

September 26, 2019
DRAFT

Scaling Wearable Cognitive Assistance

Junjue Wang
junjuew@cs.cmu.edu

June 2018

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Mahadev Satyanarayanan (Satya) (Chair)
Daniel Siewiorek
Martial Hebert
Roberta Klatzky
Padmanabhan Pillai (Intel Labs)

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

Copyright © 2018 Junjue Wang
junjuew@cs.cmu.edu

September 26, 2019
DRAFT

Keywords: Wearable Cognitive Assistance, Edge Computing, Cloudlet, Scalability

Contents

1	Introduction	1
1.1	Thesis Statement	3
1.2	Thesis Overview	3
2	Background	5
2.1	Edge Computing	5
2.2	Gabriel Platform	7
2.3	Example Gabriel Applications	9
2.3.1	RibLoc Application	9
2.3.2	DiskTray Application	9
2.3.3	Model Car Assembly Application	9
2.4	Application Latency Bounds	9
3	Application-Agnostic Techniques to Reduce Network Transmission	13
3.1	EARLYDISCARD Strategy	13
3.1.1	Description	13
3.1.2	Experimental Setup	15
3.1.3	Results of Early Discard Filters	15
3.1.4	Use of Sampling	16
3.1.5	Effects of Video Encoding	20
3.2	JUST-IN-TIME-LEARNING Strategy To Improve Early Discard	21
3.2.1	Description	21
3.2.2	Experimental Setup	22
3.2.3	Results	23

3.3	Evaluation	23
3.4	Discussion	23
4	Conclusion and Future Work	25
	Bibliography	27

Chapter 1

Introduction

It has been a long endeavour to augment human cognition with machine intelligence. As early as in 1945, Vannevar Bush envisioned a machine Memex that provides "enlarged intimate supplement to one's memory" and can be "consulted with exceeding speed and flexibility" in the seminal article *As We May Think* [5]. This vision has been brought closer to reality by years of research in computing hardware, artificial intelligence, and human-computer interaction. In late 90s to early 2000s, Smailagic et al. [41] [42] [40] created prototypes of wearable computers to assist several cognitive tasks, for example, displaying inspection manuals in a head-up screen to facilitate aircraft maintenance. Around the same time, Loomis et al. [25] [24] explored using computers carried in a backpack to help the blind navigate through auditory cues. Davis et al. [9] [8] developed a context-sensitive intelligent visitor guide leveraging hand-portable multimedia systems. While these research work pioneered cognitive assistance and its related fields, their robustness and functionality are limited by the technologies of their time.

More recently, as underlying technologies experience significant advancement, a new genre of applications, Wearable Cognitive Assistance (WCA) [15] [6], has emerged that pushes the boundaries of augmented cognition. WCA applications continuously process data from body-worn sensors and provide just-in-time guidance to help a user complete a specific task. For example, an IKEA Lamp assistant [6] has been built to assist the assembly of a table lamp. To use the application, a user wears a head-mounted smart glass that continuously captures her actions and surroundings from a first-person viewpoint. In real-time, the camera stream is analyzed to identify the state of the assembly. Audiovisual instructions are generated based on the detected state. The instructions either demonstrate a subsequent procedure or alert and correct a mistake.

Since its conceptualization in 2004 [34], WCA has attracted many research interests from both academia and industry. The building blocks for its vision came into place by 2014, enabling the first implementation of this concept in *Gabriel* [15]. In 2017, Chen et al [7] described a number of applications of this genre, quantified their latency requirements, and profiled the end-to-end latencies of their implementations. In late 2017, SEMATECH and DARPA jointly funded \$27.5 million of research on such applications [29, 43]. At the Mobile World Congress in February 2018, wearable cognitive assistance was the focus of an entire session [31]. For AI-based military use cases, this class of applications is the centerpiece of "Battlefield 2.0" [10]. By mid-

2019, WCA was being viewed as a prime source of “killer apps” for edge computing [35, 39].

Different from previous research efforts, the design goals of WCA advance the frontier of mobile computing in multiple aspects. First, wearable devices, particularly head-mounted smart glasses, are used to reduce the discomfort caused by carrying a bulky computation device. Users are freed from holding a smartphone and therefore able to interact with the physical world using both hands. The convenience of this interaction model comes at the cost of constrained computation resources. The small form-factor of smart glasses significantly limits their onboard computation capability due to size, cooling, and battery life reasons. Second, placed at the center of computation is the unstructured high-dimensional image and video data. Only these data types can satisfy the need to extract rich semantic information to identify the progress and mistakes a user makes. Furthermore, state-of-art computer vision algorithms used to analyze image data are both compute-intensive and challenging to develop. Third, many cognitive assistants give real-time feedback to users and have stringent end-to-end latency requirements. An instruction that arrives too late often provides no value and may even confuse or annoy users. This latency-sensitivity further increases their high demands of system resource and optimizations.

To meet the latency and the compute requirements, previous research leverages edge computing and offloads computation to a cloudlet. A cloudlet [37] is a small data-center located at the edge of the Internet, one wireless hop away from users. Researchers have developed an application framework for wearable cognitive assistance, named Gabriel, that leverages cloudlets, optimizes for end-to-end latency, and eases application development [6] [15] [7]. On top of Gabriel, several prototype applications have been built, such as Ping-Pong Assistance, Lego Assistance, Sandwich Assistance, and Ikea Lamp Assembly Assistance. Using these applications as benchmarks, Chen et al. [7] presented empirical measurements detailing the latency contributions of individual system components. Furthermore, a multi-algorithm approach was proposed to reduce the latency of computer vision computation by executing multiple algorithms in parallel and conditionally selecting a fast and accurate algorithm for the near future.

