Bandwidth-efficient Live Video Analytics for Drones via Edge Computing

Junjue Wang*, Ziqiang Feng*, Zhuo Chen*, Shilpa George*, Mihir Bala†, Padmanabhan Pillai‡, Shao-Wen Yang‡, and Mahadev Satyanarayanan*
* Computer Science Department, Carnegie Mellon University, {junjuew, zf, zhuoc, shilpag, satya}@cs.cmu.edu
† University of Michigan, mihirkb@umich.edu
‡ Intel Labs, {padmanabhan.s.pillai, shao-wen.yang}@intel.com

***Abstract*—Real-time video analytics on small autonomous drones poses several difficult challenges at the intersection of wireless bandwidth, processing capacity, energy consumption, result accuracy, and timeliness of results. In response to these challenges, we describe four strategies to build an adaptive computer vision pipeline for search tasks in domains such as search-and-rescue, surveillance, and wildlife conservation. Our experimental results show that a judicious combination of drone-based processing and edge-based processing can save substantial wireless bandwidth and thus improve scalability, without compromising result accuracy or result latency.**

I. Introduction

Continuous video transmission from a swarm of drones places severe stress on the wireless spectrum. Hulu estimates that its video streams require 13 Mbps for 4K resolution and 6 Mbps for HD resolution using highly optimized offline encoding [1]. Live streaming is less bandwidth-efficient, as confirmed by our measured bandwidth of 10 Mbps for HD feed at 25 FPS from a drone. Just 50 drones transmitting HD video streams continuously can saturate the theoretical uplink capacity of 500 Mbps in a 4G LTE cell that covers a large rural area [2]. This is clearly not scalable.

In this paper, we show how edge computing can greatly reduce the per-drone bandwidth demand for video analytics, without compromising the timeliness or accuracy of results. We focus on a class of drones that are autonomous, rather than tele-operated. Once mission-specific flight control software is loaded, autonomous drones can fly completely disconnected. If any wireless bandwidth is consumed, it is solely for real-time analytics. In this paper, "drone" will always mean "autonomous drone."

We present techniques for an adaptive computer vision pipeline for small drones that leverages edge computing to enable dynamic, mission-specific optimizations. Adaptation is crucial to meeting the requirements of diverse missions. For example, rapid discovery of survivors is the dominant concern in a search-and-rescue mission, while stealth may be the crucial requirement of a military mission. Drones have the potential to transform such diverse domains as forestry [3], warfare [4], traffic management [5], and disaster recovery [6]. A swarm of drones, working cooperatively and loosely supervised by a single human operator, as illustrated in Figure 1, has been proposed in the literature as a powerful future paradigm for search tasks [7]. This paper is a step towards realizing that vision.

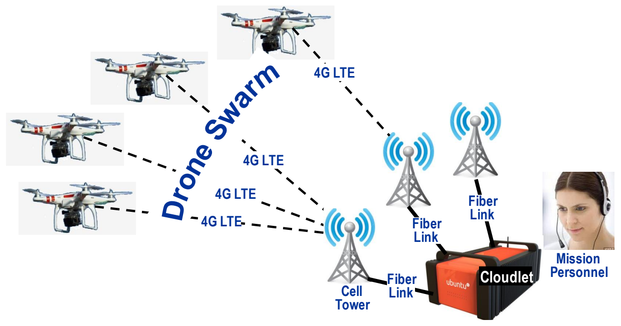


Figure 1. Typical Mission Elements

The main contribution of this paper is to introduce and examine bandwidth saving strategies when offloading computation to an edge node for real-time drone video analysis. In contrast to previous works [8] [9] [10], we leverage state-of-the-art deep neural networks (DNNs) to selectively transmit interesting data from a drone video stream and explore mission-specific optimizations.

Our contributions are as follows:

- A bandwidth-efficient architecture based on edge computing that enables live video analytics for small drones.
- Four different strategies to reduce total transmission: EarlyDiscard, Just-in-Time-Learning, Reachback and Context-Aware.
- Experimental evidence that demonstrates the effectiveness of these strategies in saving bandwidth while minimally impacting result accuracy and latency.

II. Video Processing on Small Drones

In the context of real-time video analytics, small drones represent a "perfect storm" of fundamental mobile computing challenges that were identified two decades ago [11]. Two challenges have specific relevance here. First, mobile elements are resource-poor relative to static elements. Second, mobile connectivity is highly variable in performance and reliability. We discuss their implications below.

A. *Payload and Drone Size*

A high-resolution video camera can be small and light, and be easily carried even by a very small drone. Flash storage to preserve captured video at full fidelity can also fit easily into such a drone. For example, a 16 GB flash chip can store over five hours of HD video, using Netflix's estimate of 3 GB per hour [12]. From our measurement, a drone-encoded HD video occupies 4.7 GB storage space per hour, yielding over three hours of storage on that flash chip. Finally, in spite of the small size and the light weight of a smartphone, its sensing and processing capability are adequate for GPS-based autonomous navigation and flight control [13]. The drone version of this hardware can be even smaller and lighter since interaction components such as the touch-screen display can be omitted. For these reasons, "small" in the context of this paper means "just powerful enough to carry a smartphone as payload." To anticipate future improvements in smartphone technology, our experiments also consider more powerful devices such as the Intel® Joule [14] and the NVIDIA Jetson [15] that are physically compact and light enough to be credible as small drone payloads in a few years.

Unfortunately, the hardware needed for deep video stream processing in real time is larger and heavier than can fit on a small drone. State-of-art techniques in image processing use DNNs that are compute- and memory-intensive. Figure 3 presents experimental results on two fundamental computer vision tasks, image classification and object detection, on five different devices. In the figure, MobileNet V1 and ResNet101 V1 are image classification DNNs. Others are object detection DNNs. Both tasks used publicly available pretrained DNN models. We carefully choose hardware platforms to represent a range of computation capabilities a small drone can carry including the Intel® Aero Drone platform [16] shown in Fig. 2 and NVIDIA Jetson TX2 [17]. In Fig. 3, we present the best results we could obtain on each platform. This is not intended to directly compare frameworks and platforms (as others have been doing [18]), but rather to illustrate the differences between drone-mountable platforms and fixed infrastructure servers.

Image classification maps an image into categories, with each category indicating whether one or many particular objects (e.g., a human survivor, a specific animal, or a car) exist in the image. The prediction speed using two different DNNs are shown. MobileNet V1 [19] is a DNN designed for mobile devices from the ground-up by reducing the number of parameters and simplifying the computation using depth-wise separable convolution. ResNet101 V1 [20] is a more accurate but also more resource-hungry DNN that won the ImageNet classification challenge in 2015 [21].

Object detection is a harder task than image classification, because it requires bounding boxes to be predicted around the specific areas of an image that contains a particular class of object. Object detection DNNs are built on top of image classification DNNs by using image classification DNNs as low-level feature extractors. Since feature extractors in object detection DNNs can be changed, the DNN structures excluding feature detectors are referred as object detection meta-architectures. We benchmarked two object detection DNN meta-architectures: Single Shot Multibox Detector (SSD) [22] and Faster R-CNN [23]. We used multiple feature extractors for each meta-architecture. The meta-architecture SSD uses simpler methods to identify potential regions for objects and therefore requires less computation and runs faster. On the other hand, Faster R-CNN [23] uses a separate region proposal neural network to predict regions of interest and has been shown to achieve higher accuracy [24]. Figure 3 presents results in four columns: SSD combined with MobileNet V1 or Inception V2, and Faster R-CNN combined with Inception V2 or ResNet101 V1 [20]. The combination of Faster R-CNN and ResNet101 V1 is one of the most accurate object detectors available today [21]. The entries marked "ENOMEM" correspond to experiments that were aborted because of insufficient memory.

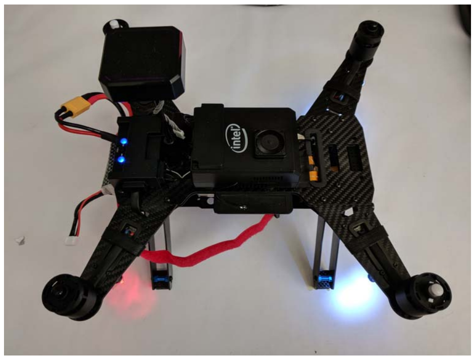


Figure 2. Intel® Aero Drone exemplifies our target class of small drones

These results demonstrates the computation gap between mobile and static elements. While the most accurate object detection model Faster R-CNN Resnet101 V1 can achieve more than two FPS on a server GPU, it either takes several seconds on mobile platforms or fails to execute due to insufficient memory. In addition, the figure also confirms that sustaining open-ended real-time video analytics on smartphone form factor computing devices is well beyond the state of the art today and may remain so in the near future. This constrains what is achievable with small drones.

