DroneNet-Sim: A Learning-based Trace Simulation Framework for Control Networking in Drone Video Analytics

Chengyi Qu, Alicia Esquivel Morel, Drew Dahlquist, Prasad Calyam
University of Missouri-Columbia, USA
(cqy78,ace6qv,dgdtx5)@mail.missouri.edu;calyamp@missouri.edu

ABSTRACT

Unmanned Aerial Vehicles (UAVs) or drones equipped with cameras are extensively used in different applications for environmental situational awareness such as: smart agriculture, border security, intelligent transportation. Realistic UAV testbed building for developing novel network control algorithms relating to video streaming/analytics is time-consuming and difficult. Challenges arise when executing high-scale drone video analytics experiments due to: constraints in drone manufacturing, government regulation restrictions, and limited energy resources. Also, developing algorithms requires ability to understand impact of network protocol selection to handle diverse UAV mobility models as well as dynamic network status during edge-network video processing. In this paper, we propose a novel learning-based trace simulation framework viz., "DroneNet-Sim" that integrates simulation on both drone and networking sides. It allows for experimentation with network protocol selection (i.e., TCP/HTTP, UDP/RTP, QUIC) and video properties selection (i.e., codec, resolution) to ensure satisfactory video quality delivery in different drone flight scenarios. Using machine learning models, we show how DroneNet-Sim can process real-world drone traces that include various mobility models, geospatial link information and on-time network status obtained from real-world data-gathering efforts. Trace-based experiments with our DroneNet-Sim shows how video quality delivery (i.e., PSNR) using suitable control networking matches real-world measurements in terms of machine learning model accuracy.

CCS CONCEPTS

• **Networks** → **Transport protocols**; • **Information systems** → *Users and interactive retrieval.*

KEYWORDS

Drone video analytics, networking protocols, trace-based simulation, machine learning

ACM Reference Format:
Chengyi Qu, Alicia Esquivel Morel, Drew Dahlquist, Prasad Calyam. 2020. DroneNet-Sim: A Learning-based Trace Simulation Framework for Control Networking in Drone Video Analytics. In ,. ACM, New York, NY, USA, 6 pages. https://doi.org/N/A

N/A, N/A

ACM ISBN N/A…$15.00
https://doi.org/N/A

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) or drones equipped with high-resolution cameras are increasingly being used to visualize and monitor targets of interest. In most cases, drones allow computer systems, e.g., Ground Control Stations (GCS), to actively control their location, allowing them to be coordinated to enhance the efficiency in providing environmental situational awareness for applications in e.g., smart agriculture, border security, intelligent transportation. As shown in Figure 1, these systems often feature air-to-ground wireless links that connect the edge-network with UAVs and Ground Control Stations (GCS). Peer-to-peer or centralized-control communication networks are typically setup using two networking modes: UAV-to-UAV and UAV-to-Infrastructure [6].

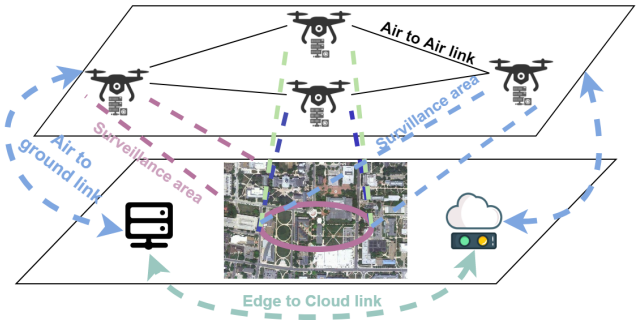

Figure 1: Overview of multiple drone-GCS communication using an edge-network infrastructure.

However, realistic UAV testbed building for developing novel network control algorithms relating to video streaming/analytics is time-consuming and difficult. Challenges arise when executing high-scale drone video analytics experiments due to: constraints in drone manufacturing, government regulation restrictions, and limited energy resources. Also, developing algorithms requires the ability to understand impact of network protocol selection to handle diverse UAV mobility models as well as dynamic network status during edge-network video processing. In addition, experimenters find it difficult to create repeatable and reproducible experiments using realistic variants of the setup shown in Figure 1 to test major hypotheses involving different application scenarios.