While previous research has demonstrated the technical feasibility of wearable cognitive assistants and meeting latency requirements, many practical concerns have not been addressed. First, previous work operates the wireless networks and cloudlets at low utilization in order to meet application latency. The economics of practical deployment preclude operation at such low utilization. In contrast, resources are often highly utilized and congested when serving many users. How to efficiently scale Gabriel applications to a large number of users remains to be answered. Second, previous work on the Gabriel framework reduces application development efforts by managing client-server communication, network flow control, and cognitive engine discovery. However, the framework does not address the most time-consuming parts of creating a wearable cognitive assistance application. Experience has shown that developing computer vision modules that analyze video feeds is a time-consuming and painstaking process that requires special expertise and involves rounds of trial and error. Developer tools that alleviate the time and the expertise needed can greatly facilitate the creation of these applications.

1.1 Thesis Statement

In this thesis, we address the problem of scaling wearable cognitive assistance. Scalability here has a two-fold meaning. First, a scalable system should enable the most associated clients with fixed amount of infrastructure and has ways to serve more clients as resources increase. Second, we want to enable small software team to quickly create, deploy, and manage these applications. Notably, we claim that:

Two critical challenges to the widespread adoption of wearable cognitive assistance are 1) the need to operate cloudlets and wireless network at low utilization to achieve acceptable end-to-end latency 2) the level of specialized skills and the long development time needed to create new applications. These challenges can be effectively addressed through system optimizations, functional extensions, and the addition of new software development tools to the Gabriel platform.

The main contributions of this thesis are as follows:

1. We propose application-agnostic and application-aware techniques to reduce offered load from WCA mobile client when oversubscription occurs.
2. We provide a profiling-based cloudlet resource allocation mechanism that takes into account of applications' adaptation behaviors.
3. We create a suite of development and deployment tools to reduce the time and lower the barrier of entry for WCA developments.

1.2 Thesis Overview

The remainder of this thesis is organized as follows.

- In Chapter ??, we describe

Chapter 2

Background

2.1 Edge Computing

Edge computing is a nascent computing paradigm that has gained considerable traction over the past few years. It champions the idea of placing substantial compute and storage resources at the edge of the Internet, in close proximity to mobile devices or sensors. Terms such as “cloudlets” [36], “micro data centers (MDCs)” [14], “fog” [3], and “mobile edge computing (MEC)” [4] are used to refer to these small, edge-located computing nodes. We use these terms interchangeably in the rest of this thesis. Edge computing is motivated by its potential to improve latency, bandwidth, and scalability over a cloud-only model. More practically, some efforts stem from the drive towards software-defined networking (SDN) and network function virtualization (NFV), and the fact that the same hardware can provide SDN, NFV, and edge computing services. This suggests that infrastructure providing edge computing services may soon become ubiquitous, and may be deployed at greater densities than content delivery network (CDN) nodes today.

Satya et al. [38] best describes the modern computing landscape with edge computing using a tiered model, shown in Figure 2.1. Tiers are separated by distinct yet stable sets of design constraints. From left to right, this tiered model represents a hierarchy of increasing physical size, compute power, energy usage, and elasticity. Tier-1 represents today’s large-scale and heavily consolidated data-centers. Compute elasticity and storage permanence are two dominating themes here. Tier-3 represents IoT and mobile devices, which are constrained by their physical size, weight, and heat dissipation. Sensing is the key functionality of tier-3 devices. For example, today’s smartphones are already rich in sensors, including camera, microphone, accelerometers, gyroscopes and GPS. In addition, an increasing amount of IoT devices with specific sensing modalities are getting adopted, e.g. smart speakers, security cameras, and smart thermostats.

With the large-scale deployment of tier-3 devices, there exists a tension between the gigantic amount of data collected and generated by them and their limited capabilities to process these data on-board. For example, most surveillance cameras are limited in computation to run state-of-art computer vision algorithms to analyze the videos they capture. To overcome this ten-

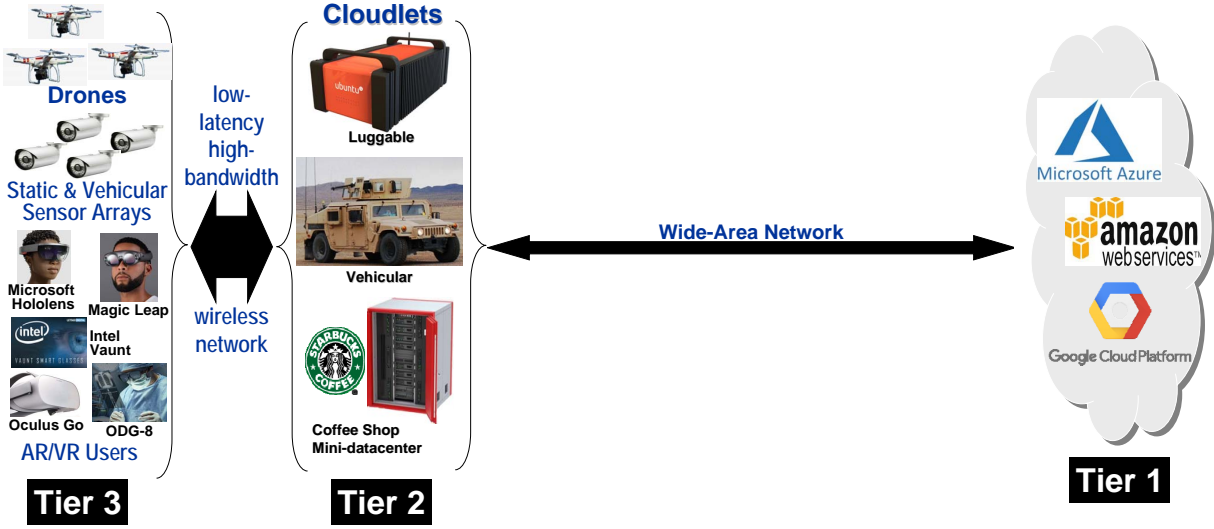


Figure 2.1: Tiered Model of Computing

sion, a tier-3 device could offload computation over network to tier-1. This capability was first demonstrated in 1997 by Noble et al. [28], who used it to implement speech recognition with acceptable performance on a resource-limited mobile device. In 1999, Flinn et al. [12] extended this approach to improve battery life. These concepts were generalized in a 2001 paper that introduced the term *cyber foraging* for the amplification of a mobile device’s data or compute capabilities by leveraging nearby infrastructure [33]. Thanks to these research efforts, computation offloading is widely used by IoT devices today. For example, when a user asks an Amazon Echo smart speaker “Alexa, what is the weather today?”, the user’s audio stream is captured by the smart speaker and transmitted to the cloud for speech recognition, text understanding, and question answering.

