multi-drone configurations with wireless network control. The real-world traces involve

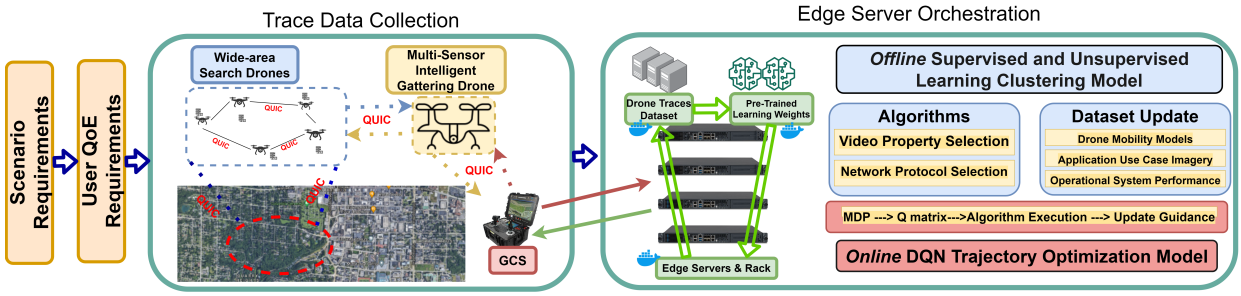

Fig. 2: multi-drone video analytics with edge network orchestration process diagram showing how application user requirements are considered in a trace dataset collection based on which we develop *offline* and *online* edge server orchestration algorithms.

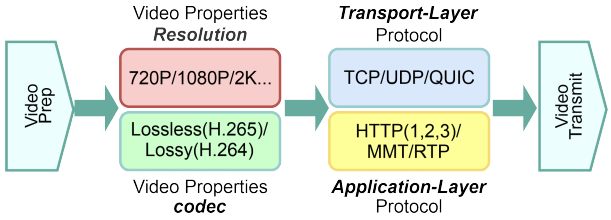

Fig. 3: Network protocol and video properties selection process applied after the video preparation from the drone camera, and before the transmission process from drone to ground.

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.

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. To aid the decision making, we map the application-layer and transport-layer protocols and associate them with video transmission properties of: network protocol and video properties. In terms of the input objects, we considered two broad categories with total of five parameter: (i) Analytics Layer: Real-time analytics level, video quality level, and (ii) System Layer: Parallel level, bandwidth usage, and on-flight CPU level. As for the testing dataset, only the input object will be used, and the supervised learning algorithm will learn the desired output value by itself.

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 [17]. 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 [8] in the learning engine, and information related to drone mobility models, application imagery and operational system performance 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.

IV. 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 [18] 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 categorizing 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.

2) Testing Phase: To test the accuracy of the machine learning model predictions, we use 95% confidence interval range of accuracy on correctly categorized data. In addition, to evaluate the overall video quality, we use PSNR as a performance metric as detailed in Section VI-B.

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 determe 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.

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

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.

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.

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. Reinforcement Learning based Drone Trajectory Optimization

In this section, we first describe the motivations for our *online* orchestration. Following this, we present our reinforcement learning based drone trajectory optimization algorithm.

A. Orchestration Motivation for Online Learning

Unsupervised learning could receive cluster combination options *offline* i.e., in the pre-takeoff stage of drones. However, during mid-flight operation, there might be a necessity to change the network protocol or video properties *online* to achieve better results which satisfy user experience expectations. For this purpose, we need to analyze the trajectories of drones and video quality by selection of pertinent network protocol and video codecs. We propose a multi-agent DQN algorithm for intelligent path planning of the drones.

B. Multi-Agent Deep Q-Network

To achieve intelligent trajectory learning, we propose a multi-agent Deep Q-Network [19], [20] 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 [21] (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 [22]. 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 [23] by considering a continuous state space that allows the drones to explore and exploit [24] 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.

C. 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 -

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

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 -

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

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.

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

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.

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

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-

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

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 -

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

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 -

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

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.

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 [25]. 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 [26].

TABLE I: Environment Settings used in Experiments

Application Settings		**Network Settings**	
Number of drones:	10-30	Application 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 I.