Trace-based simulation approaches for drone video analytics involving different control networking configurations can provide potential solutions to overcome the above challenges. Prior works that use trace-based simulation focus mainly on either drone simulations [7], [1] or network simulations [4], [5], [3] in a disjointed manner. Consequently, they are not helpful for network protocol experimentation with edge-cloud configurations. In addition, realistic testbeds for drone video analytics need to feature realistic environmental conditions that may affect the performance of the

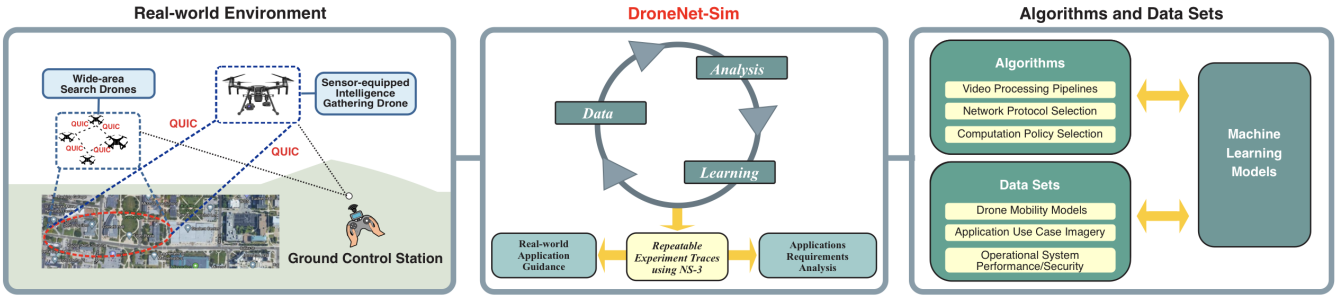

Figure 2: Illustration showing our flight trace generation and learning-based simulation approach that is based on network protocol prediction using DyCOCo algorithms [11] for hierarchical drone-GCS scenarios.

video streaming between the drone and GCS. This in turn affects the performance of video streaming in terms of end-to-end delay, frame blurring, stalling and distortion, which impacts user expectations in video quality delivery [2]. Based on our observations from [8], there is a need for integration of machine learning models within the trace-based simulations to mimic dynamics of networking environments occurring during drone control orchestration in drone video analytics.

In this paper, we propose a novel learning-based trace simulation framework viz., "DroneNet-Sim" that synergistically integrates simulation on both drone and networking levels. It allows for simulation experiments with network protocol selection (i.e., TCP/HTTP, UDP/RTP, QUIC) and video properties selection at drones/GCS (i.e., codec, resolution) to ensure satisfactory video quality delivery in different drone flight scenarios. Using various machine learning models, it allows for simulations that involve prediction of non-ideal network conditions due to improper network protocol selection and video properties selection. This in turn leads to impaired video quality and affects the ability of drone video analytics to help assess applications to chart a plan of action across wide-area scenes.

Our DroneNet-Sim implementation [10] integrates traces generated from real-world experiments with the NS-3 simulator, and allows for experimentation with low-latency network protocols such as QUIC [3]. The simulator scripting supports the ability to integrate different commercial drone configurations, wireless communication links (air-to-air; air-to-ground), as well as network protocols on data transmission. Another major contribution of DroneNet-Sim in this work involves the rich web-based visualization user interface for configuration of simulation experiments with contextual markers of simulation objects. The user interface thus allows for convenient visibility into the various simulation related data sets such as: application use case imagery, drone mobility and operational system/network performance.

The remainder of the paper is organized as follows: In Section 2, we detailed the learning-based trace simulation approach and our user interface design. Section 3 presents various supervised machine learning algorithms for heuristic decision-making within trace-based simulations. Section 4 describes our performance evaluation experiments and results on control networking in drone video analytics. Section 5 concludes the paper.

2 LEARNING-BASED SIMULATION APPROACH

In this section, we first describe the components of the learning-based trace simulation approach. Following this, we outline the analysis process and the learning model. Lastly, we discuss the trace-based simulation and user interface design considerations.

