Video Analytics with Zero-streaming Cameras

Mengwei Xu, *Peking University/Beijing University of Posts and Telecommunications;*
Tiantu Xu, *Purdue ECE;* Yunxin Liu, *Institute for AI Industry Research* (*AIR*),
Tsinghua University; Felix Xiaozhu Lin, *University of Virginia*

https://www.usenix.org/conference/atc21/presentation/xu

This paper is included in the Proceedings of the
2021 USENIX Annual Technical Conference.

July 14–16, 2021

978-1-939133-23-6

Video Analytics with Zero-streaming Cameras

Mengwei Xu[1,2]*, Tiantu Xu[3]*, Yunxin Liu[4], and Felix Xiaozhu Lin[5]

[1] *Peking University*
[2] *Beijing University of Posts and Telecommunications*
[3] *Purdue ECE*
[4] *Institute for AI Industry Research (AIR), Tsinghua University*
[5] *University of Virginia*

Abstract

Low-cost cameras enable powerful analytics. An unexploited opportunity is that most captured videos remain "cold" without being queried. For efficiency, we advocate for these cameras to be *zero streaming*: capturing videos to local storage and communicating with the cloud only when analytics is requested.

How to query zero-streaming cameras efficiently? Our response is a camera/cloud runtime system called DIVA. It addresses two key challenges: to best use limited camera resource during video capture; to rapidly explore massive videos during query execution. DIVA contributes two unconventional techniques. (1) When capturing videos, a camera builds sparse yet accurate landmark frames, from which it learns reliable knowledge for accelerating future queries. (2) When executing a query, a camera processes frames in multiple passes with increasingly more expensive operators. As such, DIVA presents and keeps refining inexact query results throughout the query's execution. On diverse queries over 15 videos lasting 720 hours in total, DIVA runs at more than 100× video realtime and outperforms competitive alternative designs. To our knowledge, DIVA is the first system for querying large videos stored on low-cost remote cameras.

1 Introduction

Cameras are pervasive: a survey of 61 organizations shows that from 2015 to 2018 their average number of cameras has increased by almost 70%, from 2,900 to 4,900 [6]. Insights of videos can be extracted by queries such as "get the daily peak pedestrian count in the past week" [36, 67, 82, 101]. Four recent trends motivate our work.

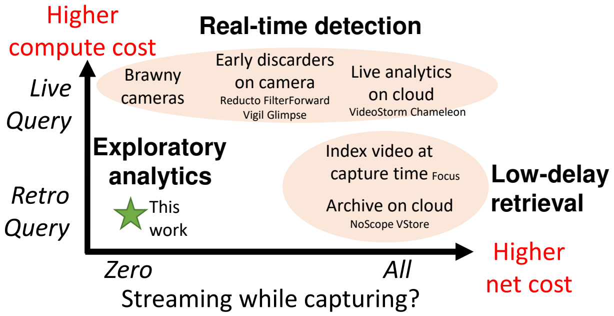


Figure 1: **The design space** of video analytics systems, showing this work and prior systems.

(1) Low-cost, wireless cameras grow fast As key complements to high-end cameras, low-cost cameras (<$40) are increasingly pervasive [17, 18, 34]. These cameras often have limited compute resources yet spacious storage. Being wireless, these cameras are meant to be installed by individuals or small businesses with ease just as other wireless sensors.

(2) Most videos are cold Users deploy cameras to knowingly capture excessive videos, expecting that most videos will never be queried [72]. This is because interesting events are often unforeseeable, e.g., car accidents; the need for examining such events emerges well after the fact. §2.1 presents a 6-month study of real-world camera deployment, where only <0.005% of captured videos are eventually queried.

(3) Transmitting cold videos wastes wireless bandwidth Cold videos should not compete with human users for network bandwidth. If streaming video in real-time, a single camera generates traffic at 0.2 MB/s–0.4 MB/s (720P@1–30 FPS); with multiple cameras on one network, their always-on streams easily consume most, if not all, bandwidth of consumer WiFi, which is 0.2 MB/s–3 MB/s (median: 0.99) in a recent global survey [9] and less than 1.5 MB/s in an academic study [47]. A dedicated network for cameras is expensive, as the network monetary cost will exceed the camera cost in several months [14].

(4) Camera storage can retain videos long enough A cheap

*Mengwei Xu and Tiantu Xu contributed equally to the paper.
*Work done during Mengwei Xu's visit to Purdue University.

camera can already store videos for weeks or months. Such retention periods already satisfy many video scenarios [2, 10]. In fact, legal regulations often *prevent* retention longer than a few months, mandating video deletion for privacy [1, 7]. Existing measures can assure data security of on-camera videos. §2.3 will provide evidence in detail.

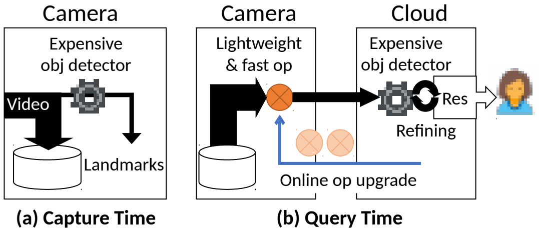
Zero streaming & its use cases How to analyze cold videos produced by numerous low-cost cameras? We advocate for a system model dubbed "zero streaming". (1) Cameras continuously capture videos to their local storage without uploading any. (2) Only in response to a retrospective query, the cloud reaches out to the queried camera and coordinates with it to process the queried video. (3) While the video is being processed, the system presents users with inexact yet useful results; it continuously refines the results until query completion [50]. In this way, a user may *explore* videos through interactive queries, e.g., aborting an ongoing query based on inexact results and issuing a new query with revised parameters [45, 46]. Zero streaming has rich use cases, for example:

- To trace the cause of recent frequent congestion on a highway, a city planner queries cameras on nearby local roads, requesting car counts seen on these local roads.
- To understand how recent visitors impact bobcat activities, a ranger queries all the park's cameras, requesting time ranges where the cameras capture bobcats.

Advantages Zero streaming suits resource-frugal cameras in large deployment. When capturing videos, cameras require no network or external compute resources. Only to process a query, the cameras require networks such as long-range wireless [35] and cloud resources such as GPU. Zero streaming adds a new point to the design space of video analytics shown in Figure 1. It facilitates retrospective, exploratory analytics, a key complement to real-time event detection and low-delay video retrieval [51,55,65,99]. The latter demands higher compute or network resources per camera and hence suits fewer cameras around hot locations such as building entrances.