However, offloading computation to the cloud has its own downside. Because of the consolidation needed to achieve the economy of scale, today’s datacenters are “far” from tier-3 devices. The latency, throughput, and cost of wide-area network (WAN) significantly limit the amount of applications that can benefit from computation offloading. Even worse, it is the logical distance in the network that matters rather than the physical distance. Routing decisions in today’s Internet are made locally and are based on business agreements, resulting in suboptimal solutions. For example, using traceroute, we determine that a LTE packet originating from a smartphone on the campus of Carnegie Mellon University (CMU) in Pittsburgh to a nearby server actually traverses to Philadelphia, a city several hundreds miles away. This is because Philadelphia has the nearest peering point of the particular commercial LTE network in use to the public Internet. In 2010, Li et al. [23] report that the average round trip time (RTT) from 260 global vantage points to their optimal Amazon EC2 instances is 74 ms. In addition to long network delay, the high network fan-in of datacenters means its aggregation network need to carry significant amount of traffic. As the number of tier-3 devices is expected to grow exponentially, these network links face significant challenges to handle the ever-increasing volume of ingress traffic.

To counter these problems, edge computing, shown as the tier-2 in Figure 2.1, is proposed.

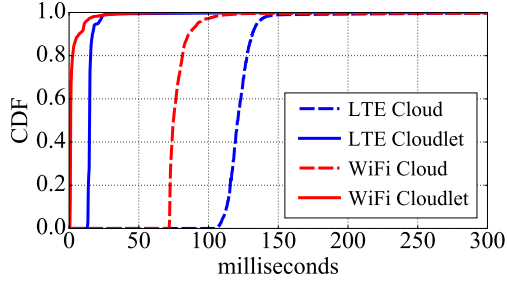


Figure 2.2: CDF of ping RTTs

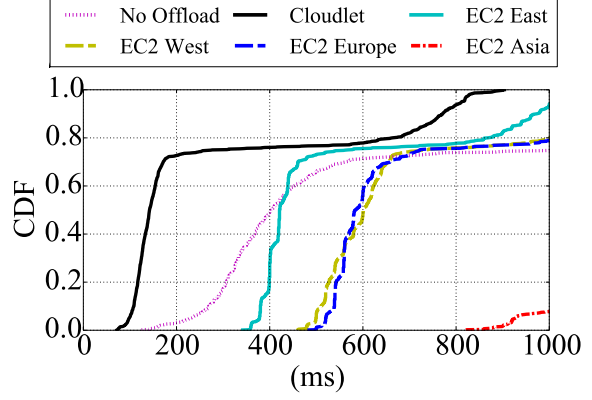


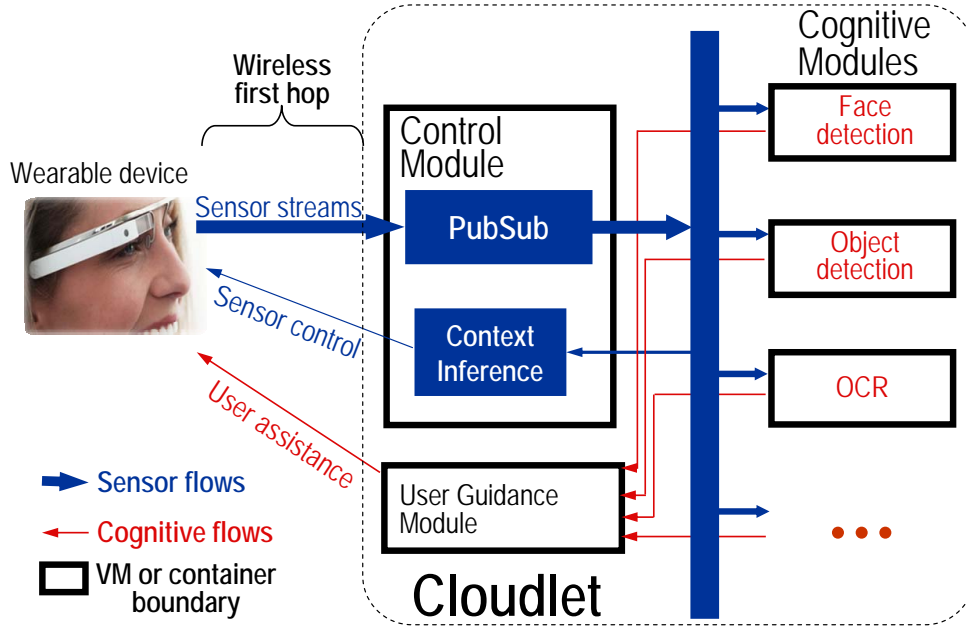
Figure 2.3: FACE Response Time over LTE

Cloudlets at tier-2 creates the illusion of bringing Tier-1 “closer”. They are featured by their network proximity to tier-3 devices and their significantly larger compute and storage compared to tier-3 devices. While tier-3 devices typically run on battery and are optimized for low energy consumption, saving energy is not a major concern for tier-2 as they are either plugged into the electric grid or powered by other sources of energy (e.g. fuels in a vehicle). Cloudlets serve two purposes in the tiered model. First, they provide infrastructure for compute-intensive and latency-sensitive applications for tier-3. Wearable cognitive assistance is an exemplar of these applications. Second, by processing data closer to the source of the content, it reduces the excessive traffic going into tier-1 datacenters. Figure 2.2 shows the RTT comparison of PING to the cloud and the cloudlet over WiFi and LTE. Cloudlet’s RTT is on average 80 to 100ms shorter than its counterpart to the cloud. Figure 2.3 shows the impact of network latency on an application that recognizes faces. Three offloading scenarios are considered: offloading to the cloud, offloading to the cloudlet, and no offloading. The data transmitted are images captured by a smartphone. As we can see, the limited bandwidth of the cellular network further worsen the response time when offloading to the cloud. In fact, for this particular application, even local execution outperforms offloading to the nearest cloud due to network delay. The optimal computational offload location is cloudlet, whose median response time is more than 200ms faster than local execution and about 250 ms faster than the nearest cloud.

The low-latency and high-bandwidth compute infrastructure provided by cloudlets is an indispensable foundation for latency-sensitive and compute-intensive wearable cognitive assistance. Cloudlet also poses unique challenges for scalability as resources are a lot more limited compared to datacenters. How to scale WCAs to many users using cloudlets is the key question this thesis set out to investigate.

2.2 Gabriel Platform

The Gabriel platform [7, 15], shown in Figure 2.4, is the first application framework for wearable cognitive assistance. It consists of a front-end running on wearable devices and a back-end



(Source: Chen et al [7])

Figure 2.4: Gabriel Platform