These constraints do not apply to much larger drones that can carry more substantial computing hardware and energy sources. However, there are compelling reasons to use the smallest drone that meets mission requirements such as flight duration. First, in the context of drone flight regulation (a topic of intense discussion in many countries), small drones are likely to receive favorable consideration since they have less potential to cause severe damage in case of accidents.

M: MobileNet V1; R: ResNet101 V1; S-M: SSD MobileNet V1; S-I: SSD Inception V2;
F-I: Faster R-CNN Inception V2; F-R: Faster R-CNN ResNet101 V1

				Image Classification		Object Detection			
	Weight (g)	CPU	GPU	M (ms)	R (ms)	S-M (ms)	S-I (ms)	F-I (ms)	F-R (ms)
Nexus 6	184	4-core 2.7 GHZ Krait 450, 3GB Mem	Adreno 420	353 (67)	983 (141)	441 (60)	794 (44)	ENOMEM	ENOMEM
Intel® Joule 570x	25	4-core 1.7 GHz Intel Atom® T5700, 4GB Mem	Intel® HD Graphics (gen 9)	37 (1)‡	183 (2)†‡	73 (2)‡	442 (29)	5125 (750)	9810 (1100)
Intel® Aero Drone		4-core 1.6 GHz Intel Atom® x7-Z8750, 4GB Mem	Intel® HD Graphics (gen 8)	42 (1)‡	223 (1)†‡	89 (1)‡	860 (27)	7461 (230)	13 340 (59)
NVIDIA Jetson TX2	85	2-Core 2.0 GHz Denver2 + 4-Core 2.0 GHz Cortex-A57, 8GB Mem	256 cuda core 1.3 GHz NVIDIA Pascal	13 (0)†	92 (2)†	192 (18)	285 (7)†	ENOMEM	ENOMEM
Rack-mounted Server		2x 36-core 2.3 GHz Intel® Xeon® E5-2699v3 Processors, 128GB Mem	2880 cuda core 875MHz NVIDIA Tesla K40, 12GB GPU Mem	4 (0)‡	33 (0)†	12 (2)‡	70 (6)	229 (4)†	438 (5)†

Figures above are means of 3 runs across 100 random images. The time shown includes only the forward pass time using batch size of 1. ENOMEM indicates failure due to insufficient memory. Figures in parentheses are standard deviations. The weight figures for Joule and Jetson include only the modules without breakout boards. Weight for Nexus 6 includes the complete phone with battery and screen. Numbers are obtained with TensorFlow (TensorFlow Lite for Nexus 6) unless indicated otherwise.
† indicates GPU is used. ‡ indicates Intel® Computer Vision SDK beta 3 is used.

Figure 3. Deep Neural Network Inference Speed

Second, larger drones are less maneuverable and more prone to interference with other drone traffic in congested spaces. Third, they are also more expensive to purchase, operate and maintain. Reduced size and weight have many benefits.

B. *Result Latency, Offloading & Scalability*

Result latency is the delay between first capture of a video frame in which a particular result (e.g., image of a survivor) is present, and report of its discovery to mission personnel after video processing. Operating totally disconnected, a small drone can capture and store video, but defer its processing until the drone completes its mission and returns. At that point, the data can be uploaded from the drone to the cloud and processed there. This approach completely eliminates the need for real-time video processing, obviating the challenges of payload limits and drone size discussed in the previous section. Unfortunately, this approach delays the discovery and use of knowledge in the captured data by a substantial amount (e.g., many tens of minutes to a few hours). Such delay may be unacceptable in use cases such as search-and-rescue or military surveillance. In this paper, we focus on approaches that aim for much smaller result latency: ideally, close to real-time.

A different approach is to offload video processing during flight over a wireless link to an edge computing node called a *cloudlet.* With this approach, even a small drone can leverage the substantial processing capability of a ground-located cloudlet, without concern for its weight, size, heat dissipation, or energy usage. Much lower result latency is now possible. However, even if cloudlet resources are viewed as "free" from the viewpoint of mobile computing, the drone consumes wireless bandwidth in transmitting video.

Today, 4G LTE offers the most plausible wide-area connectivity from a drone in flight to its associated cloudlet. The much higher bandwidths of 5G are still many years away, especially at global scale. More specialized wireless technologies for drones, such as Lightbridge 2 [25], can also be used. Regardless of specific wireless technology, the principles and techniques described in this paper apply.

Scalability, in terms of maximum number of concurrently operating drones within a 4G LTE cell becomes an important metric. In this paper we explore how the limited processing capability on a drone can be used to greatly decrease the volume of data transmitted, thus improving scalability while minimally impacting result accuracy and result latency.

Note that the uplink capacity of 500 Mbps per 4G LTE cell assumes standard cellular infrastructure that is undamaged. In natural disasters and military combat, this infrastructure may be destroyed. Emergency substitute infrastructure, such as Google and AT&T's partnership on balloon-based 4G LTE infrastructure for Puerto Rico after hurricane Maria [26], can only sustain much lower uplink bandwidth per cell, e.g. 10Mbps for the balloon-based LTE [27]. Conserving wireless bandwidth from drone video transmission then becomes even more important, and the techniques described here will be even more valuable.

Result accuracy influences a second dimension of scalability, namely the ability of one individual to supervise the result streams from many drones. The output of each video processing pipeline should only demand occasional human attention. The accuracy, sophistication, and speed of this pipeline determines the cognitive load on mission personnel for a given video stream. For example, a pipeline that has virtually no false positives or false negatives in detecting survivors will consume less supervisory human attention than a mediocre pipeline. That will allow one person to confidently supervise a large swarm that rapidly covers a large search area.

III. System Architecture

A. *Overview*

Figure 1 shows the key components of a typical mission. A swarm of drones flies over a certain coverage area. Each drone has sufficient storage on board to store all the video captured during the mission at full fidelity. During flight, the drone can transmit all or part of the video captured to a ground-based cloudlet over a wireless network. Preliminary filtering of incoming video on board the drone determines which subsets of the video stream, along with possible annotations from the filtering, are transmitted. The sophistication of this filtering depends on the computational power available on the drone, since it has to be performed at a rate that is at least loosely correlated with the video capture frame rate. Mission personnel can view the data in near real-time after processing by the cloudlet, and can take actions based on what they learn. For example, a search and rescue mission Graphical User Interface (GUI) may highlight the map coordinates of a survivor seen in the video feed. A rescue team can then be dispatched to those coordinates. As Figure 1 suggests, the coverage area of a drone swarm may span multiple 4G LTE cells. Multiple swarms may concurrently operate within a cell, and across cells. Depending on the use case, many different types of cloudlets may be used. These can range from small standalone units to one or more racks of equipment within a small edge-located building. In Fig. 1, the cloudlet is connected to the LTE base station and packets from drones are routed directly to the cloudlet without traversing the Internet backbone. While existing LTE infrastructure is more convoluted because of its legacy Evolved Packet Core, efforts are being made by industry to simplify connectivity in order to harness the benefits of edge computing [28] [29] [30]. For illustration, Figure 1 shows an Ubuntu Orange Box [31] as the cloudlet. This self-contained cluster of Xeon processors with storage and networking is a "data center in a box" that can be easily transported to a mission site.

In this paper, we focus on the computer vision processing pipeline of a single drone. While the problems of swarm management and coordination are interesting, they are outside the scope of this paper. Only two aspects of swarms are significant here. First, swarms increase the total communication volume; it is thus important that each drone be frugal in its bandwidth usage. Second, swarms increase the total cognitive load on mission personnel. The result accuracy of each video processing pipeline needs to be high.

Our focus on mission-specific video processing allows us to ignore the sensing and processing required for flight control and navigation. We assume that these more basic capabilities are provided by a separate drone subsystem. As explained in Section I, the total wireless bandwidth demand from this subsystem is negligible since we are focusing exclusively on autonomous drones. Virtually all of the bandwidth demand from such a drone comes from its transmission of mission-specific video to its cloudlet.