DIVA To support querying zero-streaming cameras, we present a camera/cloud runtime called DIVA. As shown in Figure 2, a camera captures video to local storage; it deletes videos after their maximum retention period. In response to a query, the camera works in conjunction with the cloud: the camera runs operators, implemented as lightweight neural nets (NNs), to *rank* or *filter* frames; the cloud runs full-fledged object detection to validate results uploaded from the camera. DIVA thus does not sacrifice query accuracy, ensuring it as high as that of object detection by the cloud.

The major challenges to DIVA are two. (1) During video capture: how should cameras best use limited resources for future queries? (2) To execute a query: how should the cloud and the camera orchestrate to deliver useful results rapidly? Existing techniques are inadequate. Recent systems pre-process ("index") video frames as capturing them [51] and answer queries based on indexes only. Yet, as we will show in §8, low-cost cameras can hardly build quality indexes in real-time. Many systems process video frames in a streaming fashion [40, 42, 92, 97, 100], which however miss key opportunities in retrospective queries.



(a) Capture Time (b) Query Time

Figure 2: **Overview of DIVA**

To this end, DIVA has two unconventional designs.

- **During video capture: building sparse but sure landmarks to distill long-term knowledge** (Figure 2(a)) To optimize future queries, our key insight is that *accurate* knowledge on a *sparse* sample of frames is much more useful than *inaccurate* knowledge on *all* frames. This is opposite to existing designs that detect objects with low accuracy on all/most frames as capturing them [40, 51]. On a small sample of captured video frames dubbed *landmarks*, the camera runs generic, expensive object detection, e.g., YOLOv3 [77]. Constrained by camera hardware, landmarks are sparse in time, e.g., 1 in every 30 seconds; yet, with high-accuracy object labels, they provide reliable spatial distributions of various objects over long videos. High accuracy is crucial, as we will validate through evaluation (§8.3). DIVA optimizes queries with landmarks: it prioritizes processing of frame regions with object skewness learned from landmarks; it bootstraps operators with landmarks as training samples. Landmarks only capture a small fraction of object instances; those uncaptured do not affect correctness/accuracy (§4).
- **To execute queries: multipass processing with online operator upgrade** (Figure 2(b)) To process large videos, our key insight is to refine query results in multiple passes, each pass with a more expensive/accurate operator. Unlike prior systems processing all frames in one pass and delivering results in one shot [40, 58, 59], multipass processing produces useful results during query execution, enabling users to explore videos effectively. To do so, DIVA's cloud trains operators with a wide spectrum of accuracies/costs. Throughout query execution, the cloud keeps pushing new operators to the camera, picking the next operator based on query progress, network conditions, and operator accuracy. The early operators quickly *explore* the frames for inexact answers while later operators slowly *exploit* for more exact answers.

On 720-hour videos in total from 15 different scenes, DIVA runs queries at more than 100× video realtime on average, with typical wireless conditions and low-cost hardware. DIVA returns results quickly: compared to executing a query to completion, DIVA takes one order of magnitude shorter time to return half of the result frames. Compared to competitive alternatives, DIVA speeds up queries by at least 4×.

Contributions We have made the following contributions.

• Zero streaming, a new model for low-cost cameras to operate on frugal networks while answering video queries.

• Two novel techniques for querying zero-streaming cameras: optimizing queries with accurate knowledge from sparse frames; processing frames in multiple passes with operators continuously picked during a query.

• DIVA, a concrete implementation that runs queries at more than 100× realtime with uncompromised query accuracy. To our knowledge, DIVA is the first system designed for querying large videos stored on low-cost remote cameras.

Ethical considerations In this study: all visual data used is from the public domain; no information traceable to human individuals is collected or analyzed.

2 Motivations

2.1 Cold videos are already pervasive

Case study: Cold videos in real-world deployment We conduct an IRB-approved study examining existing camera deployment on PKU campus. Spanning 1 mi^2, the campus hosts tens of thousands of employees and operates more than 1,000 cameras. All captured videos are stored for a few months for retrospective queries before deletion. The camera deployment supports AI-based queries, e.g., object detection, *not* traceable to unique persons, and reviews by human analysts. We analyzed system logs spanning six continuous months: in over 3,000,000 hours of videos (5.4 PB) have been captured, only <0.005% video data from <2% cameras are queried.

Why are most videos cold? (1) Interesting video events are both unpredictable (thus the need for capturing excessive videos) and sparse (thus low chances for footage being queried). For example, severe traffic breakdown contributes to less than 5% of the time per day [89]; Foreign intelligence surveillance court only reviewed a tiny fraction of video for terrorism events [93]. (2) Analyzing videos is expensive: it still requires a GPU of a few thousand dollars for high-accuracy object detection over a video stream [59]. (3) In years to come, cheap cameras will produce more videos.

2.2 Target queries and their execution

We target ad-hoc queries [51, 59, 96, 100]. The query parameters, including object classes, video timespans, and expected accuracies, are specified at query time rather than video capture time. Such queries are known for flexibility.

High-accuracy object detection is essential Object detection is the core of ad-hoc queries [58]. Minor accuracy loss in object detection may result in substantial loss in query performance, as we will demonstrate in §8. While NNs significantly advance object detection, new models with higher accuracy demand much more compute. For instance, compared to YOLOv3 (2018) [77], CornerNet (2019) [64] improves Average Precision by 28% while being 5× more expensive.

Low-cost cameras cannot answer queries without cloud Cameras in real-world deployment are reported to be resource-constrained [65]. Low-cost cameras (<$40) have wimpy cores, e.g., Cortex-A9 cores for YI Home Camera [18] and MIPS32 cores for WyzeCam [17]; their DRAM is no more than a few GBs [15, 16]. In recent benchmarks, they run state-of-the-art object detection at 0.1 FPS [8, 66], incapable of keeping up with video capture at 1–30 FPS [51, 59]. NN accelerators still cannot run high-accuracy object detection fast enough at low enough monetary cost, e.g., Intel's Movidius ($70) runs YOLOv3 at no faster than 0.5 FPS. In the foreseeable future, we expect that the resource gap between high-accuracy object detection and low-cost camera continues to exist.