running on cloudlets. The Gabriel front-end performs preprocessing of sensor data (e.g., compression and encoding), which it then streams over a wireless network to a cloudlet. The Gabriel back-end on the cloudlet has a modular structure. The *control module* is the focal point for all interactions with the wearable device and can be thought as an agent for a particular client on the cloudlet. A publish-subscribe (PubSub) mechanism distributes the incoming sensor streams to multiple *cognitive modules* (e.g., task-specific computer vision algorithms) for concurrent processing. Cognitive module outputs are integrated by a task-specific *user guidance module* that performs higher-level cognitive processing such as inferring task state, detecting errors, and generating guidance in one or more modalities (e.g., audio, video, text, etc.). The Gabriel platform automatically discovers cognitive engines on the local network via a universal plug-and-play (UPnP) protocol. The platform is designed to run on a small cluster of machines with each modules capable of being separated or co-located with other modules via process, container, or virtual machine virtualization.

The original Gabriel platform was built with a single user in mind, and did not have mechanisms to share cloudlet resources in a controlled manner. It did, however, have a token-based transmission mechanism. This limited a client to only a small number of outstanding operations, thereby offering a simple form of rate adaptation to processing or network bottlenecks. We have retained this token mechanism in our system, described in the rest of this thesis. In addition, we have extended Gabriel with new mechanisms to handle multi-tenancy, perform resource allocation, and support application-aware adaptation. We refer to the two versions of the platform as “Original Gabriel” and “Scalable Gabriel.”

2.3 Example Gabriel Applications

Many applications have been built on top of the Gabriel platform. Figure 2.5 provides a summary of applications built by Chen et al [6]. These applications run on multiple wearable devices such as Google Glass, Microsoft HoloLens, Vuzix Glass, and ODG R7. At a high level, the cloudlet workflows of these applications are similar, and consist of two major phases. The first phase uses computer vision to extract a symbolic, idealized representation of the state of the task, accounting for real-world variations in lighting, viewpoint, etc. The second phase operates on the symbolic representation, implements the logic of the task at hand, and occasionally generates guidance for the user. In most WCA applications, the first phase is far more compute intensive than the second phase.

Building on top of previous experience, we built a few more complex applications focusing on assembly tasks. Below are brief descriptions of these applications with distinct challenges highlighted.

2.3.1 RibLoc Application



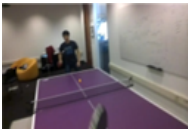









2.3.2 DiskTray Application

2.3.3 Model Car Assembly Application

2.4 Application Latency Bounds

Not only are the accuracy of the instructions important to WCAs, the speed at which these instructions are given is also important. With the human in the loop, the response times of WCAs are closely related to the human speed and the task speed. For example, for assembly tasks, an instruction delivered too late can cause user annoyance and frustration. For fast-paced games such as PINGPONG, an instruction delivered after the user has made a move becomes useless. To systematically quantify task-dependent application latency bounds, Chen et al. [7] employed three approaches: identify human speed through literature review (e.g. the time it takes for a human to recognize a face), physics-based calculation (e.g. the average speed of a PINGPONG ball while rallying), and user study (e.g. the maximum delay before a user is annoyed). Figure 2.6 presents the summary of their results. Each application is assigned both a tight and a loose bound from the application perspective. The tight bound represents an ideal target, below which the user is insensitive to improvements. Above the loose bound, the user becomes aware of slowness, and user experience and performance is significantly impacted. Latency improvements between the two limits may be useful in reducing user fatigue.