B. *Reducing Bandwidth Demand*

Our goal is to reduce the total volume of data transmitted from a drone to cloudlet during a mission, while preserving excellent result latency and high result accuracy. For any given event, such as the first appearance of a specific survivor in a video frame, the lowest attainable result latency is the sum of four components: (a) capture and processing delay in the drone; (b) transmission delay over the wireless link; (c) processing delay in the cloudlet; (d) reaction time of mission personnel. We assume that (d) is invariant across all the strategies that we study, and therefore omit it from further discussion. Assuming that the cloudlet is powerful enough to meet the peak processing demand of all video streams from a swarm, component (c) can also be viewed as invariant across strategies. Thus, (a) and (b) are the primary variables of interest in our study. In particular, we focus on reducing the total amount of data transmitted and study its impact on (a) and (b). While various conditions in real networks also influence the total bandwidth consumed, we take a network transparent view in this paper.

As the baseline for comparison, we use the approach of performing no processing on the drone: all video is transmitted immediately. We call this the DUMBDRONE strategy. Among all strategies, it will generate the highest volume of data during a mission. Section IV presents this baseline bandwidth demand on a suite of benchmark videos that are used for evaluation in the rest of the paper. Today's teleoperated drones essentially use the DUMBDRONE strategy.

In Sections V to VIII, we describe and evaluate four strategies to reduce bandwidth demand. We list them below with very brief descriptions, and discuss them fully in their respective sections. These strategies are not mutually exclusive, and may be combined into mission-specific optimized pipelines.

- EARLYDISCARD: Use limited processing on the drone to avoid transmitting "uninteresting" video frames.
- JUST-IN-TIME-LEARNING (JITL): Use real-time machine learning on the early part of an input video stream to adapt and improve drone processing on the later part of the video stream.
- REACHBACK: Compensate for over-zealous filtering on the drone by having the cloudlet reach back and request suppressed video segments from drone storage to discover false negatives.
- CONTEXTAWARE: Exploit unique opportunities for optimization that are only possible because of specific attributes of the current mission and video stream.

C. Key Performance Indicators

In evaluating these alternative strategies, our key metric of interest is the total volume of data (number of bytes) transmitted over the duration of a mission. Peak bandwidth demand, averaged over a short interval such as one second, is also of interest. Low values of both metrics are desirable since they indicate better scalability.

Result latency is also of interest. For all detected events, small result latency is ideal since it enables the fastest possible response. In computing the statistics of this variable (e.g., mean or standard deviation), we omit undetected events since they effectively have infinite result latency.

Precision and recall, which are the classic measures of computer vision accuracy, are also important. False negatives in the pipeline (i.e., poor recall) correspond to missed events. This can have dire consequences in the real world, such as a survivor dying because no attempt was made to rescue him. At the same time, too many false positives (i.e., poor precision) can result in cognitive overload for mission personnel. That, in turn, can lead to human errors in the mission that may also have dire consequences. In practice, a workable approach is to bias the pipeline slightly towards lower precision and higher recall. This increases cognitive load modestly, while striving to minimize missed events.

IV. DUMBDRONE STRATEGY

A. Description

As its name implies, no image processing is done on the drone in this baseline strategy. Instead, captured video is immediately written to drone storage and concurrently transmitted to the cloudlet. Result latency is very low, merely the sum of transmission delay and cloudlet processing delay.

B. Experimental Setup

To ensure experimental reproducibility, our evaluation is based on replay of a benchmark suite of pre-captured videos rather than on measurements from live drone flights. In practice, live results may diverge slightly from trace replay because of non-reproducible phenomena. These can arise, for example, from wireless propagation effects caused by varying weather conditions, or by seasonal changes in the environment such as the presence or absence of leaves on trees. In addition, variability can arise in a drone's pre-programmed flight path due to collision avoidance with moving obstacles such as birds, other drones, or aircraft.

All of the pre-captured videos in the benchmark suite are publicly accessible, and have been captured from aerial viewpoints. They characterize drone-relevant scenarios such as surveillance, search-and-rescue, and wildlife conservation that were mentioned in Section I. Figure 4 presents this benchmark suite of videos, organized into five tasks. All the tasks involve detection of tiny objects on individual frames. Task T5 additionally involves action detection, which operates on short video segments rather than individual frames. Although T2 is also nominally about action detection (moving cars), it is implemented using object detection on individual frames and then comparing the pixel coordinates of vehicles in successive frames.

C. Results

Figure 5 presents the key performance indicators on the object detection tasks T1 and T2. We use the well-labeled dataset to train and evaluate Faster-RCNN with ResNet 101. We report the precision and recall at maximum F1 score. Peak bandwidth is not shown since it is identical to average bandwidth demand for continuous video transmission. As shown earlier in Figure 3, the accuracy of this algorithm comes at the price of very high resource demand. This can only be met today by server-class hardware that is available in a cloudlet. Even on a cloudlet, the figure of 438 milliseconds of processing time per frame indicates that only a rate of two frames per second is achievable. Sustaining a higher frame rate will require striping the frames across cloudlet resources, thereby increasing resource demand considerably. Note that the results in Figure 3 were based on 1080p frames, while tasks T1 and T5 use the

Task	Detection Goal	Data Source	Data Attributes	Training Subset	Testing Subset
T1	People in scenes of daily life	Okutama Action Dataset [32]	33 videos 59842 fr 4K@30 fps	9 videos 17763 fr	6 videos 20751 fr
T2	Moving cars	Stanford Drone Dataset [33]	60 videos 522497 fr 1080p@30 fps	16 videos 179992 fr	14 videos 92378 fr Combination of test videos from each dataset.
T3	Raft in flooding scene	YouTube collec-tion [34]	11 videos 54395 fr 720p@25 fps	8 videos 43017 fr	
T4	Elephants in natural habitat	YouTube collec-tion [35]	11 videos 54203 fr 720p@25 fps	8 videos 39466 fr	
T5	Pushing or pulling Suitcases	Okutama Action Dataset	Same as T1	Same as T1	

fr = "frames"
fps = "frames per second"
No overlap between training and testing subsets of data

Figure 4. Benchmark Suite of Video Traces

Task	Total Bytes (MB)	Avg BW (Mbps)	Recall	Precision
T1	924	10.7	74%	92%
T2	2704	7.0	66%	90%

Peak bandwidth demand is same as average since video is transmitted continuously. Precision and recall are at the maximum F1 score.

Figure 5. Baseline Object Detection KPIs

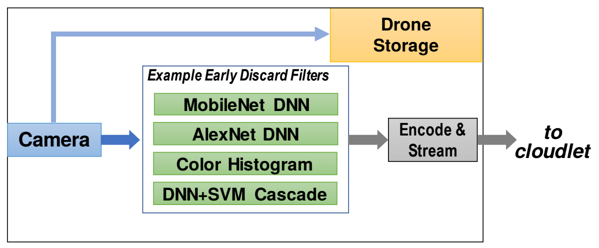


Figure 6. Drone-based Early Discard

higher resolution of 4K. This will further increase demand on cloudlet resources.

Clearly, the strategy of blindly shipping all video to the cloudlet and processing every frame is resource-intensive to the point of being impractical today. It may be acceptable as an offline processing approach in the cloud, but is unrealistic for real-time processing on cloudlets. We therefore explore an approach in which a modest amount of computation on the drone is able, with high confidence, to avoid transmitting many video frames and thereby saving wireless bandwidth as well as cloudlet processing resources. This leads us to the EarlyDiscard strategy of the next section.

V. EarlyDiscard Strategy

A. *Description*

EarlyDiscard is based on the idea of using on-board processing to filter and transmit only interesting frames in order to save bandwidth when offloading computation. Previous work [36] [37] leveraged pixel-level features and multiple sensing modalities to select interesting frames from hand-held or body-worn cameras. In this work, we explore the use of DNNs to filter frames from aerial views. The benefits of using DNNs are twofold. First, DNNs are trained and specialized for each task, resulting in their high accuracy and robustness. Second, no additional hardware is added to existing drone platforms.

Although smartphone-class hardware is incapable of supporting the most accurate object detection algorithms at full frame rate today, it is typically powerful enough to support less accurate algorithms. These *weak detectors* are typically designed for mobile platforms or were the state of the art just a few years ago. In addition, they can be biased towards high recall with only modest loss of precision. In other words, many clearly irrelevant frames can be discarded by a weak detector, without unacceptably increasing the number of relevant frames that are erroneously discarded. This asymmetry is the basis of the early discard strategy.