2.3 A case for zero streaming

Streaming cold videos wastes bandwidth As discussed in §1, cameras are cheap while wireless spectrum is precious. Deploying streaming cameras on a shared network incurs poor experience [3, 11] and draws researcher attention [40, 100]. Dedicated networks are costly [14] and thus only suit a small number of cameras in critical locations. While wireless bandwidth grows, consumer demand grows even faster, e.g., 20× for VR/AR and 10× for gaming [5]. Cold video traffic should not contend with consumers for network bandwidth.

Streaming optimizations cannot offset the waste One may reduce FPS or resolution of streamed videos. Even if users tolerate the resultant lower query accuracy, the saved bandwidth is incomparable to the waste on overwhelmingly streamed cold videos, as we will experimentally show (§8). On-camera "early filters" [40, 42, 65] are still suboptimal when querying massive *cold* videos. (1) Without knowing query objects/parameters at video capture time, a camera may run a generic filter, e.g., discarding no-motion frames; it still streams substantial survival frames (e.g., consider a street-view camera). As stated above, most of these frames will remain cold and hence wasted. (2) The camera may run a large set of specific filters covering all possible query objects/parameters. Even if possible, this incurs a much higher compute cost to camera.

Edge processing does not justify streaming Cameras may stream to edge servers. Yet, streaming hundreds if not thousands of *always-on*, *cold* video streams, even if possible on certain wireless infrastructures, still wastes precious wireless spectrum at the edge [69]. Furthermore, deploying and managing video edge servers can be challenging and costly in many scenarios, such as construction sites and remote farms.

Size	Yr.2017	Yr.2020	720p@30FPS	720p@1FPS
128GB	$45	$17	∼11 days	∼3 weeks
256GB	$150	$28	∼3 weeks	∼ 6 weeks

Table 1: Cheap μSD cards on cameras retain long videos for humans to review [4] or for machines to analyze [51].

Cameras can retain videos long enough Table 1 shows the price of μSD cards has been dropped by 2.6×–5.4× in the past few years. Cameras can retain videos for several weeks and for several months soon. Such a retention period is already adequate for most retrospective query scenarios, where videos are retained from a few weeks to a few months based on best practice and legal regulations [1, 2, 7, 10]. For privacy, many regulations *prohibit* video retention longer than a few months and mandate deletion afterwards [1, 7].

Our model & design scope To harness cold videos, we advocate for zero streaming. We focus on cold videos being queried for the first time and querying individual cameras. We intend our design to form the basis of future enhancement and extension, e.g., resource scheduling for multiple queries/users/tenants [40], caching for repetitive queries [95], exploiting past queries for refinement [41], and exploiting cross-camera topology [54]. We address limited compute resource on cameras [15] and limited network bandwidth [47]. We do not consider the cloud as a limiting factor, assuming it runs fast enough to process frames uploaded from cameras.

3 The DIVA Overview

Query types Concerning a specific camera, an ad-hoc query $(\mathcal{T}, \mathcal{C})$ covers a video timespan $\mathcal{T}$, typically hours or days, and an object class $\mathcal{C}$ as detectable by modern NNs, e.g., any of the 80 classes of YOLOv3 [77]. As summarized in Table 2, DIVA supports three query types: ***Retrieval***, e.g., "retrieve all images that contain buses from yesterday"; ***Tagging***, e.g., "return all time ranges when any deer shows up in the past week", in which the time ranges are returned as metadata but not images; ***Counting***, e.g., "return the maximum number of cars that ever appear in any frame today".

System components DIVA spans a camera and the cloud. Between them, the network connection is only provisioned at query time. To execute a query, a camera runs lightweight NNs, or operators, to *filter* or *rank* the queried frames for upload. On the uploaded frames, the cloud runs generic, high-accuracy object detection and materializes query results. Table 2 summarizes executions for different queries:

- The camera executes ***rankers*** for *Retrieval* and *max Count* queries. A ranker scores frames; a higher score suggests that a frame is more likely to contain *any* object of interest (for Retrieval) or *a large count* of such objects (for max Count).

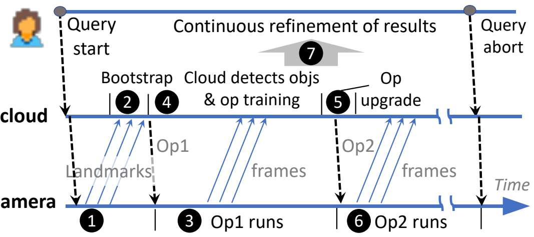


Figure 3: **The workflow of a query's execution.**

- The camera executes ***filters*** for *Tagging* queries. A filter scores frames; it resolves any frame scored below/above two pre-defined thresholds as negative/positive, and deems other frames as unresolved. For each resolved frame, the camera uploads a positive/negative tag; the camera either uploads unresolved frames for the cloud to decide or defer them to more accurate filters on camera in subsequent passes.

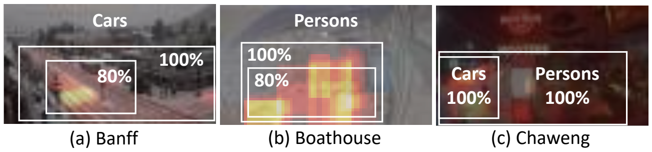
Query execution Upon receiving a query, the cloud retrieves all landmarks in queried video as low-resolution thumbnails, e.g., 100×100, with object labels and bounding boxes (Figure 3 ❶). The cloud uses landmarks: ***(1) to estimate object spatial distribution***, e.g., "90% queried objects appear in a 100×100 region on the top-right", which is crucial to query optimization (§4); ***(2) as the initial training samples*** for bootstrapping a family of camera operators (❷). The camera filters/ranks frames and uploads the ranked or surviving frames (❸). The cloud processes the uploaded frames and emits results, e.g., positive frames. It trains operators for higher accuracy (❹). Observing resource conditions and positive ratios in uploaded frames, the cloud upgrades the operator on camera (❺). With the upgraded operator, the camera continues to process remaining frames (❻). Step (❹)–(❻) repeat until query abort or completion. Throughout the query, the cloud keeps refining the results presented to the user (❼).

Notable designs (1) The camera processes frames in multiple passes, one operator in each pass. (2) The camera processes and uploads frames asynchronously. For instance, when the camera finishes ranking 100 out of total 1,000 frames, it may have uploaded the top 50 of the 100 ranked frames. This is opposed to common ranking which holds off frame upload until all the frames are ranked [38, 53, 61]. (3) The processing/upload asynchrony facilitates video exploration: it amortizes query delay over many installments of results; it pipelines query execution with user thinking [45]. Table 2 summarizes a user's view of query results and the performance metrics. While such online query processing has been known [43, 71], we are the first applying it to visual data.