2.1 Components Overview

Figure 2 illustrates our learning-based trace simulation approach to experiment with multi-drone edge-network/server scenarios. In the application scenarios, we assume hierarchical drones for video capture co-ordination. Specifically, we are interested in the communication links involving a number of search drones that have peer-interactions in order to inform a more highly-capable sensor-equipped intelligence collection drone to obtain environmental situational awareness. Three main logical modules are included as shown in Figure 2 in the system: (i) the trace-based simulation and visualization module (*data*), (ii) application environment analytics module (*analysis*), and (iii) learning-based optimal scheme selection module (*learning*). The problems of insufficient computing resources and network bandwidth at the edge-network can be addressed using computation offloading to GCS, which is solved based on our previous work in DyCOCo [11] [12]. Using a machine learning approach and DyCOCo algorithms, we can learn from analyzed flight traces with corresponding network settings and video transmission properties to label the status and categorize the conditions. However, learning a static condition is not ideal for all the time steps during a flight. We use a network simulator to simulate the updated traces that reflect the dynamic status. With the updated traces, we can visualize the drone flight traces to provide useful guidance for: (a) drone operators to perform control action decisions, and (b) help experimenters to test hypotheses.

2.2 Analysis Process

For the *analysis* and *learning* steps, we consider application requirements in e.g., traffic management, disaster response, and smart farming. The requirements involve real-world environments with a hierarchical drone-GCS system that involves local(on the drone) or edge-network (at the GCS) video recording and processing. Search drones have low computational resources on-board to avoid large payload that limits the flight time. Thus, they will typically offload their tasks to the GCS to save time, energy and resources. For choosing among the various transmission protocols and video codec types in the search drones, our approach involves choosing

amongst multiple options to match application context and requirements. For instance, if one application scenario only allows specific transport protocol, e.g., UDP, or specific video resolution, e.g., 720P, we will maintain the settings and enhance the video transmission quality and analytics accuracy by computation offloading algorithms in DyCOCo [11]. If the application scenarios have custom settings or drone operators/researchers have specific performance requirements, we use machine learning models to analyze the requirements and learn from previous similar flight environment traces (see Section 3 for details). Specifically, we apply learning-based model prediction results into simulation of network protocols such as e.g., QUIC protocol. The learning is performed in terms of: drone mobility model, geographical environment, expected flight traces and predicted wireless signal strength. Such a learning can consequently provide a pertinent solution on networking protocol and video properties selection.

2.3 Trace-based Simulation Design

We use a trace-based simulation approach due to the motivations in Section 1 for drone video analytics to select suitable control networking configurations related to network protocol selection and video properties selection. The simulator scripting supports the ability to integrate traces of different commercial drone configurations, wireless communication links, as well as network protocols on data transmission. In our DroneNet-Sim framework, we use NS-3 as the underlying network simulator to generate traces. After we get the machine learning weights from historic traces, we run scripts on NS-3 with the same network protocols and video properties settings to generate new traces.

Our approach to use NS-3 as the underlying trace-based network simulator is motivated by the fact that it is a popular network simulation tool. It effectively handles trace-based models that provide high availability and reproducible features. An extension given in [5] developed an advanced trace-based model that supports multiple-access wireless scenarios. This model can reproduce past experiments and generate traces with the same settings in real-world experiments. Our trace-based simulation supports a similar advanced model and implements different transport layer and application layer protocols supported in NS-3, which includes: TCP and UDP default model, simple IETF QUIC model [3], HTTP simulation model and HTTPs simulation model.

Multi-drone experiments are hardly repeatable in most cases given the complexity of application scenarios and corresponding mission requirements. Given the same input on drone information, scene information and application requirements, drone flights can end up producing very different output results. This is because, wireless communications are influenced by external random phenomena such as noise/interference, and also by multi-path factors. Moreover, realistic experiments are difficult to reproduce due to testbed operational constraints and availability. To generate traces that nearly match the actual flight traces in simulation, we need to analyze the flight traces in different settings and configure the simulator correspondingly.

2.4 User Interface Design

To assist users to visualize experiments with algorithms and data sets, we have implemented a web-based user interface (UI) in DroneNet-Sim. Trace generation and visualization are fully decoupled so that the simulator is flexible and extensible for new capabilities. Our UI provides visualization in an animated and intuitive manner to the simulator users (i.e., scientists, researchers and hobbyists). In addition, the web-based UI is designed to present multi-drone flight animations based on traces generated by either real-world experiments or trace-based simulation. Further, it supports multi-drone visualization animation along with network information and tracking accuracy over prescribed time periods.