As shown in Figure 6, we envision a choice of weak detectors being available as early discard filters on a drone, with the specific choice of filter being mission-specific. Relative to the measurements presented in Figure 3, early discard only requires image classification: it is not necessary to know exactly where in the frame a relevant object occurs. This suggests that MobileNet would be a good choice as a weak detector. Its speed of 13 ms per frame on Jetson yields more than 75 fps. We therefore use MobileNet on the drone for early discard in our experiments.

Pre-trained classifiers for MobileNet are available today for objects such as cars, animals, human faces, human bodies, watercraft, and so on. However, these DNN classifiers have typically been trained on images that were captured from a human perspective — often by a camera held or worn by a person. A drone, however, has an aerial viewpoint and objects look rather different. To improve classification accuracy on drones, we used *transfer learning* [38] to finetune the pre-trained classifiers on small training sets of images that were captured from an aerial viewpoint. This involves initial re-training of the last DNN layer, followed by re-training of the entire network until convergence. Transfer learning enables accuracy to be improved significantly for aerial images without incurring the full cost of creating a large training set captured from an aerial viewpoint.

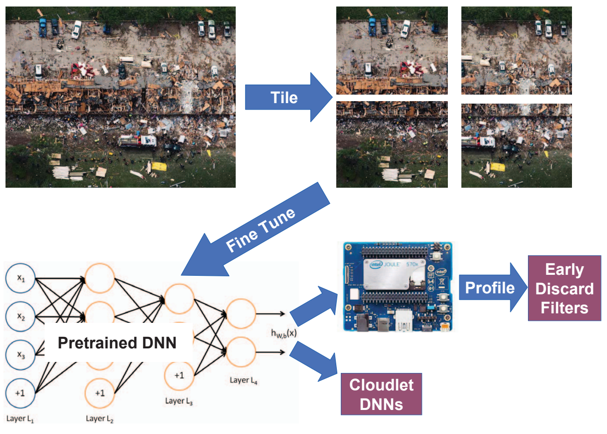


Figure 7. Tiling and DNN Fine Tuning

Drone images are typically captured from a significant height, and hence objects in such an image are small. This interacts negatively with the design of many DNNs, which first transform an input image to a fixed low resolution — for example, 224x224 pixels in MobileNet. Many important but small objects in the original image become less recognizable. It has been shown that small object size correlates with poor accuracy in DNNs [24]. To address this problem, we *tile* high resolution frames into multiple sub-frames and then perform recognition on the sub-frames. This is done offline for training, as shown in Figure 7, and also for online inference on the drone and on the cloudlet. The lowering of resolution of a sub-frame by a DNN is less harmful, since the scaling factor is smaller. Objects are represented by many more pixels in a transformed sub-frame than if the entire frame had been transformed. The price paid for tiling is increased computational demand. For example, tiling a frame into four sub-frames results in four times the classification workload.

B. Experimental Setup

Our experiments on the EarlyDiscard strategy used the same benchmark suite described in Section IV-B. We used Jetson TX2 as the drone platform. We use both frame-based and event-based metrics to evaluate the MobileNet filters.

C. Results of Early Discard Filters

EarlyDiscard is able to significantly reduce the bandwidth consumed while maintaining high result accuracy and low average delay. For three out of four tasks, the average bandwidth is reduced by a factor of ten. Below we present our results in detail.

Effects of Tiling: Tiling is used to improve the accuracy for high resolution aerial images. We used the Okutama Action Dataset, whose attributes are shown in row T1 of Figure 4, to explore the effects of tiling. For this dataset, Figure 8 shows how speed and accuracy change with tile size. Accuracy

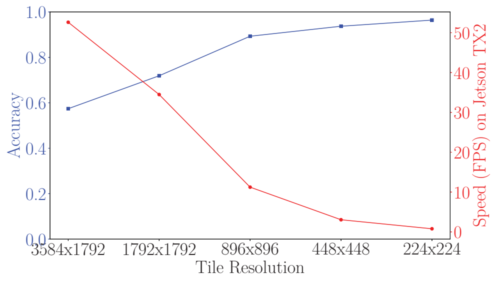


Figure 8. Speed-Accuracy Trade-off of Tiling

improves as tiles become smaller, but the sustainable frame rate drops. We group all tiles from the same frame in a single batch to leverage parallelism, so the processing does not change linearly with the number of tiles. The choice of an operating point will need to strike a balance between the speed and accuracy. In the rest of the paper, we use two tiles per frame by default.

Drone Filter Accuracy: The output of a drone filter is the probability of the current tile being "interesting." A tunable *cutoff threshold* parameter specifies the threshold for transmission to the cloudlet. All tiles, whether deemed interesting or not, are still stored in the drone storage for post-mission processing.

Figure 9 shows our results on all four tasks. Events such as detection of a raft in T3 occur in consecutive frames, all of which contain the object of interest. A correct detection of an event is defined as at least one of the consecutive frames being transmitted to the cloudlet. Blue lines in Figure 9 shows how the event recalls of drone filters for different tasks change as a function of cutoff threshold. The MobileNet DNN filter we used is able to detect all the events for T1 and T4 even at a high cutoff threshold. For T2 and T3, the majority of the events are detected. Achieving high recall on T2 and T3 (on the order of 0.95 or better) requires setting a low cutoff threshold. This leads to the possibility that many of the transmitted frames are actually uninteresting (i.e., false positives).

False negatives: As discussed earlier, false negatives are a source of concern with early discard. Once the drone drops a frame containing an important event, improved cloudlet processing cannot help. The results in the third column of Figure 10 confirm that there are no false negatives for T1 and T4 at a cutoff threshold of 0.5. For T2 and T3, lower cutoff thresholds are needed to achieve perfect recalls.

Result latency: The contribution of early discard processing to total result latency is calculated as the average time difference between the first frame in which an object occurs (i.e., first occurrence in ground truth) and the first frame containing the object that is transmitted to the backend (i.e., first detection). The results in the fourth column of Figure 10 confirm that early discard contributes little to result latency. The amounts range from 0.1 s for T1 to 12.7 s for T3. At the

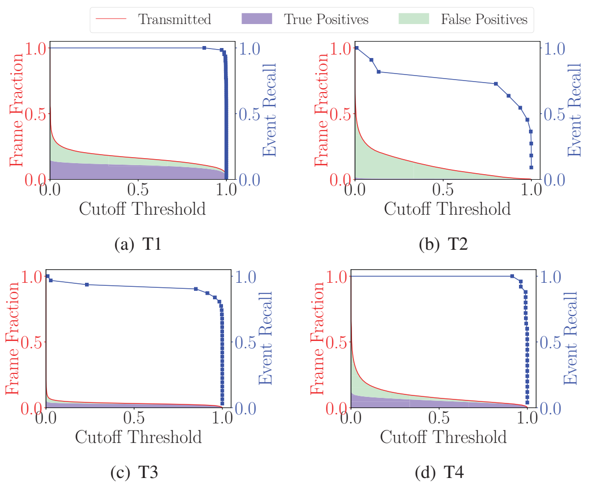


Figure 9. Where the Bandwidth Goes

	Task Total Events	Detected Events	Avg Delay (s)	Total Data (MB)	Avg B/W (Mbps)	Peak B/W (Mbps)
T1	62	100 %	0.1	441	5.10	10.7
T2	11	73 %	4.9	13	0.03	7.0
T3	31	90 %	12.7	93	0.24	7.0
T4	25	100 %	0.3	167	0.43	7.0

Figure 10. Recall, Event Latency and Bandwidth at Cutoff Threshold 0.5

timescale of human actions such as dispatching of a rescue team, these are negligible delays.

Bandwidth: Columns 5–7 of Figure 10 pertain to wireless bandwidth demand for the benchmark suite with early discard. The figures shown are based on H.264 encoding of each individual frames in the drone-cloudlet video transmission. Average bandwidth is calculated as the total data transmitted divided by mission duration. Comparing column 5 of Figure 10 with column 2 of Figure 5, we see that all videos in the benchmark suite are benefited by early discard (Note T3 and T4 have the same test dataset as T2). For T2, T3, and T4, the bandwidth is reduced by more than 10x. The amount of benefit is greatest for rare events (T2 and T3). When events are rare, the drone can drop many frames.

Figure 9 provides deeper insight into the effectiveness of cutoff-threshold on event recall. It also shows how many true positives (violet) and false positives (aqua) are transmitted. Ideally, the aqua section should be zero. However for T2, most frames transmitted are false positives, indicating the early discard filter has low precision. The other tasks exhibit far fewer false positives. This suggests that the opportunity exists for significant bandwidth savings if precision could be further improved, without hurting recall.