Limitations DIVA is not designed for several cases and may underperform: querying very short video ranges, e.g., minutes, for which simply uploading all queried frames may suffice without operators; querying non-stationary cameras for which landmarks may not yield accurate object distribution. DIVA is vulnerable to loss of video data in case of camera stor-

Type & Semantics	Execution	User's view of query results	Performance Metrics
Retrieval. Get positive video frames (i.e., containing C) within T	Camera: multipass ranking of frames Uploaded: ranked frames Cloud: object detection for identifying true positives	• Positive frames being uploaded; • Estimated % of positives retrieved	The rate of the user receiving positive frames
Tagging. Get time ranges from T that contain C	Camera: multipass filtering of frames Uploaded: unresolved frames; tags of resolved frames Cloud: object detection to tag unresolved frames	• A video timeline with pos/neg ranges; • Tagging resolution, i.e., 1 in every N adjacent frames tagged	The refining rate of tagging resolution seen by the user
Counting. Get max/mean/median count of C across all frames in T	Camera: multipass ranking (max) or random sampling (mean/median) of frames Uploaded: ranked or sampled frames Cloud: object detection to count objects	• Running counts that converge to ground truth; • % of frames processed; • Estimated time to complete the query	The rate of running counts converging to ground truth

Table 2: **A summary of supported queries.** $\mathcal{T}$ is the queried video timespan; $\mathcal{C}$ is the queried object class



(a) Banff (b) Boathouse (c) Chaweng

Figure 4: **Class spatial skews in videos**. In (a) Banff: 80% and 100% of cars appear in regions that are only 19% and 57% of the whole frame, respectively.

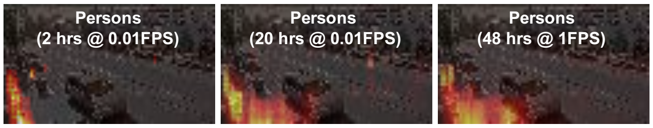


Figure 5: **Class spatial distribution can be estimated from sparse frames sampled over long video footage.** Among the three heatmaps: while sparse sampling over short footage (left) significantly differs from dense sampling of long footage (right), sparse sampling of long footage (middle) is almost equivalent to the right. Video: Tucson (see Table 4).

age failure. Users can mitigate such a risk via cross-camera data backup (RAID-like techniques) on the same local area network or by increasing camera deployment density.

4 Landmark Design

Surveillance cameras have a unique opportunity: to learn *object class distribution* from weeks of videos. We focus on **spatial skews**: objects of a given class are likely to concentrate on certain small regions on video frames. In examples of Figure 4(a)-(b), most cars appear near a stop sign; most persons appear in a shop's aisle. Such long-term skews are rarely tapped in prior computer vision work, which mostly focused on minute-long videos [52, 54, 78, 81, 102]. Compared to recent work that improved classifier performance by cropping video frames [40], DIVA takes a step further by automatically learning spatial skews from sparse frames with resource efficiency.

The design is backed up by three key observations. (1) One object class may exhibit different skews in different videos (Figure 4(a)-(c)); different classes may exhibit different skews in the same video (Figure 4(c)). (2) The skews are pervasive: surveillance cameras cover long time spans and a wide field of view, where objects are small; in the view, objects are subject to social constraints, e.g., buses stop at traffic lights, or physical constraints, e.g., humans appear on the floor. (3) The skews can be learned through *sparse* frame samples, as exemplified by Figure 5.

To exploit such an opportunity, DIVA makes the following design choices. **High-accuracy object detection:** at capture time, the camera runs an object detector with the highest accuracy as allowed by the camera's hardware, mostly memory capacity. This is because camera operators crucially depend on the correctness of landmarks, i.e., the object labels and bounding boxes. We will validate this experimentally (§8.3). **Sparse sampling at regular intervals:** to accommodate slow object detection on cameras, the camera creates landmarks at long intervals, e.g., 1 in every 30 seconds in our prototype (§8). Sparse sampling is proven valid for estimating statistics of low-frequency signals [37], e.g., object occurrence in videos in our case. We will validate this (§8.3); without assuming a priori of object distribution, regular sampling ensures unbiased estimation of the distribution [79]. Given a priori, cameras may sample at corresponding random intervals for unbiased estimation.

Key idea: exploiting spatial skews for performance The cloud learns the object class distribution from landmarks of the queried video timespan. It generates a heatmap for spatial distribution (Figure 4). Based on the heatmap, the cloud produces camera operators consuming frame regions of different locations and sizes. Take Figure 4(a) as an example: a filter may consume bottom halves of all frames and accordingly filter frames with no cars; for Figure 4(b), a ranker may consume a smaller bounding box where 80% persons appear and rank frames based on their likelihood of containing more persons. Figure 6 shows that, by zooming into smaller regions, operators run faster and deliver higher accuracy. By varying input region locations/sizes, DIVA produces a set of operators with diverse costs/accuracies. By controlling the execution order of operators, DIVA processes "popular" frame regions prior to "unpopular" regions. DIVA never omits any region when it executes a query to completion to guarantee correctness.

What happens to instances uncaptured by landmarks?

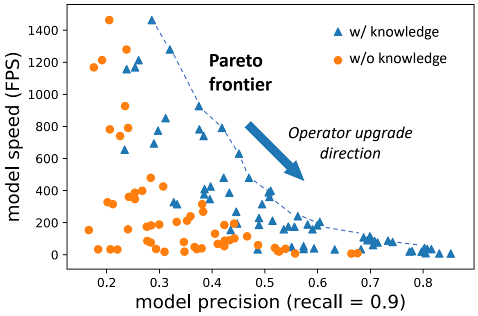


Figure 6: **On-camera operators benefit from long-term video knowledge substantially**. Each marker: an operator. For querying buses on video Banff (see Table 4).

Sparse by design, landmarks are not meant to capture all object instances; instead, they are used as inexact estimators and initial training samples. Reducing landmarks will degrade query speed, as we will experimentally quantify in §8.3. Doing so, however, does not affect query correctness or accuracy: the instances uncaptured by landmarks will be eventually processed by DIVA as a query goes on.