These application latency bounds can also be considered as application quality of service (QoS) metrics. Similar to bitrate and start-up time in video streaming, these metrics are measurable proxies to user experience. In addition, they reflect how much a user would suffer when

App Name	Example Input Video Frame	Description	Symbolic Representation	Example Guidance
Pool		Helps a novice pool player aim correctly. Gives continuous visual feedback (left arrow, right arrow, or thumbs up) as the user turns his cue stick. Correct shot angle is calculated based on fractional aiming system [1]. Color, line, contour, and shape detection are used. The symbolic representation describes the positions of the balls, target pocket, and the top and bottom of cue stick.	<Pocket, object ball, cue ball, cue top, cue bottom>	
Ping-pong		Tells novice to hit ball to the left or right, depending on which is more likely to beat opponent. Uses color, line and optical-flow based motion detection to detect ball, table, and opponent. The symbolic representation is a 3-tuple: in rally or not, opponent position, ball position. Whispers “left” or “right” or offers spatial audio guidance using [44]. Video URL: https://youtu.be/_lp32sowyUA	<InRally, ball position, opponent position>	Whispers “Left!”
Work-out		Guides correct user form in exercise actions like sit-ups and push-ups, and counts out repetitions. Uses Volumetric Template Matching [21] on a 10-15 frame video segment to classify the exercise. Uses smart phone on the floor for third-person viewpoint.	<Action, count>	Says “8”
Face		Jogs your memory on a familiar face whose name you cannot recall. Detects and extracts a tightly-cropped image of each face, and then applies a state-of-art face recognizer using deep residual network [16]. Whispers the name of a person. Can be used in combination with Expression [2] to offer conversational hints.	ASCII text of name	Whispers “Barack Obama”
Lego		Guides a user in assembling 2D Lego models. Each video frame is analyzed in three steps: (i) finding the board using its distinctive color and black dot pattern; (ii) locating the Lego bricks on the board using edge and color detection; (iii) assigning brick color using weighted majority voting within each block. Color normalization is needed. The symbolic representation is a matrix representing color for each brick. Video URL: https://youtu.be/7L9U-n29abg	[[0, 2, 1, 1], [0, 2, 1, 6], [2, 2, 2, 2]]	 Says “Put a 1x3 green piece on top”
Draw		Helps a user to sketch better. Builds on third-party app [19] that was originally designed to take input sketches from pen-tablets and to output feedback on a desktop screen. Our implementation preserves the back-end logic. A new Glass-based front-end allows a user to use any drawing surface and instrument. Displays the error alignment in sketch on Glass. Video URL: https://youtu.be/nuQpPtVJC6o		
Sand-		Helps a cooking novice prepare sandwiches according to a recipe. Since real food is perishable, we use	Object:	

	Pool	Work-out	Ping-pong	Face	Assembly Tasks (e.g. RibLoc)
Bound Range (tight-loose)	95-105	300-500	150-230	370-1000	600-2700

Figure 2.6: Application Latency Bounds (in milliseconds)

(Adapted from Chen et al [7])

system response delay increases. In this thesis, we use these bounds to formulate application utility functions to quantify user experience when an application is forced to lower its fidelity under contention.

Chapter 3

Application-Agnostic Techniques to Reduce Network Transmission

1. talk about weak and accurate detectors
2. evaluate in drone context
3. evaluate in WCA contexts

3.1 EARLYDISCARD Strategy

3.1.1 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 [17] [27] 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 ??, 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 ??, early discard only requires image classification: it is

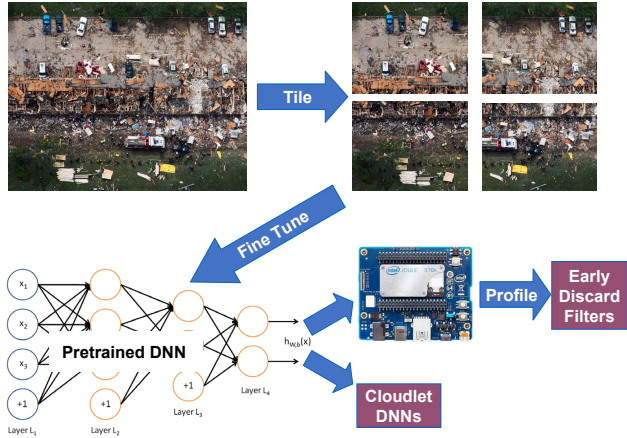


Figure 3.1: Tiling and DNN Fine Tuning

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* [46] 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.

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 [18]. 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 3.1, 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.

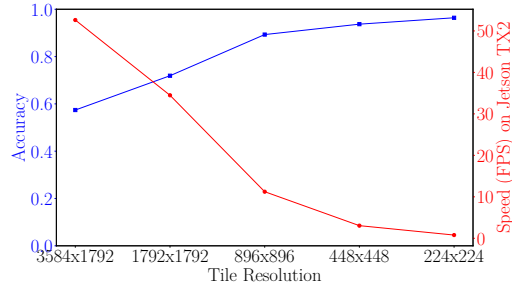


Figure 3.2: Speed-Accuracy Trade-off of Tiling

3.1.2 Experimental Setup

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

3.1.3 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 ??, to explore the effects of tiling. For this dataset, Figure 3.2 shows how speed and accuracy change with tile size. Accuracy 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 3.3 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 3.3 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 3.4 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 3.4 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 timescale of human actions such as dispatching of a rescue team, these are negligible delays.

Bandwidth: Columns 5–7 of Figure 3.4 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 3.4 with column 2 of Figure ??, 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 3.3 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.

3.1.4 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 3.5 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 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