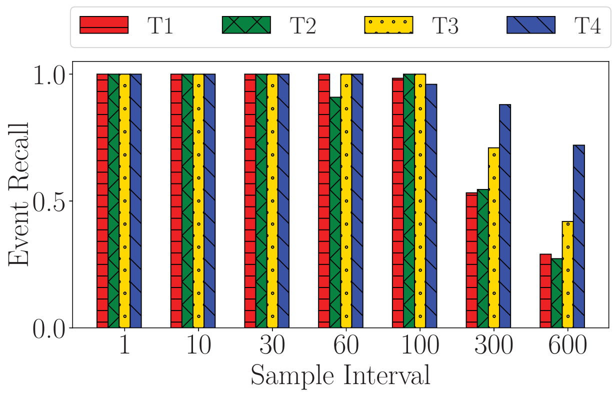


Figure 11. Event Recall at Different Sampling Intervals

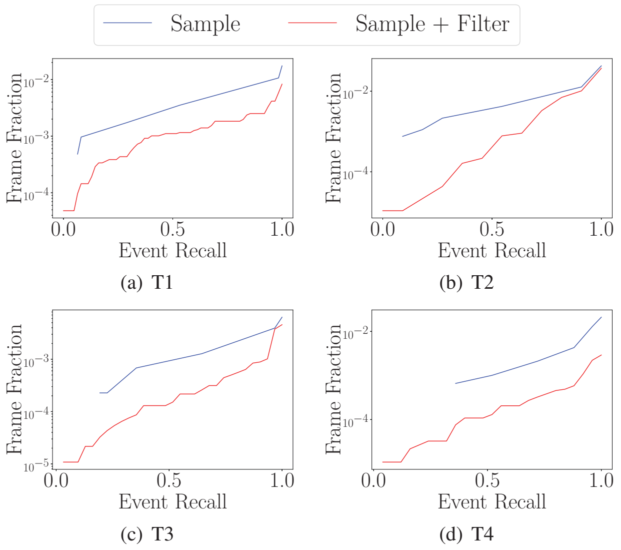


Figure 12. Sample with Early Discard. Note the log scale on y-axis.

D. *Use of Sampling*

Given the relatively low precision of the weak detectors, a significant number of false positives are transmitted. Furthermore, the occurrence of an object will likely last through many frames, so true positives are also often redundant for simple detection tasks. Both of these result in excessive consumption of precious bandwidth. This suggests that simply restricting the number of transmitted frames by sampling may help reduce bandwidth consumption.

Figure 11 shows the effects of sending a sample of frames from the drone, without any content-based filtering. Based on these results, we can reduce the frames sent as little as one per second and still get adequate recall at the cloudlet. Note that this result is very sensitive to the actual duration of the events in the videos. For the detection tasks outlined here, most of the events (e.g., presences of a particular elephant) last for many seconds (100's of frames), so such sparse sampling does not hurt recall. However, if the events

JPEG Frame Sequence (MB)	H264 High Quality (MB)	H264 Medium Quality (MB)	H264 Low Quality (MB)
5823	3549	1833	147

H264 high quality uses Constant Rate Factor (CRF) 23. Medium uses CRF 28 and low uses 40 [39].

Figure 13. Test Dataset Size With Different Encoding Settings

were of short duration, e.g., just a few frames long, then this method would be less effective, as sampling may lead to many missed events (false negatives).

Can we use content-based filtering along with sampling to further reduce bandwidth consumption? Figure 12 shows results when running early discard on a sample of the frames. This shows that for the same recall, we can reduce the bandwidth consumed by another factor of 5 on average over sampling alone. This effective combination can reduce the average bandwidth consumed for our test videos to just a few hundred kilobits per second. Furthermore, more processing time is available per processed frame, allowing more sophisticated algorithms to be employed, or to allow smaller tiles to be used, improving accuracy of early discard.

One case where sampling is not an effective solution is when all frames containing an object need to be sent to the cloudlet for some form of activity or behavior analysis from a complete video sequence (as may be needed for task T5). In this case, bandwidth will not reduce much, as all frames in the event sequence must be sent. However, the processing time benefits of sampling may still be exploited, provided all frames in a sample interval are transmitted on a match.

E. Effects of Video Encoding

One advantage of the DumbDrone strategy is that since all frames are transmitted, one can use a modern video encoding to reduce transmission bandwidth. With early discard, only a subset of disparate frames are sent. These will likely need to be individually compressed images, rather than a video stream. How much does the switch from video to individual frames affect bandwidth?

In theory, this can be a significant impact. Video encoders leverage the similarity between consecutive frames, and model motion to efficiently encode the information across a set of frames. Image compression can only exploit similarity within a frame, and cannot efficiently reduce number of bits needed to encode redundant content across frames. To evaluate this difference, we start with extracted JPEG frame sequences of our video data set. We encode the frame sequence with different H.264 settings. Figure 13 compares the size of frame sequences in JPEG and the encoded video file sizes. We see only about 3x difference in the data size for the medium quality. We can increase the compression (at the expense of quality) very easily, and are able to reduce the video data rate by another order of magnitude before quality degrades catastrophically.

However, this compression does affect analytics. Even at medium quality level, visible compression artifacts, blurring, and motion distortions begin to appear. Initial experiments analyzing compressed videos show that these distortions do have a negative impact on accuracy of analytics. Using average precision analysis, a standard method to evaluate accuracy, we see that the most accurate model (Faster-RCNN ResNet101) on low quality videos performs similarly to the less accurate model (Faster-RCNN InceptionV2) on high quality JPEG images. This negates the benefits of using the state-of-art models.

In this system, we pay a penalty of sending frames instead of a compressed low quality video stream. This overhead (approximately 30x) is compensated by the 100x reduction in frames transmitted due to sampling with early discard. In addition, the selective frame transmission preserves the accuracy of the state-of-art detection techniques.

Finally, one other option is to treat the set of disparate frames as a sequence and employ video encoding at high quality. This can ultimately eliminate the per frame overhead while maintaining accuracy. However, this will require a complex setup with both low-latency encoders and decoders, which can generate output data corresponding to a frame as soon as input data is ingested, with no buffering, and can wait arbitrarily long for additional frame data to arrive.

For the experiments in the rest of the paper, we only account for the fraction of frames transmitted, rather than the choice of specific encoding methods used for those frames.

VI. Just-in-time-Learning Strategy

A. Description

Just-in-time-learning (JITL) tunes the drone pipeline to the characteristics of the current mission in order to reduce transmitted false positives from the drone, and therefore reduce wasted bandwidth. It is inspired by the cascade architecture from the computer vision community [40], but is different in construction. A JITL filter is a cheap cascade filter that distinguishes between the EarlyDiscard DNN's *true positives* (frames that are actually interesting) and *false positives* (frames that are wrongly considered interesting). Specifically, when a frame is reported as positive by EarlyDiscard, it is then passed through a JITL filter. If the JITL filter reports negative, the frame is regarded as a false positive and will not be sent. Ideally, all *true positives* from EarlyDiscard are marked *positive* by the JITL filter, and all *false positives* from EarlyDiscard are marked *negative*. Frames dropped by EarlyDiscard are not processed by the JITL filter, so this approach can only serve to improve precision, but not recall.

Periodically during a drone mission, a JITL filter is trained on the cloudlet using the frames transmitted from the drone. The frames received on the cloudlet are predicted positive by the EarlyDiscard filter. The cloudlet, with more processing power, is able to run more accurate DNNs to identify true positives and false positives. Using this information, a small and lightweight JITL filter is trained to distinguish true positives and false positives of EarlyDiscard filters. These JITL filters are then pushed to the drone to run as a cascade filter after the EarlyDiscard DNN.

True/false positive frames have high temporal locality throughout a drone mission. The JITL filter is expected to pick up the features that confused the EarlyDiscard DNN in the immediate past and improve the pipeline's accuracy in the near future. These features are usually specific to the current flight, and may be affected by terrain, shades, object colors, and particular shapes or background textures.

JITL can be used with EarlyDiscard DNNs of different cutoff probabilities to strike different trade-offs. In a bandwidth-favored setting, JITL can work with an aggressively selective EarlyDiscard DNN to further reduce wasted bandwidth. In a recall-favored setting, JITL can be used with a lower-cutoff DNN to preserve recall.