5 Online Operator Upgrade

5.1 The rationale

Three factors determine a query's execution speed:

1. *Pending workloads*: the difficulty of the frames to be processed, i.e., how likely will the frame be mis-filtered or misranked on camera.

2. *Camera operators*: cheap operators spend less time on each frame but are more likely to mis-filter/mis-rank frames, especially difficult frames. This is shown in Figure 6.

3. *Network condition*: the available uplink bandwidth.

The three factors interplay as follows.

• **Queries executed with on-camera *rankers*** A camera ranks and uploads frames asynchronously (§3). The key is to maximize the rate of true positive frames arriving at the cloud, for which the system must balance ranking speed/accuracy with upload bandwidth. (1) When the camera runs a *cheaper* ranker, it ranks frames at a much higher rate than uploading the frames; as a result, the cloud receives frames *hastily* selected from a *wide* video timespan. (2) When the camera runs an *expensive* ranker, the cloud receives frames selected *deliberately* from a *narrow* timespan. (3) The camera should never run rankers slower than upload, which is as bad as uploading unranked frames.

As an example, E_{CHEAP} and E_{EXP} on the top of Figure 7 compare two possible executions of the same query, running cheap/expensive rankers respectively. Shortly after the query starts (❶), E_{CHEAP} swiftly explores more frames on camera; it outperforms E_{EXP} by discovering and returning more true positive frames. As both executions proceed to harder frames (❷), E_{CHEAP} makes more mistakes in ranking; it uploads an increasingly large ratio of negatives which wastes the execution time. By contrast, E_{EXP} ranks frames slower yet with much fewer mistakes, hence uploading fewer negatives. It eventually returns all positives earlier than E_{CHEAP} (❸).

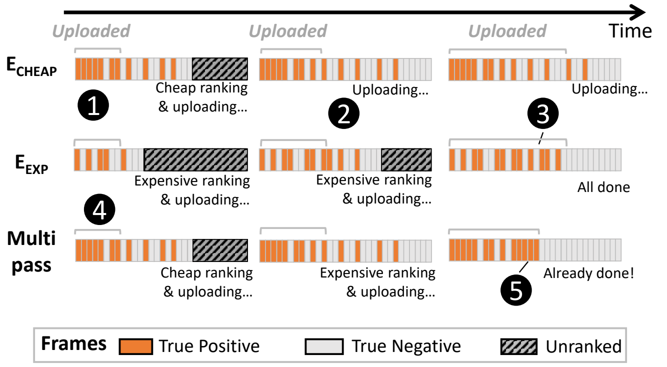

Time
Uploaded
Uploaded
Uploaded
E_CHEAP
❶
Cheap ranking & uploading...
❷
Uploading...
❸
Uploading...
E_EXP
❹
Expensive ranking & uploading...
Expensive ranking & uploading...
All done
Multi pass
Cheap ranking & uploading...
Expensive ranking & uploading...
❺
Already done!
Frames
True Positive
True Negative
Unranked

Figure 7: **Three alternative executions of a Retrieval query, showing multipass ranking (bottom) outperforms running individual rankers alone (top two)**. Each row: snapshots of the upload queue at three different moments. In a queue: ranking/uploading frames from left to right.

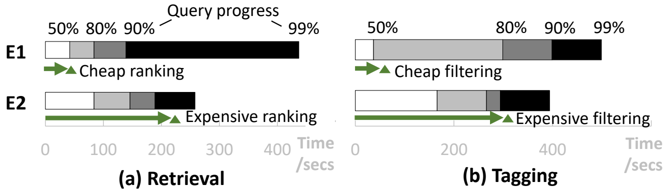


Figure 8: **Cheap/expensive camera operators excel at different query stages**. Each subfigure: two alternative executions of the same query, showing query progress (bars) and the corresponding operator's progress (arrows).

The microbenchmark in Figure 8(a) offers quantitative evidence. E1 spends *less* time (0.7×) in returning the first 90% positives, but *more* time (1.7×) in returning 99% positives. Furthermore, *lower* upload bandwidth favors a more *expensive* ranker, as the uploaded frames would contain a *higher* ratio of positives, better utilizing the precious bandwidth.

• **Queries executed with on-camera *filters*** The key is to maximize the rate of resolving frames on camera. Cheap filters excel on easy frames, resolving these frames fast with confidence. They are incapable on difficult frames, wasting time on attempting frames without much success in resolving. They would underperform *expensive* filters that spend more time per frame yet being able to resolve more frames.

The benchmark in Figure 8(b) shows two executions with cheap/expensive filters. Early in the query, E1 makes faster progress as the camera quickly resolves 50% of the frames (4× less time than E2). Later in the execution, E1 lags behind as the camera cannot resolve the remaining frames and must upload them. By contrast, E2 resolves 82% of frames on camera and only uploads the remaining 18%. As a result, E2 takes 1.3× less time in completing 90% and 99% of the query.

Summary & implications It is crucial for DIVA to pick operators with optimal cost/accuracy at query time. The choice not only varies across queries but also varies throughout a query's execution: easy frames are processed early, leaving increasingly difficult frames that call for more expensive operators. DIVA should monitor pending frame difficulty and network bandwidth and upgrade operators accordingly.

5.2 Multipass, multi-operator execution

DIVA manages operators with the following techniques. (1) A camera processes frames iteratively with multiple operators. (2) The cloud progressively updates operators on camera, from cheaper ones to more expensive ones, as the direction shown in Figure 6. In picking operators, the cloud dynamically adapts operator speed to frame upload speed. (3) The cloud uses frames received in early execution stages to train operators for later stages; as the latter operators are more expensive, they require more training samples.

- ***Multipass ranking*** This is exemplified by the bottom execution in Figure 7. The camera first runs a cheap ranker, moving positives towards the front of the upload queue (❹). Subsequently, the camera runs an expensive ranker, continuously reordering unsent frames in a smaller scope (❺). Throughout the query, the camera first quickly uploads easy frames that are quickly ranked and slows down to vet difficult frames with expensive/accurate ranking. Notably, the cheaper ranker roughly prioritizes the frames as input for the expensive ranker, ensuring the efficacy of the expensive ranker. In actual query executions, a camera switches among 4–8 operators (§8).
- ***Multipass filtering*** The camera sifts undecided, unsent frames in multiple passes, each with a more expensive filter over a sample of the remaining frames. Throughout one query, early, cheaper filters quickly filter easier frames, leaving more difficult frames for subsequent filters to resolve.