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) to measure the quality of the video streamed by the drones. 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. Values of PNSR above 25 dB and upto 40 dB are deemed to be good for user experience of video quality, and particularly 32 dB and higher values are ideal to meet most users' satisfaction levels of video quality. Hence, we assume that the transmitted video quality can be rectified effectively by camera control if the PSNR is less than or equal to a fixed threshold e.g., $\leq$30 dB. If the PSNR value is within such limits, then the impairments in the video can be mitigated by instructing the drone through the controller (at the edge server) to stream higher-resolution video. If the PSNR threshold is e.g., $>$30, then the controller communicates to a nearby drone to capture the required high-quality video in the drone-edge network. This process iterates until the required high-resolution quality of the video is streamed as part of the video streaming process across the drone-edge network.

The subjective metric we consider in the unsupervised learning based algorithm is the Mean Opinion Score (MOS), which is widely used metric for subjective assessment of video quality with human subjects. MOS is used to rank perceptual quality of an end-user on a subjective quality scale of 1 to 5. The [1, 3) range corresponds to "Poor" grade where an end-user perceives severe and frequent impairments that make the application unusable. The [3, 4) range corresponds to "Acceptable" grade where an end-user perceives intermittent impairments yet the application is mostly usable. Lastly, the [4, 5] range corresponds to "Good" grade where an end-user perceives none or minimal impairments and the application is always usable.

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 II. 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.

TABLE II: The average training 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	0.120$\pm$0.00362	**0.004$\pm$0.00272**
SVR-RBF	**0.099$\pm$0.01620**	0.068$\pm$0.00252
GPR	2.170$\pm$0.00100	1.962$\pm$0.00000
RFR	0.205$\pm$0.05400	0.156$\pm$0.05400

We compared our learning-based trace simulation results with previous policy-based estimation and pure NS-3 simulation traces [27]. The scheme in [27] uses a simple policy-based decision-making algorithm to determine network protocol and video properties. As shown in Table III, 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 III 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.

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 Index (PC). DB index considers the dispersion and separation

TABLE III: 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	(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]

TABLE IV: 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

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 IV, 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 4 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.

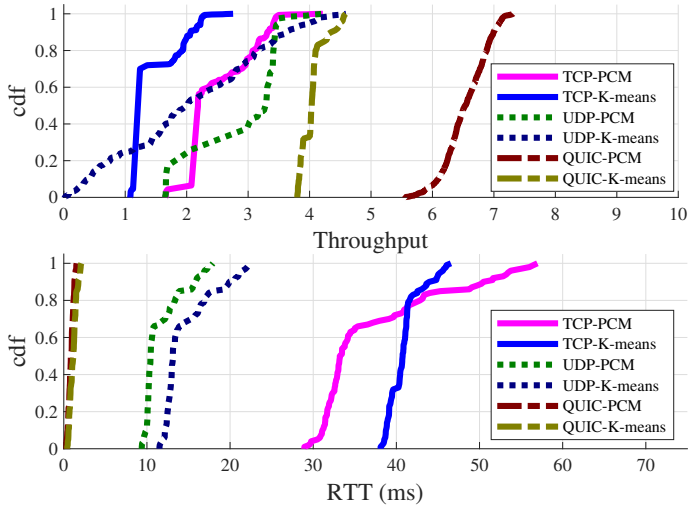

Fig. 4: Cumulative distribution function (cdf) of: network throughput (Mbps), and RTT (ms) - for the PCM and K-means clustering algorithms; each group is categorized according to TCP, UDP and QUIC protocol.

2) PCM improves overall video quality performance after transmission: Table V 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.

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

Figure 5 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 simulation could provide any potential combinations without limitations. Baseline data represents the oracle situation when

TABLE V: 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

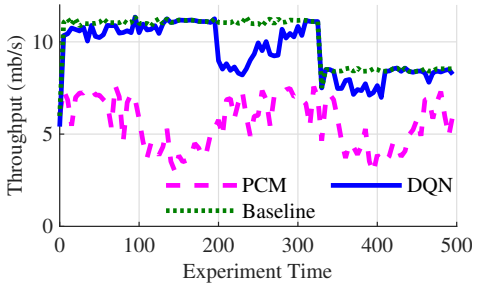

Fig. 5: Throughput performance comparison of PCM, DQN (with LR = 0.001) and Baseline in a trace-based experiment.

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 5, 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. 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 [28]. 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.

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.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

[19] 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.

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

[21] 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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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

[28] Nvidia, "Nvidia embedded systems for next-gen autonomous machines." [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/