In our implementation, we use a linear support vector machine (SVM) [41] as the JITL filter. Linear SVM has several advantages: 1) short training time in the order of seconds; 2) fast inference; 3) only requires a few training examples; 3) small in size to transmit, usually on the order of 50KB in our experiments. The input features to the JITL SVM filter are the image features extracted by the EarlyDiscard DNN filter. In our case, since we are using MobileNet as our EarlyDiscard filter, they are the 1024-dimensional vector elements from the second last layer of MobileNet. This vector, also called "bottleneck values" or "transfer values" captures high-level features that represents the content of an image. Note that the availability of such image feature vector is not tied to a particular image classification DNN nor unique to MobileNet. Most image classification DNNs can be used as a feature extractor in this way.

B. Experimental Setup

We used Jetson TX2 as our drone platform and evaluated the JITL strategy on four tasks, T1 to T4. For the test videos in each task, we began with the EarlyDiscard filter only and gradually trained and deployed JITL filters. Specifically, every ten seconds, we trained an SVM using the frames transmitted from the drone and the ground-truth labels for these frames. In a real deployment, the frames would be marked as true positives or false positives by an accurate DNN running on the cloudlet since ground-truth labels are not available. In our experiments, we used ground-truth labels to control variables and remove the effect of imperfect prediction of DNN models running on the cloudlet. In addition, we used the true and false positives from all previous intervals, not just the last ten seconds when training the SVM. The SVM, once trained, is used as a cascade filter running after the EarlyDiscard filter on the drone to predict whether the output of the EarlyDiscard filter is correct or not. For a frame, if the EarlyDiscard filter predicts it to be interesting, but the JITL filter predicts the EarlyDiscard filter is wrong, it would not be transmitted to the cloudlet. In other words, following two criteria need to be satisfied for a frame to be transmitted to the cloudlet: 1) EarlyDiscard filter predicts it to be interesting 2) JITL filter predicts the EarlyDiscard filter is correct on this frame.

C. Results

From our experiments, JITL is able to filter out more than 15% of remaining frames after EarlyDiscard without loss of event recall for three of four tasks. Figure 14 details the fraction of frames saved by JITL. The x-axis presents event recall. Y-axis represents the fraction of total frames. The blue region presents the achievable fraction of frames by EarlyDiscard. The orange region shows the additional savings using JITL. For T1, T3, and T4, at the highest event recall, JITL filters out more than 15% of remaining frames. This shows that JITL is effective at reducing the false positives thus improving the precision of the drone filter. However, occasionally, JITL predicts wrongly and removes true positives. For example, for T2, JITL does not achieve a perfect event recall. This is due to shorter event duration in T2, which results in fewer positive training examples to learn from. Depending on tasks, getting enough positive training examples for JITL could be difficult, especially when events are short or occurrences are few. To overcome this problem in practice, techniques such as synthetic data generation [42] could be explored to synthesize true positives from the background of the current flight.

VII. ReachBack Strategy

Reachback strategy is designed for drones to take advantage of their storage space. Today, enabled by inexpensive storage, tele-operated drones are designed to store the entire video footage from a flight on-board. In our live video analytics setup, when portions of video feeds are streamed back to cloudlets for analysis, the complete videos are stored locally on the drones as authoritative sources. Furthermore, the limited processing on the drone can generally only serve to down-select the frames that may be useful to downstream processing at the cloudlet. Our strategy is to tune the drone processing in favor of recall, so that interesting events and objects are not missed. As these algorithms are not perfect, it is still possible that a few critical frames are not transmitted. These frames will not be discarded, however, and will be stored safely on board the drone. The essence of the reachback strategy is to allow the cloudlet to fetch additional frames from the drone storage when needed to complete analysis.

This mechanism is particularly useful in the context of activity detection, where a consecutive set of frames are needed to accurately identify the actions in the scene, e.g.,

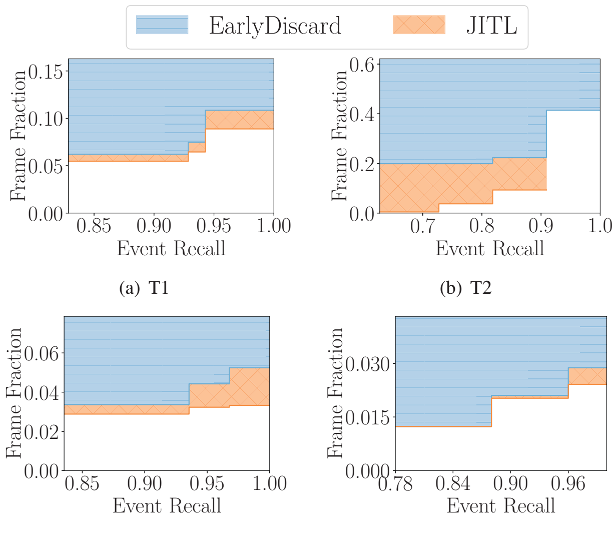


Figure 14. JITL Fraction of Frames under Different Event Recall

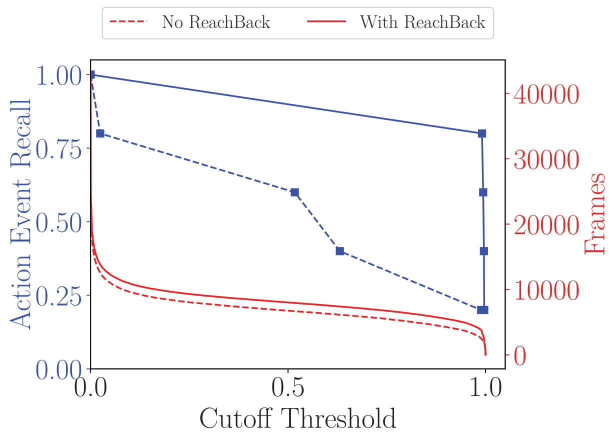


Figure 15. Effect of Reachback on bandwidth and recall

whether an elephant is flapping its ears. As early discard at the drone is performed frame by frame, only a scattered subset of the desired frames may be initially delivered to the cloudlet. If analysis on these frames indicate the event in question may have occurred, the cloudlet will request that the missing frames be sent from the drone. With the complete frame sequence, the incidence of the action can be accurately determined. We note that the reachback mechanism is useful in both an automated analytics context, as well as with a human operator in the loop. For example, a human observer may find the appearance or pose of an individual in a frame to be unusual, and can request that the complete video sequence preceding the suspicious frame to be transmitted from the drone.

We evaluate the idea of reachback to a drone using a simple activity inference task (T5). Here, the goal is to identify instances of people pushing or pulling large suitcases. We use the test videos from the Okutama dataset, in which there are 5 instances of a person pushing or pulling a suitcase-like object. We use the EarlyDiscard filter trained for T1 as the EarlyDiscard filter in this experiment. As a rough estimate, we assume 80% of event frames are needed for a general activity detector to successfully identify an action of pushing or pulling an object. In general, early discard alone cannot deliver enough frames. On the cloudlet, we use the successful detection of a person and a nearby suitcase-like object to trigger reachback. Through manual inspection of the test videos, we identify seven separate sequences where the trigger condition will be satisfied. This manual inspection could be replaced by activity recognition algorithms when available. We use reachback to retrieve temporally nearby frames that are missing, until the event has ended. We compare action event detection accuracy and bandwidth usage with and without the reachback mechanism.

Figure 15 shows that reachback can significantly improve the event recall with a marginal increase in the bandwidth usage. The dashed lines are baseline with only early discard, while the solid lines are for the system with the reachback mechanism. The blue lines indicate action event recall, based on the accuracy model where the action is detected if at least 80% of the event frames are seen. The red lines show number of frames transmitted. As we can see, with just a marginal increase in bandwidth, event recall can be greatly improved with reachback for all early discard cut-off thresholds. The bandwidth increase is due to the frames transmitted from the drone in response to the seven reachbacks (5 true positives, 2 false positives) triggered for this video sequence. Note that if the early discard is extremely aggressive, then so few frames reach the cloudlet that the reachback criteria may not be satisfied, causing reduction in event recalls.

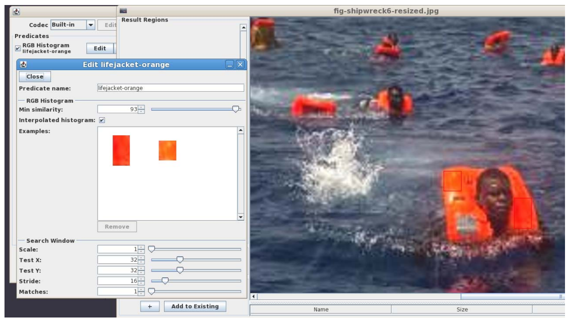
VIII. Context-Aware Strategy

A. Description

The essence of this approach is to leverage unique attributes of the current mission to improve the speed and accuracy of video processing on the drone. By definition, this approach is ad hoc in character and therefore hard to generalize or automate. However, the wins can be significant.