6 Query Execution Planning

DIVA plans a concrete query execution by (1) the camera's policy for selecting frames to process; (2) the cloud's policy for upgrading on-camera operators. We now discuss them.

6.1 Executing *Retrieval* queries

Policy for selecting frames To execute the initial operator, the camera prioritizes fixed-length video spans (e.g., 1 hour) likely rich in positive frames, estimated based on landmark frames. In executing subsequent operators, the camera processes frames in their existing ranking as decided by earlier operators, as described in §5. The camera gives opportunities to frames never ranked by prior operators, interleaving their processing with ranked frames with mediocre scores (0.5).

Policy for operator upgrade As discussed in §3, DIVA switches from cheap operators to expensive ones, and matches operator speed to frame upload rate. To capture an operator *op*'s *relative* speed to upload, it uses one simple metric: the ratio between the two speeds, i.e., $f_{op} = FPS_{op}/FPS_{net}$. Operators with higher f_{op} tend to rapidly *explore* frames while others tend to *exploit* slowly. The operator speed FPS_{op} is profiled offline. ***(1) Selecting the initial operator*** In general, DIVA should fully utilize the upload bandwidth with positive frames. As positive frames are scattered in the queried video initially, the camera should explore all frames sufficiently fast. Otherwise, it would either starve the uplink or knowingly upload negative frames. Based on this idea, the cloud picks the most accurate operator from the ones that are fast enough, i.e., $f_{op} \times R_{pos} > 1$, where R_{pos} is the ratio of positives in the queried video, estimated from landmarks. ***(2) When to upgrade: current operator losing its vigor*** The cloud upgrades operators either when the current operator finishes processing all frames, or the cloud observes a continuous quality decline in recently uploaded frames, an indicator of the current operator's incapability. To decide the latter, DIVA employs a rule: the positive ratio in recently uploaded frames are $k\times$ (default: 5) lower than the frames uploaded at the beginning; ***(3) Selecting the next operator: slow down exponentially*** Since the initial operator promotes many positives towards the front of the upload queue, subsequent operators, scanning from the queue front, likely operate on a larger fraction of positives. Accordingly, the cloud picks the most accurate operator among much slower ones, s.t. $f_{op(i+1)} > \alpha \times f_{op(i)}$, where α controls speed decay in subsequent operators. A larger α leads to more aggressive upgrade: losing more speed for higher accuracy. In the current prototype, we empirically choose $\alpha = 0.5$. Since f is relative to FPS_{net} measured at every upgrade, the upgrade adapts to network bandwidth change *during* a query.

6.2 Executing *Tagging* queries

Recall that for ***Tagging***, a camera runs multipass filtering; the objective of each pass is to tag, as positive (P) or negative (N), at least one frame from every K adjacent frames. We call K the group size; DIVA pre-defines a sequence of group sizes as refinement levels, e.g., K = 30, 10, ..., 1. As in prior work [51, 58, 59], the user specifies tolerable error as part of her query, e.g., 1% false negative and 1% false positive; DIVA trains filters with thresholds to meet the accuracy.

Policy for selecting frames The goal is to quickly tag easy frames in individual groups while balancing the workloads of on-camera processing and frame upload. An operator *op* works in two stages of each pass. i) *Rapid attempting*. *op* scans all the groups; it attempts one frame per group; if it succeeds, it moves to the next group; it adds undecidable frames (U) to the upload queue. ii) *Work stealing*. *op* steals work from the end of upload queue. For an undecidable frame f belonging to a group g, *op* attempts other untagged frames in g; once it succeeds, it removes f from the upload queue as f no longer needs tagging in the current pass. After one pass, the camera switches to the next refinement level (e.g., 10 → 5). It keeps all the tagging results (P,N,U) while cancels all pending uploads. It re-runs the frame scheduling algorithm until it meets the finest refinement level or query terminated.

Policy for operator upgrade Given an operator *op* and γ_{op}, the ratio of frames it can successfully tag, DIVA measures *op*'s efficiency by its effective tagging rate, $FPS_{op} \times \gamma_{op} + FPS_{net}$, as a sum of *op*'s successful tagging rate and the uploading rate. As part of operator training, the cloud estimates γ_{op} for all the candidate operators by testing them on all landmarks (early in query) and uploaded frames (later in query). To select every operator, initial or subsequent, the cloud picks the candidate with the highest effective tagging rate. The cloud upgrades operators either when the current operator has attempted all remaining frames or another candidate having an effective tagging rate β× or higher (default value 2).

6.3 Executing *Counting* queries

Max Count: Policy for selecting frames To execute the initial operator, the camera randomly selects frames to process, avoiding the worst cases that the max resides at the end of the query range. For subsequent operators, the camera processes frames in existing ranking decided by earlier operators.

Max Count: Policy for operator upgrade As the camera runs rankers, the policy is similar to that for *Retrieval* with a subtle yet essential difference. To determine whether the current operator shall be replaced, the cloud must assess the quality of recently uploaded frames. While for *Retrieval*, DIVA conveniently measures the quality as the ratio of *positive* frames, the metric does not apply to *max Count*, which seeks to discover *higher* scored frames. Hence, DIVA adopts the Manhattan distance as a quality metric among the permutations from the ranking of the uploaded frames (as produced by the on-camera operator) and the ranking that is re-computed by the cloud object detector. A higher metric indicates worse quality hence more urgency for the upgrade.

Average/Median Count: no on-camera operators After the initial upload of landmarks, the camera randomly samples frames in queried videos and uploads them for the cloud to refine the average/median statistics. To avoid any sampling bias, the camera does not prioritize frames; it instead relies on the Law of Large Numbers (LLN) [48] to approach the average/median ground truth through continuous sampling.

Cameras	**Rpi3** (default): Raspberry Pi 3 **($35)**. 4xCortex-A53, 1GB DRAM **Odroid**: XU4 (**$49**) 4xCortexA15 & 4xCortexA7, 2GB DRAM
CloudServer	2x Intel Xeon E5-2640v4, 128GB DRAM GPU: Nvidia Titan V

(a) *Hardware platforms*

	Cam:Landmarks	**Cam:Query**	**Cloud:Query**
ClondOnly	–	Only upload frames	Yv3 on all uploaded frames
OptOp	Yv3 every 30 secs	Run one optimal op	
PreIndexAll	YTiny every sec	Parse YTiny result	
DIVA	Yv3 every 30 secs	Multi passes & ops	

(b) *DIVA and the baselines*. The table summarizes their executions for capture and query. NNs: Yv3 – YOLOv3, high accuracy (mAP=57.9); YTiny – YOLOv3-tiny, low accuracy (mAP=33.1).