Figure 3 shows the user interface screenshot of DroneNet-Sim. We adopt best practices for the front-end and back-end design and development of the web-based UI. The interactions between the front-end and the back-end are executed in a matter of seconds in order to provide an interactive user experience for the DroneNet-Sim. Importantly, the web-based UI is built on top of open-source frameworks, including JavaScript as the main component that provides reliability of the data displayed. For the front-end, we leverage the simplicity of Vuetifyjs, which is built on top of Vuejs and provides clean, semantic and reusable components to help maintain the source code improvements in a sustainable manner. Lastly, the back-end of our web-based UI runs on top of Node.js. Node.js is a JavaScript runtime and an open-source server environment that runs on various platforms including Windows, Linux, Mac OS and others. It uses JavaScript on the server and can generate dynamic page content. In addition, the back-end relies on AdonisJS, which is a Node.js Model View Controller framework that provides flexible coding structures.

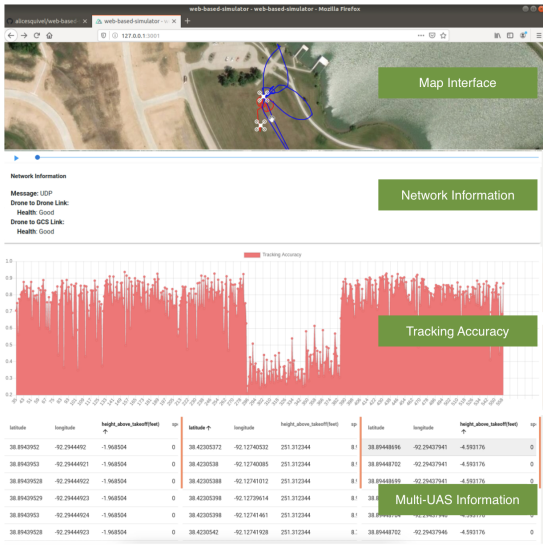

Figure 3: UI of an exemplar DroneNet-Sim simulation.

3 MODELING USING SUPERVISED LEARNING

In this section, we first introduce our problem formulation, which includes analysis on drone video properties, networking protocols and application pipelines. Next, we present our machine learning algorithms to integrate traces from real-world experiments into analytics-layer and system-layer categories. Lastly, we discuss results on training and testing phases in choosing weighted machine learning models for control networking at the edge-network.

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

3.1 Problem formulation

For users to choose from the available options of network protocols and video properties shown in Figure 4, it is essential to collect application requirements. Table 1 describes the application requirement parameters in a multi-UAS setup organized under two broad categories: (i) Analytics Layer, and (ii) System Layer. Based on the user requirements, we will map different solutions into lists of possible choices with our categories. Our categories support different kinds of protocols, video codecs and video resolution choices as shown in Figure 4. The candidate choices represent one option in each of the four categories, including: (i) Application-layer Protocol (choice between HTTP/1.1, HTTP/2, and HTTP/3; may also include MMT (MPEG media transport) and RTP in some specific cases), (ii) Transport-layer Protocol (choice between TCP, UDP and QUIC), (iii) Video Codec (choice between H.265 HEVC (High Efficiency Video Coding) and H.264 AVC (Advanced Video Coding), and (iv) Video Resolution (choice between 720p, 1080p and 2K). To make the prediction easier, we map the Application-layer and Transport-layer Protocols and associate them with video transmission properties of: network protocol, video codec and video resolution.

In real scenarios, due to the uncertain wireless environment, it is difficult to predict the network conditions to give a precise decision during the drone flight. Moreover, it is very likely to have rapid network protocol changes during the flight paths corresponding to the application requirements of environmental situational awareness. Thus, a network protocol and video properties selection at the beginning of a drone flight is significant. Hence, we use machine learning to choose the best control networking configurations by learning from a database of real-world labeled traces. Our database comprises of 400 real-world traces collected from DyCOCo [11] experiments and an open-source drone video database [13]. The traces we consider for training include: (i) settings of transport protocols, application protocols and video properties as preliminary application data, (ii) captured network packets, video streams and flight traces as the real-time/live application data, and (iii) measurement of the overall network status, video quality measurements during the whole flight as a post-application data. All the measurements in the post-application data map to either high, medium, or low quality scores rated subjectively by users. The standard on how we give a high score is given by the parameter the user provides in the preliminary application data. If the average of user scores results in higher values, we will evaluate traces as a 'high case'. If the average results match the values given by the user, we will evaluate traces as a 'medium case'. Similarly, if average results are less than those expected by the users, we will treat the traces as a 'low case'.