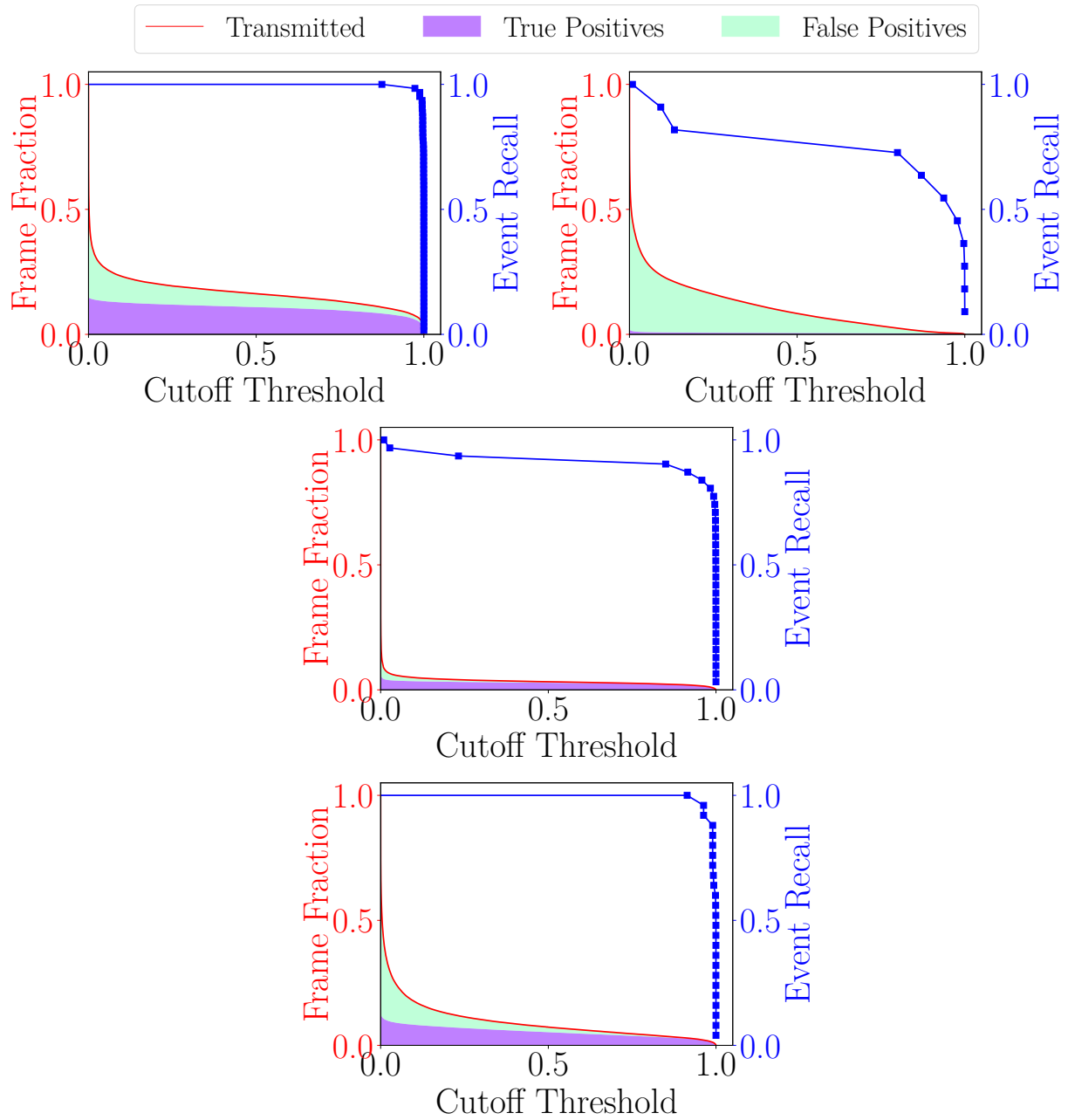


Figure 3.3: Where the Bandwidth Goes

	Task Total Events	Dete- cted 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 3.4: Recall, Event Latency and Bandwidth at Cutoff Threshold 0.5