Table 3: **Experiment configurations**

	Name	**Object**	**Description**
T	JacksonH [25]	car	A busy intersection in Jackson Hole, WY
	JacksonT [26]	car	A night street in Jackson Hole, WY
	Banff [20]	bus	A cross-road in Banff, Alberta, Canada
	Mierlo [29]	truck	A rail crossing in Netherlands
	Miami [28]	car	A cross-road in Miami Beach, FL
	Ashland [19]	train	A level crossing in Ashland, VA
	Shibuya [31]	bus	An intersection in Shibuya (渋谷), Japan
O	Chaweng [22]	bicycle	Absolut Ice Bar (outside) in Thailand
	Lausanne [27]	car	A pedestrian plaza in Lausanne, Switzerland
	Venice [32]	person	A waterfront walkway in Venice, Italy
	Oxford [30]	bus	A street beside Oxford Martin school, UK
	Whitebay [33]	person	A beach in Virgin Islands
I	CoralReef [23]	person	An aquarium video from CA
	BoatHouse [21]	person	A retail store from Jackson Hole, WY
W	Eagle [24]	eagle	A tree with an eagle nest in FL

Table 4: **15 videos used for test**. Each video: 720P at 1FPS lasting 48 hours. Column 1: video type. T – traffic; O/I – outdoor/indoor surveillance; W – wildlife.

7 Implementation and Methodology

Operators We architect on-camera operators as variants of AlexNet [63]. We vary the number of convolutional layers (2–5), convolution kernel sizes (8/16/32), the last dense layer's size (16/32/64); and the input image size (25×25/50×50/100×100). We empirically select 40 operators to be trained by DIVA online; we have attempted more but see diminishing returns. These operators require small training samples (e.g., 10K images) and run fast on camera.
Background subtraction filters static frames at low overhead [51]. DIVA employs a standard technique [12]: during video capture, a camera detects frames that have little motion (< 1% foreground mask) and omits them in query execution. On our camera hardware (Table 3), background subtraction is affordable in real time during capture. For fair comparisons, we augment all baselines with background subtraction.

Videos & Queries We test DIVA on 15 videos captured from 15 live camera feeds (Table 4). Each video lasts continuous 48 hours including daytime and nighttime, collected between Oct. 2018 to Mar. 2019. We preprocess all videos to be 720P at 1 FPS, consistent with prior work [51]. We test *Retrieval*/*Tagging*/*Counting* queries on 6/6/3 videos. We intentionally choose videos with disparate characteristics and hence different degrees of difficulty. For instance, Whitebay is captured from a close-up camera, containing clear and large persons; Venice is captured from a high camera view and hence contains blurry and small persons. For each video, we pick a representative object class to query; across videos, these classes are diverse. For *Tagging*, we set query error to be $< 1\%$ FN/FP as prior work did [59].

A query's accuracy is reflected by its execution progress. For retrieval/counting, we report accuracy as the fraction of positive frames returned. There is no false positive because the cloud always runs the high-accuracy object detector as a "safety net", of which the output is regarded as the ground truth. For tagging, we report accuracy as query errors, meaning the percentage of frames mistakenly tagged. To issue a query, the user sets the target error, which by default is 1% as in prior work. Table 2 and §6 provide more details.

Test platform & parameters As summarized in Table 3(a), we test on embedded hardware similar to low-cost cameras [15, 16]. We use Rpi3 as the default camera hardware and report its measurement unless stated otherwise. During query execution, both devices set up a network connection with 1MB/s default bandwidth to emulate typical WiFi condition [47]. Note that this network bandwidth is *not* meant for streaming; it is only for a camera while the camera is being queried. We run YOLOv3 as the high-accuracy object detector on camera and cloud (Table 3(b)). In calculating accuracy, we use the output of YOLOv3 as the ground truth as in prior work [51, 59]. On Rpi3, we partition YOLOv3 into three stages, each fitting into DRAM separately. We will study alternative models, landmarks, and resources in §8.3.

Baselines As summarized in Table 3(b), we compare DIVA with three alternative designs augmented with background subtraction and only process/transmit non-static video frames.

- **`CloudOnly`**: a naive design that uploads all queried frames at query time for the cloud to process.
- **`OptOp`**: in the spirit of NoScope [59], the camera runs only one ranker/filter specialized for a given query, selected by a cost model for minimizing full-query delay. To make `OptOp` competitive, we augment it with landmark frames to reduce the operator training cost. Compared to DIVA, `OptOp`'s key differences are the lack of operator upgrade and the lack of operator optimization by long-term video knowledge.
- **`PreIndexAll`**: in the spirit of Focus [51], the camera runs a cheap yet generic object detector on all frames. We pick YOLOv3-tiny (much cheaper than YOLOv3) as the detector affordable by Rpi3 in real time (1 FPS). The detector plays the same role as an operator in DIVA, except that it runs at capture time: for *Retrieval* and *Counting*, the detector's output scores are used to prioritize frames to upload at query time; for *Tagging*, the output is used to filter the frames that have enough confidence. `PreIndexAll` implements all runtime features of Focus except feature clustering. We left out clustering because we find it performs poorly on counting queries. Compared to DIVA, `PreIndexAll`'s key differences are: it answers queries solely based on the indexes built at capture time; it requires no operator training or processing actual images at query time.

8 Evaluation

8.1 End-to-end performance

Full query delay is measured as: (*Retrieval*) the time to receive 99% positive frames as in prior work [51]; (*Tagging*) the time taken to tag every frame; (*Counting*) the time to reach the ground truth (max) or converge within 1% error of the ground truth (avg/median). Overall, DIVA delivers good performance and outperforms the baselines significantly.