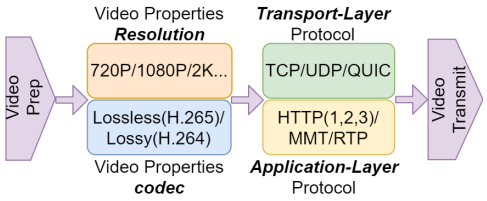

Figure 4: DroneNet-Sim network protocol and video properties choices.

3.2 Training Phase

The training phase is a step 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 control networking solutions 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.

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

Model Type	Protocol Prediction	Video Property Prediction
KRR	0.120±0.00362	**0.004±0.00272**
SVR-RBF	**0.099±0.01620**	0.068±0.00252
GPR	2.170±0.00100	1.962±0.00000
RFR	0.205±0.05400	0.156±0.05400

With the collected data sets, we used machine learning to predict the networking 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 supervised learning approach to achieve prediction and classification. In the training phase, four machine learning models from the Sci-Kit Learn toolkit [9] 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 these machine learning models, we were primarily concerned about the prediction accuracy on networking

protocols and video properties. However, with all machine learning approaches, they achieve nearly the same accuracy on prediction (0.95±0.036). Hence, we prefer the shortest training time with relative smaller RMSE (Root Mean Square Error) as the better choice shown in Table 2. For the network protocol prediction, we found that the machine learning model that had the shortest training time was the SVR-RBF model. For the video properties prediction, the best performance was seen in the Kernel-ridge Regression.

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 choose the most optimal choice in terms of network protocols and video properties, and configure them as preliminary application settings for the drone flight paths.

3.3 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 (peak signal-to-noise ratio) as a performance metric. 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, and PSNR measures the video quality at the destination of a communication link.

4 PERFORMANCE EVALUATION

In this section, we first introduce the evaluation setup and data sets collected from both real-world experiments and drone simulator in 3 use cases. Next, we compare the end-to-end performance of TCP and QUIC transport protocols used by the drones in the NS-3 simulations to show our trace-based selection strategy efficacy. Lastly, we present the traces generated from real-world experiments and compare the results with pure NS-3 simulation to show the advantages of using our learning-based approach.

4.1 Evaluation Setup

For evaluation of our DroneNet-Sim framework, we collected data from a multi-hierarchical drone configuration use centralized-control communication networks, where we consider various mobility models and different amount of drones as fleets. Each drone is operated by one GCS, and all traces and computation processes are controlled by one single edge server. To simplify our experiments, two parts of data are considered: drone video capture data and drone trace data. On drone video capture data gathering, we collected 50 video clips from 3 types of drones (DJI Phantom3, JDI Mavic 2 and Parrot Drone (on drone simulator), each with 3 different mobility settings (i.e., mission-based plan model, Gaussian-Markov model and random model). Drone video clips are transmitted to GCS via a simple, reliable and widely available data link, such as IEEE 802.11. We calculate metrics on both video codec type, and video resolution. On drone flight trace gathering, we use real-world experiments to generate a fleet of 2 and 4 drones, and use a drone simulator [7] to generate a fleet of 6 drone cases. Traces are formatted as JSON files and stored in a NoSQL database. We calculate metrics on: (i) the goodput through a network monitor at a GCS, and (ii) detailed flight traces information (e.g., longitude, height, speed (mph), ascent (feet)). For every millisecond, one throughput data point is sent from the drones to the GCS for recording/archival. In addition, after the real-world experiment, we calculate the PSNR for transmitted videos to validate the utility of our machine learning model predictions.

4.2 Drone and Network Simulation Results

Figure 5 shows the experiment on traces generated from real-world application as well as drone simulator configurations featuring multi-drone scenarios and different mobility models, in terms of goodput performance. Experiment used the same mission-plan based mobility model for all scenarios in action for different fleet of drones. In this mission-plan based mobility model, each drone gets specific tasks to fly against each other with a maximum relative distance of 15 meters, and lastly flies back to the base station. As we can observe in Figure 5, when drone flies far away from other peers, the goodput between drones increasingly become lower.