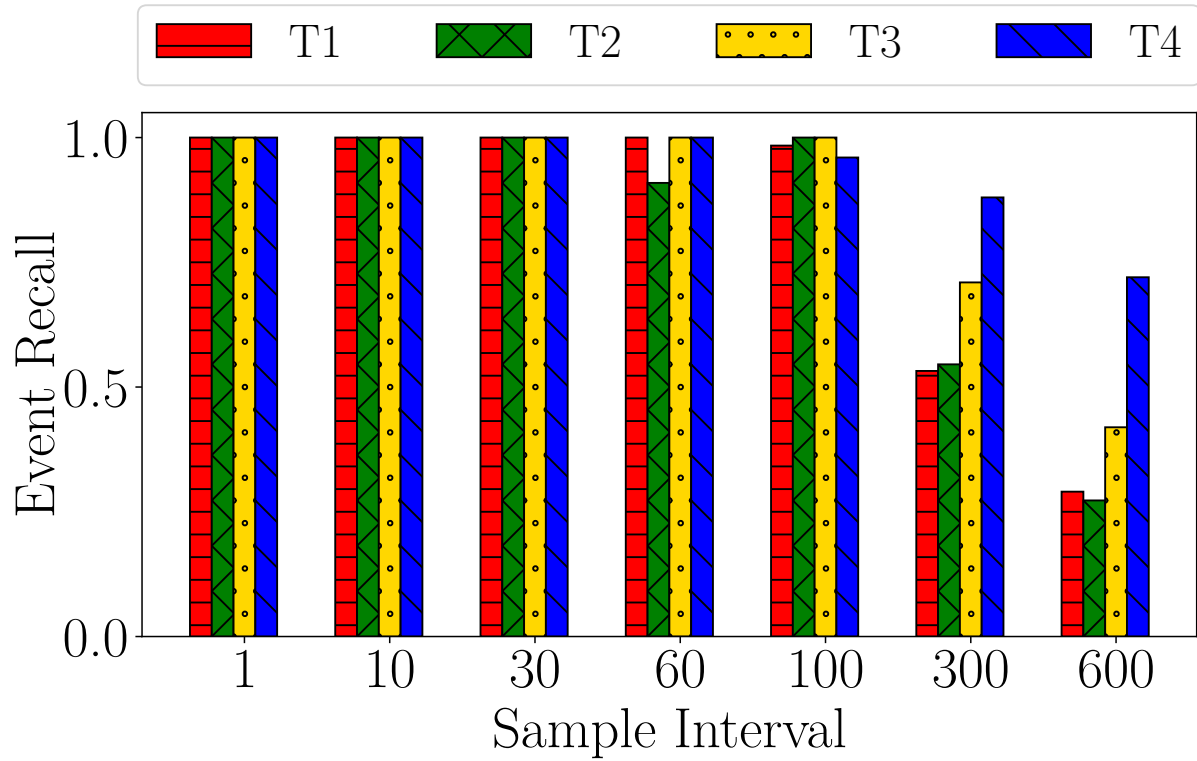


Figure 3.5: Event Recall at Different Sampling Intervals

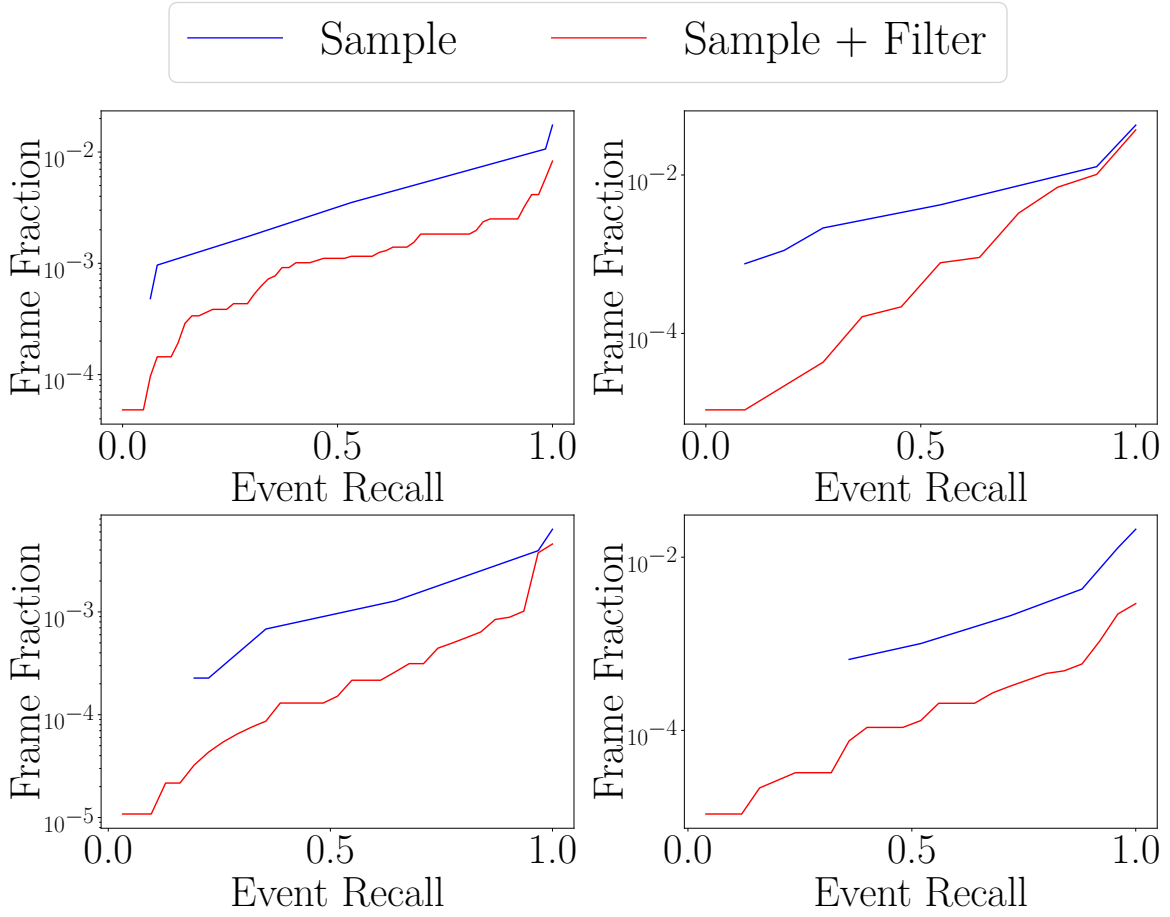


Figure 3.6: Sample with Early Discard. Note the log scale on y-axis.

JPEG Frame Se- quence (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 [26].

Figure 3.7: Test Dataset Size With Different Encoding Settings

events (false negatives).

Can we use content-based filtering along with sampling to further reduce bandwidth consumption? Figure 3.6 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.

3.1.5 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 3.7 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.

3.2 JUST-IN-TIME-LEARNING **Strategy To Improve Early Discard**

3.2.1 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 [45], 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) [13] 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.

3.2.2 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.



Figure 3.8: JITL Fraction of Frames under Different Event Recall

3.2.3 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 3.8 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 [11] could be explored to synthesize true positives from the background of the current flight.

3.3 Evaluation

3.4 Discussion

Chapter 4

Conclusion and Future Work

Bibliography

- [1] Fractional-ball aiming. <http://billiards.colostate.edu/threads/aiming.html>. Accessed on November 27, 2015. ??
- [2] Asm Iftekhar Anam, Shahinur Alam, and Mohammed Yeasin. Expression: A dyadic conversation aid using google glass for people with visual impairments. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 211–214. ACM, 2014. ??
- [3] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, Finland, 2012. 2.1
- [4] Gabriel Brown. Converging Telecom & IT in the LTE RAN. White Paper, Heavy Reading, February 2013. 2.1
- [5] Vannevar Bush and Vannevar Bush. As we may think. *Resonance*, 5(11), 1945. 1
- [6] Zhuo Chen. *An Application Platform for Wearable Cognitive Assistance*. PhD thesis, Carnegie Mellon University, 2018. 1, 2.3
- [7] Zhuo Chen, Wenlu Hu, Junjue Wang, Siyan Zhao, Brandon Amos, Guanhang Wu, Kiryong Ha, Khalid Elgazzar, Padmanabhan Pillai, Roberta Klatzky, Dan Siewiorek, and Mahadev Satyanarayanan. An Empirical Study of Latency in an Emerging Class of Edge Computing Applications for Wearable Cognitive Assistance. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, October 2017. 1, 2.2, ??, 2.4, 2.6
- [8] Keith Cheverst, Nigel Davies, Keith Mitchell, Adrian Friday, and Christos Efstratiou. Developing a context-aware electronic tourist guide: some issues and experiences. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 17–24. ACM, 2000. 1
- [9] Nigel Davies, Keith Mitchell, Keith Cheverst, and Gordon Blair. Developing a context sensitive tourist guide. In *1st Workshop on Human Computer Interaction with Mobile Devices, GIST Technical Report G98-1*, volume 1, 1998. 1
- [10] Zak Doffman. Battlefield 2.0: How Edge Artificial Intelligence is Pitting Man Against Machine. *Forbes*, November 2018. 1
- [11] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *The IEEE international conference on computer vision (ICCV)*, 2017. 3.2.3