As an illustrative example, consider searching for survivors in the ocean after a shipwreck. Suppose the standard approach for detecting survivors in search and rescue missions involves detection of human faces and bodies (similar to T1), or distinctive actions such as waving. These are used as the basis of early discard at the beginning of the mission. During the mission, as personnel review results that are presented to them after cloudlet processing, they

(a) Motivating Image



Codec: Built-in

Predicates: ✓ RGB Histogram lifejacket-orange

Edit lifejacket-orange

Predicate name: lifejacket-orange

RGB Histogram

Min similarity: 93

Interpolated histogram: ✓

Examples: 🟥 🟧

Search Window

Scale: 1

Test X: 32

Test Y: 32

Stride: 16

Matches: 1

(b) GUI to Define Color Filter

Figure 16. Example of Context-Aware Strategy

notice that all survivors are wearing flotation devices (life jackets) that have a distinctive color. Against the blue-green background of the ocean, detecting this color is a fast, accurate, and computationally cheap way of detecting survivors. Figure 16(a) gives an example of such a scene. However this optimization is unique to this mission. On a different mission that also involves a shipwreck, the life jackets may be of a different color. Or, because of the late evening timing of the mission and the consequent low angle of the sun, too many false positives may arise from reflections off the water if reddish-orange colors are used as the basis of early discard.

By the classic metrics of machine learning, the use of such heuristics is viewed as "overfitting" and therefore something to be avoided at all costs. Yet, in practical terms and in the narrow context of this mission, the heuristic offers many advantages without compromising accuracy. On some missions, the heuristic may even be more accurate than a DNN. We consider it important to allow mission personnel to take advantage of such context-aware optimizations.

As shown earlier in Figure 6, we support many pre-installed filters on the drone to implement context-aware early discard. These filters are parameterized, and the parameters (such as the specific color of the life jackets) can be supplied at runtime over the wireless link. As discussed earlier, the default filter is the union of a set of MobileNet DNNs that have each been trained on a specific type of object (e.g., human face, human body and raft). Other filters can be activated at runtime.

	Precision using DNN (%)	Precision using color filter (%)	Recall (%)
Video 1	92.4	95.3	89.1
Video 2	51.9	76.1	90.0
Video 3	41.3	84.3	88.6

(a) Accuracy

	Jetson		Joule		Nexus 6	
	DNN	Color	DNN	Color	DNN	Color
Video 1	13	6.2	37	9.8	352	27.5
Video 2		6.3		9.7		26.3
Video 3		9.5		12.3		36.1

(b) Processing time (ms)

Figure 17. Detection Results on T3 Using Color Filters

Mission personnel can specify parameters to filters by example, as shown in the GUI in Figure 16(b). This is done by drawing bounding boxes around the relevant parts of images that were presented after cloudlet processing. Filters can be selectively activated or deactivated, and combined to generate complex search predicates. They can be used on the drone both for early discard of future video, as well as re-examination of stored video. When the accuracy of the uploaded filter is better than that of the default DNN filter, a re-examination of stored video can yield hits that were missed earlier. These new hits can be downloaded to the cloudlet for further processing. The context-aware filter is thus being used both for reachback from already-captured video, as well as for early discard on future video.

B. Experimental Setup

To demonstrate the effectiveness of this strategy, we apply it using a simple color filter for T3. In each raft search video, we randomly pick a frame that contains a raft (true positive), and obtain the color of the most distinctive region of the raft. Using the hue, saturation, and value (HSV) color space attributes of this region, we apply a color filter to all the other frames of the video. If a significantly large area of a frame passes this filter, the frame is marked as positive. Otherwise, it is marked as negative.

C. Results

Figure 17 shows using this approach can both improve accuracy and reduce computation on three representative test videos in T3. For all three videos, the precision using a color filter is better than the precision using a DNN. The difference is modest for Video 1, but considerable for Video 2 and Video 3. In other words, the context aware approach is consistently more accurate. This improvement in accuracy does not come at a sacrifice in speed. On the contrary, Figure 17(b) shows that the the color filter is significantly faster than the DNN, ranging from 2x to over an order of magnitude faster depending on the device and data set. These results show the high value of using context-aware knowledge. What the DNN provides is generality, combined with reasonable accuracy. At the beginning of a mission, when little is known about the context-specific search attributes of the target, the DNN is the only choice. As the mission progresses, the early results may hint at the attributes of a highly effective and cheap context-aware filter.

IX. Discussion

The techniques presented in Sections V to VIII are not mutually exclusive. Instead, they are designed to be used collectively to form a mission-specific pipeline. The EarlyDiscard technique employs on-board filters to select interesting frames and suppress the transmission of mundane frames to save bandwidth. In particular, cheap yet effective DNN filters are trained offline to fully leverage the large quantity of training data and the high learning capacities of DNNs. Building on top of EarlyDiscard, JITL adapts an EarlyDiscard filter to a specific mission environment online. Throughout a flight, JITL continuously evaluates the EarlyDiscard filter and reduces the number of false positives by predicting whether an EaryDiscard decision is made correctly. These two techniques together reduce the total number of unnecessary frames transmitted. In addition, some missions need consecutive frames instead of individual images to do tasks such as activity recognition. Reachback compensates for these scenarios when EarlyDiscard and JITL are deployed. Once the cloudlet identifies an interesting frame from the data sent back by the drone, nearby frames are pulled from storage on the drone. Furthermore, either an algorithm or a person in the loop can determine when to trigger reachback. Besides reachback, the person in the loop may also identify unique characteristics to create more effective context-aware filters to increase accuracy and reduce on-board computation.

X. Related Work

Interest in drones has exploded in the recent past, both in industry and in the research community. Gartner [43] estimates that the global market revenue for drones will exceed $6 billion in 2017, and will grow to exceed $11 billion in 2020. The research literature also reflects growing interest in drones. Bregu et al. [13] explored how the characteristics of existing control logic and hardware could be used to create a notion of reactive control of drones. Gowda et al. [44] showed how the orientation of drones could be tracked using multiple GPS receivers. Mao et al. [45] described an indoor system in which a drone follows a user and records videos.

The work presented in this paper is disjoint from these previous drone-centric research efforts. Our focus is on reducing wireless transmission for live video from autonomous drones in use cases such as search and rescue, surveillance, and wildlife conservation. Wang et al. [8] shares our concern for wireless bandwidth, but focuses on coordinating a network of drones to capture and broadcast live sport event. In addition, Wang et al [10] explored adaptive video streaming with drones using content-based compression and video rate adaptation. While we share their inspiration, our work leverages characteristics of DNNs and explore human-in-the loop to enable mission-specific optimization strategies including reachback and context-awareness.

Much previous work on static camera networks and video analytics systems explored efficient use of compute and network resources at scale. Zhang et al. [46] studied resource-quality trade-off under result latency constraints in video analytics systems. Kang et al. [47] worked on optimizing DNN queries over videos at scale. While they focus on supporting a large number of computer vision workload, our work optimizes for the first hop wireless bandwidth. In addition, Zhang et al. [9] designed a wireless distributed surveillance system that supports a large geographical area through frame selection and content-aware traffic scheduling. In contrast, our work uses drone moving cameras. We explore techniques that tolerate changing scenes in video feeds and strategies that can leverage the human operator.

Some previous work on computer vision in mobile settings has relevance to aspects of our system design. Chen et al. [48] explore how continuous real-time object recognition can be done on mobile devices. They meet their design goals by combining expensive object detection with computationally cheap object tracking. Although we do not use object tracking in our work, we share the resource concerns that motivate that work. Naderiparizi et al. [37] describe a programmable early-discard camera architecture for continuous mobile vision. Our work shares their emphasis on early discard, but differs in all other aspects. In fact, our work can be viewed as complementing that work: their programmable early-discard camera would be an excellent choice for our drones. Lastly, Hu et al [36] have investigated the approach of using lightweight computation on a mobile device to improve the overall bandwidth efficiency of a computer vision pipeline that offloads computation to the edge. We share their concern for wireless bandwidth, and their use of early discard using inexpensive algorithms on the mobile device. However, their work is not in a drone setting and has no counterpart to just-in-time learning, reachback, or context-aware discard described in our work.