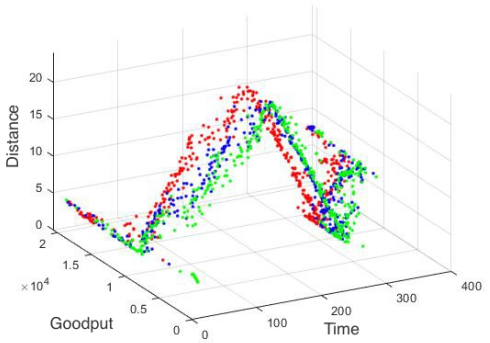

Figure 5: Drone traces with communication goodput on 3 UAV fleets; 2 (red), 4 (blue), and 6 (green) drones with one edge server connecting with RTP/UDP.

In addition to using a drone simulator to generate traces, DroneNet-Sim also supports various transport layer protocol changes. To simplify our experiment, we compare the performance of QUIC and TCP networking protocols in terms of round-trip times (RTT) using the underlying NS-3 simulator within DroneNet-Sim. Figure 6 shows average RTT Cumulative Distribution Function (CDF) pertaining to implementations of QUIC and TCP protocols in both real-world experiments, and NS-3 simulations. We can see that NS-3 can achieve better results without exceeding the limitation of protocol settings. For instance, TCP in NS-3 performance is always lower than QUIC in the real-world experiment. These results prove that we can easily distinguish protocol performances in either real-world experiments or in simulations. Accordingly, all traces can be generally combined together for machine learning training and testing, and for ultimately providing better solutions for meeting application requirements of environmental situational awareness.

4.3 Learning-based trace simulation results

We compared our learning-based trace simulation results with our previous policy-based estimation and pure NS-3 simulation traces [12]. The scheme in [12] uses a simple policy-based decision making algorithm to provide protocol and video properties based on prior experiences. As shown in Table 3, without the help of machine learning predictions, the accuracy on a given selection of networking protocol and video property is relatively low and uncertain. Even for the PSNR testing after reproducing the traces, policy-based estimation can only achieve PSNR results around 30.

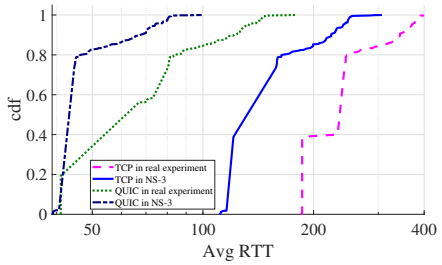

Figure 6: Comparison between applying real-world TCP, NS-3 TCP, real-world QUIC and NS-3 QUIC connections with the same scenario settings in terms of average RTT (in ms).

We evaluated the performance of our 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 Mpbs to 2 Gbps. Table 3 provides the machine learning model results on networking protocol and video properties selection. We compared results with real-world experiments for each model and also calculated the 95% CI of accuracy for each model. We firstly conclude that - RFR gives more accuracy on networking protocol and video property selection, however the decision performance is in the unstable range of PSNR within the transmitted video. Secondly, we conclude that the KRR can generate better results with a stable range of PSNR, although the performance is lower than other machine learning models.

Table 3: Policy-based estimate (baseline) and machine learning prediction results comparison.

Model Type	Protocol 95% CI	Video Property 95% CI	PSNR
Policy-based Estimate	(0.267, 0.6577)	(0.1904,0.2715)	(26.71, 33.59)
KRR	(0.652, 0.928)	(0.803,0.927)	(39.86, 46.93)
SVR-RBF	(0.779, 0.833)	(0.9034,0.952)	(33.59, 46.93)
GPR	(0.813,0.882)	(0.7523,0.8)	(32.26, 39.86)
RFR	(0.9124,0.96)	(0.9032,0.97)	(32.26, 49.37)

Figure 7 shows how our learning-based trace simulation approach performs compared with real-world traces and pure NS-3 traces. Pure NS-3 traces are generated based on the original settings of network protocols. It is obvious that our learning-based traces can generate more accurate results than the pure NS-3 traces model. This is because, our learning-based model can learn all the features from drone traces and infer similar traces with setting changes.