- [12] Jason Flinn and Mahadev Satyanarayanan. Energy-aware Adaptation for Mobile Applications. In *Proceedings of the Seventeenth ACM Symposium on Operating systems Principles*, Charleston, SC, 1999. 2.1
- [13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics, 2001. 3.2.1
- [14] Kira Greene. AOL Flips on ‘Game Changer’ Micro Data Center. <http://blog.aol.com/2012/07/11/aol-flips-on-game-changer-micro-data-center/>, July 2012. 2.1
- [15] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. Towards Wearable Cognitive Assistance. In *Proceedings of ACM MobiSys*, 2014. 1, 2.2
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. ??
- [17] Wenlu Hu, Brandon Amos, Zhuo Chen, Kiryong Ha, Wolfgang Richter, Padmanabhan Pillai, Benjamin Gilbert, Jan Harkes, and Mahadev Satyanarayanan. The Case for Offload Shaping. In *Proceedings of HotMobile 2015*, Santa Fe, NM, February 2015. 3.1.1
- [18] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3.1.1
- [19] Emmanuel Iarussi, Adrien Bousseau, and Theophanis Tsandilas. The drawing assistant: Automated drawing guidance and feedback from photographs. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2013. ??
- [20] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. ??
- [21] Yan Ke, Rahul Sukthankar, and Martial Hebert. Event detection in crowded videos. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007. ??
- [22] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009. ??
- [23] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010. 2.1
- [24] Jack M Loomis, Reginald G Golledge, Roberta L Klatzky, Jon M Speigle, and Jerome Tietz. Personal guidance system for the visually impaired. In *Proceedings of the first annual ACM conference on Assistive technologies*, pages 85–91. ACM, 1994. 1
- [25] Jack M Loomis, Reginald G Golledge, and Roberta L Klatzky. Navigation system for the blind: Auditory display modes and guidance. *Presence*, 7(2):193–203, 1998. 1

- [26] Loren Merritt and Rahul Vanam. Improved rate control and motion estimation for h. 264 encoder. In *IEEE International Conference on Image Processing*, 2007. ??
- [27] Saman Naderiparizi, Pengyu Zhang, Matthai Philipose, Bodhi Priyantha, Jie Liu, and Deepak Ganesan. Glimpse: A Programmable Early-Discard Camera Architecture for Continuous Mobile Vision. In *Proceedings of MobiSys 2017*, June 2017. 3.1.1
- [28] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, J. Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile Application-Aware Adaptation for Mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, Saint-Malo, France, October 1997. 2.1
- [29] John Oakley. Intelligent Cognitive Assistants (ICA) Workshop Summary and Research Needs. https://www.nsf.gov/crssprgm/nano/reports/ICA2_Workshop_Report_2018.pdf, February 2018. 1
- [30] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010. ??
- [31] Tiernan Ray. An Angel on Your Shoulder: Who Will Build A.I.? *Barron's*, February 2018. 1
- [32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. ??
- [33] Mahadev Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, 8(4), 2001. 2.1
- [34] Mahadev Satyanarayanan. Augmenting Cognition. *IEEE Pervasive Computing*, 3(2), April-June 2004. 1
- [35] Mahadev Satyanarayanan and Nigel Davies. Augmenting Cognition through Edge Computing. *IEEE Computer*, 52(7), July 2019. 1
- [36] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, 8(4), October-December 2009. 2.1
- [37] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, pages 14–23, 2009. 1
- [38] Mahadev Satyanarayanan, Wei Gao, and Brandon Lucia. The computing landscape of the 21st century. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, pages 45–50. ACM, 2019. 2.1
- [39] Mahadev Satyanarayanan, Guenter Klas, Marco Silva, and Simone Mangiante. The Seminal Role of Edge-Native Applications. In *Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE)*, Milan, Italy, July 2019. 1
- [40] Asim Smailagic and Daniel Siewiorek. Application design for wearable and context-aware computers. *IEEE Pervasive Computing*, 1(4):20–29, 2002. 1

- [41] Asim Smailagic and Daniel P Siewiorek. A case study in embedded-system design: The vuman 2 wearable computer. *IEEE Design & Test of Computers*, 10(3):56–67, 1993. 1
- [42] Asim Smailagic, Daniel P Siewiorek, Richard Martin, and John Stivoric. Very rapid prototyping of wearable computers: A case study of vuman 3 custom versus off-the-shelf design methodologies. *Design Automation for Embedded Systems*, 3(2-3):219–232, 1998. 1
- [43] Sherry Stokes. New Center Headquartered at Carnegie Mellon Will Build Smarter Networks To Connect Edge Devices to the Cloud. <https://www.cmu.edu/news/stories/archives/2018/january/conix-research-center.html>, January 2018. 1
- [44] Titus JJ Tang and Wai Ho Li. An assistive eyewear prototype that interactively converts 3d object locations into spatial audio. In *Proceedings of the 2014 ACM International Symposium on Wearable Computers*, pages 119–126. ACM, 2014. ??
- [45] Paul Viola and Michael Jones. Robust Real-time Object Detection. In *International Journal of Computer Vision*, 2001. 3.2.1
- [46] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014. 3.1.1