XI. Conclusion

The emergence of autonomous drones has the potential to transform many domains. Today, progress is clouded by regulatory and political uncertainty surrounding the use of drones. We are confident, however, that these are temporary inhibitors. This work looks ahead to a future when the use of video sensing on autonomous drones is widespread, both in day to day activities as well as in emergency situations.

In this paper, we address several difficult mobile computing challenges that arise in performing real-time video analytics on small autonomous drones. These challenges lie at the intersection of wireless bandwidth, processing capacity, result accuracy, and timeliness of results. To address these challenges, we have developed an adaptive computer vision pipeline for search tasks in domains such as search-and-rescue, surveillance, and wildlife conservation. We explore an early discard strategy to selectively send the most interesting frames and reduce precious bandwidth between the drone and a ground-based cloudlet. We propose additional strategies including just-in-time learning, reachback, and context-based filtering to further improve bandwidth efficiency. Our experimental results show that this judicious combination of drone-based processing and edge-based processing can save substantial wireless bandwidth and thus improve scalability, without compromising result accuracy or result latency. We believe such a drone architecture can greatly improve the scalability of search in terms of number of concurrent drones in flight and in terms of the amount of operator attention needed to monitor a swarm of drones.

Acknowledgements

We wish to thank our shepherd, Sanjay Rao, and the anonymous reviewers for their guidance in strengthening this paper. This research was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0051 and by the National Science Foundation (NSF) under grant number CNS-1518865. Additional support was provided by Intel, Vodafone, Deutsche Telekom, Verizon, Crown Castle, NTT, and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view(s) of their employers or the above-mentioned funding sources.

References

[1] Hulu, "Internet speed requirements for streaming HD and 4K Ultra HD," https://help.hulu.com/en-us/requirements-for-hd, 2017, Last accessed: May 16, 2017.

[2] LteWorld, "LTE Advanced: Evolution of LTE," http://lteworld.org/blog/lte-advanced-evolution-lte, August 2009, Last accessed: Jan 11, 2016.

[3] C. Wiltz, "How an Autonomous Drone Flies With Deep Learning," https://www.designnews.com/electronics-test/how-autonomous-drone-flies-deep-learning/172264901156787, May 2017, Last accessed: Sept 12, 2018.

[4] D. Martin, "New generation of drones set to revolutionize warfare," https://www.cbsnews.com/news/60-minutes-autonomous-drones-set-to-revolutionize-military-technology-2/, August 2017, Last accessed: Sept 12, 2018.

[5] E. N. Barmpounakis, E. I. Vlahogianni, and J. C. Golias, "Unmanned aerial aircraft systems for transportation engineering: Current practice and future challenges," *International Journal of Transportation Science and Technology*, vol. 5, pp. 111–122, 2016.

[6] J. Bateman, "China's Launching Drones to Fight Back Against Earthquakes," *Wired*, January 2017.

[7] D. Hambling, "The next era of drones will be defined by 'swarms'," http://www.bbc.com/future/story/20170425-were-entering-the-next-era-of-drones, April 2017.

[8] X. Wang, A. Chowdhery, and M. Chiang, "Networked drone cameras for sports streaming," in *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 308–318.

[9] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 2015, pp. 426–438.

[10] X. Wang, A. Chowdhery, and M. Chiang, "SkyEyes: adaptive video streaming from UAVs," in *Proceedings of the 3rd Workshop on Hot Topics in Wireless*. ACM, 2016.

[11] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, Ottawa, Canada, 1996.

[12] Netflix Help Center, "How can I control how much data Netflix uses?" 2016, Last accessed: Sept 12, 2018.

[13] E. Bregu, N. Casamassima, D. Cantoni, L. Mottola, and K. Whitehouse, "Reactive Control of Autonomous Drones," in *Proceedings of MobiSys 2016*, Singapore, June 2016.

[14] D. Hardawar, "Intel's Joule is its most powerful dev kit yet," https://www.engadget.com/2016/08/16/intels-joule-is-its-most-powerful-dev-kit-yet/, August 2016, Last accessed: Sept 12, 2018.

[15] NVIDIA, "The Most Advanced Platform for AI at the Edge," http://www.nvidia.com/object/embedded-systems.html, 2017, Last accessed: Sept 12, 2018.

[16] Intel, "Intel Aero Ready to Fly Drone," https://www.intel.com/content/www/us/en/products/drones/aero-ready-to-fly.html, 2018, Last accessed: Sept 12, 2018.

[17] NVIDIA, "TEAL Drone," https://developer.nvidia.com/embedded/community/reference-platforms/teal-drone, 2017, Last accessed: Sept 12, 2018.

[18] X. Zhang, Y. Wang, and W. Shi, "pcamp: Performance comparison of machine learning packages on the edges," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge'18)*, 2018.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[24] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[25] DJI, Inc., "Lightbridge 2 — Professional Quality Live Streaming From the Sky," https://www.dji.com/lightbridge-2, 2017, Last accessed: November 19, 2017.

[26] J. Morse, "Alphabet officially flips on Project Loon in Puerto Rico," http://mashable.com/2017/10/20/puerto-rico-project-loon-internet, October 2017, Last accessed: Sept 12, 2018.

[27] V. Sankaran, "Google Xs ambitious Loon and Wing projects graduate into independent companies," https://thenextweb.com/google/2018/07/12/google-xs-ambitious-loon-and-wing-projects-graduate-into-independent-companies, July 2018, Last accessed: Sept 12, 2018.

[28] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[29] Open Edge Computing Initiative, "Living Edge Lab," http://openedgecomputing.org/lel.html, 2018, Last accessed: Sept 12, 2018.

[30] Saguna Networks, "Saguna open-ran mec platform," https://www.saguna.net/saguna-open-ran/, Last accessed: August 28, 2018.

[31] S. J. Vaughan-Nichols, "Canonical's cloud-in-a-box: The ubuntu orange box," https://www.zdnet.com/article/canonicals-cloud-in-a-box-the-ubuntu-orange-box/, 2015, Last accessed: August 28, 2018.

[32] M. Barekatain, M. Martí, H.-F. Shih, S. Murray, K. Nakayama, Y. Matsuo, and H. Prendinger, "Okutama-action: An aerial view video dataset for concurrent human action detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 2153–2160.

[33] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European Conference on Computer Vision*, 2016.

[34] "YouTube Collection 1," https://www.dropbox.com/sh/zksp1pzc1ix5hlw/AAB3HEhx-yLAJVR1Q3HnFpsWa?dl=0, 2017.

[35] "YouTube Collection 2," https://www.dropbox.com/sh/3uly2qqwbzjasaa/AABiWSzPD-5uzmvCy3meqPKma?dl=0, 2017.

[36] W. Hu, B. Amos, Z. Chen, K. Ha, W. Richter, P. Pillai, B. Gilbert, J. Harkes, and M. Satyanarayanan, "The Case for Offload Shaping," in *Proceedings of HotMobile 2015*, 2015.

[37] S. Naderiparizi, P. Zhang, M. Philipose, B. Priyantha, J. Liu, and D. Ganesan, "Glimpse: A Programmable Early-Discard Camera Architecture for Continuous Mobile Vision," in *Proceedings of MobiSys 2017*, June 2017.

[38] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[39] L. Merritt and R. Vanam, "Improved rate control and motion estimation for h. 264 encoder," in *IEEE International Conference on Image Processing*, 2007.

[40] P. Viola and M. Jones, "Robust Real-time Object Detection," in *International Journal of Computer Vision*, 2001.

[41] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics, 2001, vol. 1, no. 10.

[42] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *The IEEE international conference on computer vision (ICCV)*, 2017.

[43] Gartner, "Gartner Says Almost 3 Million Personal and Commercial Drones Will Be Shipped in 2017," February 2017, Last accessed: Sept 12, 2018.

[44] M. Gowda, J. Manweiler, A. Dhekne, R. R. Choudhury, and J. D. Weisz, "Tracking Drone Orientation with Multiple GPS Receivers," in *Proceedings of MobiCom 2016*, 2016.

[45] W. Mao, Z. Zhang, L. Qiu, J. He, Y. Cui, and S. Yun, "Indoor Follow Me Drone," in *Proceedings of MobiSys 2017*, June 2017.

[46] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance." in *NSDI*, vol. 9, 2017, p. 1.

[47] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: optimizing neural network queries over video at scale," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1586–1597, 2017.

[48] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices," in *Proceedings of ACM SenSys*, 2015.