5 CONCLUSION

In this paper, we presented DroneNet-Sim, a framework to realize learning-based control networking in drone video analytics scenarios. We considered scenarios where realism and high-scale are desired to experiment with application requirements, which can be achieved using a trace-based simulation approach. We conducted DroneNet-Sim experiments generated from real-world application configurations featuring multi-drone scenarios and different mobility models, in terms of goodput performance. In addition, we compared the performance of QUIC and TCP networking protocols in

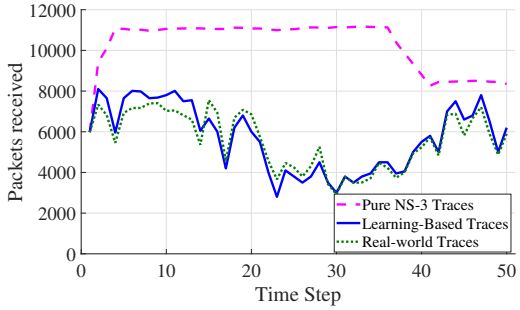

Figure 7: Trace-based simulation fits real-world experiments, in comparison with pure NS-3 simulation.

terms of round-trip times (RTT) in the underlying NS-3 simulation environment within DroneNet-Sim. Further, using machine learning models, we showed how DroneNet-Sim can process real-world drone traces that include various mobility models, geospatial link information and on-time network status obtained from real-world data-gathering efforts. Further, trace-based experiments with our DroneNet-Sim implementation showed how video quality delivery (i.e., PSNR) using suitable control networking matches real-world measurements in terms of machine learning model accuracy.

Our future work involves evaluating the DroneNet-Sim using more advanced settings in network protocols considering real-world experiments. Thus, more comprehensive data for various drone scenarios can be used in the evaluations of the generated traces for different application requirements.

REFERENCES

[1] S. Baidya, Z. Shaikh, and M. Levorato. 2018. FlyNetSim: An Open Source Synchronized UAV Network Simulator based on ns-3 and Ardupilot. In *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, 37–45.

[2] P. Calyam, M. Haffner, E. Ekici, and C.G.Lee. 2007. Measuring Interaction QoE in Internet Videoconferencing. In *Real-Time Mobile Multimedia Services*, Dilip Krishnaswamy, Tom Pfeifer, and Danny Raz (Eds.). Springer, 14–25.

[3] A. De Biasio et al. 2019. A QUIC Implementation for Ns-3. In *Proceedings of the 2019 Workshop on Ns-3 (WNS3 2019)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3321349.3321351

[4] H. Fontes, R. Campos, and M. Ricardo. 2018. Improving the Ns-3 TraceBasedPropagationLossModel to Support Multiple Access Wireless Scenarios. In *Proceedings of the 10th Workshop on Ns-3 (WNS3 '18)*. Association for Computing Machinery, New York, NY, USA, 77–83. https://doi.org/10.1145/3199902.3199912

[5] H. Fontes, R. Campos, and M. Ricardo. 2018. Improving the Ns-3 TraceBasedPropagationLossModel to Support Multiple Access Wireless Scenarios. In *Proceedings of the 10th Workshop on Ns-3 (WNS3 '18)*. Association for Computing Machinery, New York, NY, USA, 77–83. https://doi.org/10.1145/3199902.3199912

[6] M. Khan, I. Qureshi, and F. Khanzada. 2019. A Hybrid Communication Scheme for Efficient and Low-Cost Deployment of Future Flying Ad-Hoc Network (FANET). 3 (02 2019), 22. https://doi.org/10.3390/drones3010016

[7] J. Modares, N. Mastronarde, and K. Dantu. 2016. UB-ANC emulator: An emulation framework for multi-agent drone networks. In *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. 252–258.

[8] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah. 2019. A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems. *IEEE Communications Surveys Tutorials* 21, 3 (2019), 2334–2360.

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

[10] C. Qu and A. Esquivel. 2020. DroneNetsim. https://github.com/CaesarQu/DroneCOCoNet-Sim.

[11] C. Qu, S. Wang, and P. Calyam. 2019. DyCOCo: A Dynamic Computation Offloading and Control Framework for Drone Video Analytics. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. 1–2.

[12] R. Rao, C. Qu, et al. 2020. Dynamic Computation Off-loading and Control based on Occlusion Detection in Drone Video Analytics. *ACM ICDCN* (2020).

[13] P. Zhu et al. 2018. Vision Meets Drones: A Challenge. *arXiv preprint arXiv:1804.07437* (2018).