- *Retrieval* (Figure 9(a)). On videos each lasting 48 hours, DIVA spends ~1,900 seconds on average, i.e., 89× of video realtime. On average, DIVA's delay is 3.8×, 3.1×, and 2.0× shorter than `CloudOnly`, `PreIndexAll`, and `OptOp`, respectively.
- *Tagging* (Figure 9(b)). DIVA spends ~581 seconds on average (297× realtime). This delay is 16.0×, 2.1×, and 4.3× shorter than `CloudOnly`, `PreIndexAll`, and `OptOp`, respectively.
- *Counting* (Figure 10). DIVA's average/median take several seconds to converge. For *average Count*, DIVA's delay is 65.1× and up to three orders of magnitude shorter than `CloudOnly` and `PreIndexAll`. For *median Count*, DIVA's delay is 68.3× shorter than the others. For *max Count*, DIVA spends 34 seconds on average (635× realtime), which is 5.8×, 5.0×, and 1.3× shorter than `CloudOnly`, `PreIndexAll`, and `OptOp`.

Query progress DIVA makes much faster progress in most time of query execution. It *always* outperforms `CloudOnly` and `OptOp` during *Retrieval/Tagging* (Figure 9). It *always* outperforms alternatives in median/average count (Figure 10).

Why DIVA outperforms the alternatives? The alternatives suffer from the following. (1) Inaccurate indexes. `PreIndexAll` resorts to inaccurate indexes (YOLOv3-tiny) built at capture time. Misled by them, *Retrieval* and *Tagging* upload too much garbage; *Counting* includes significant errors in the initial estimation, slowing down convergence. (2) Lack of long-term knowledge. `OptOp`'s operators are either slower or less accurate than DIVA, as illustrated in Figure 6. (3) One operator does not fit an entire query. Optimal at some point (e.g., 99% Retrieval), the operator runs too slow on easy frames which could have been done by cheaper operators.

Why DIVA underperforms (occasionally)? On short occasions, DIVA may underperform `PreIndexAll` at early query stages, e.g., BoatHouse in Figure 9. Reasons: (1) `PreIndexAll`'s inaccurate indexes may be correct on *easy* frames; (2) `PreIndexAll` does not pay for operator bootstrapping as DIVA. Nevertheless, `PreIndexAll`'s advantage is transient. As easy frames are exhausted, indexes make more mistakes on the remaining frames and hence slow down the query.

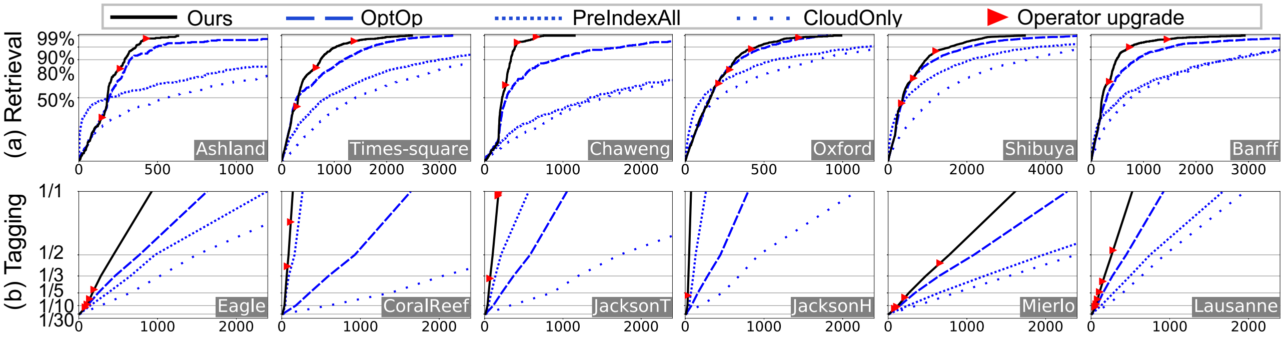


Figure 9: **On *Retrieval* and *Tagging* queries, DIVA shows good performance and outperforms the alternatives**. x-axis for all: query delay (secs). y-axis for (a): % of retrieved instances; for (b): refinement level (1/N frames).

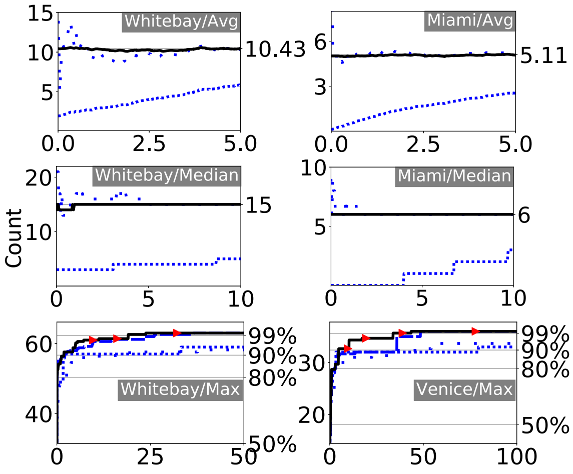


Figure 10: **On *Counting* queries, DIVA shows good performance and outperforms the alternatives**. Legend: see Figure 9. x-axis for all: query delay (secs). y-axis for left plots: count; for top two right plots: ground truth for avg/median queries; for bottom right plot: % of max value.

Can DIVA outperform under different network bandwidths at query time? Table 5 summarizes DIVA's query delays at 9 bandwidths evenly spaced in [0.1 MB/s, 10 MB/s] which cover typical WiFi bandwidths [9]. We have observed that: on lower bandwidths, DIVA's advantages over baselines are more significant; at high bandwidths, DIVA's advantages are still substantial (>2× in most cases) yet less pronounced. The limitation is not in DIVA's design but rather its current NNs: we find it difficult to train operators that are both fast enough to keep up with higher upload bandwidth and accurate enough to increase the uploaded positive ratio proportionally.

vs. "all streaming": query speed As "all streaming" uploads all videos to the cloud before a query starts, the query speed is bound by cloud GPUs but not network bandwidth. With our default experiment setting (1 GPU and 1MB/s network bandwidth), "all streaming" still runs queries much slower than zero streaming. Adding more cloud GPUs will eventually make "all streaming" run faster than DIVA.

	Retrieval	*Tagging*	*Count/Max*	*Count/Avg&Med*
CloudOnly	4.5/14.9/52.2	3.61/3.9/5.1	2.8/21.1/42.5	6.9/83.4/439.2
OptOp	2.2/4.1/4.9	2.0/2.3/2.6	1.2/1.5/2.1	6.9/83.4/439.2*
PreIndexAll	1.9/3.8/11.6	3.2/3.6/4.9	1.2/8.9/18.2	2.5/14.0/41.3

*: Fall back to CloudOnly as the camera does not execute NN for these query types

Table 5: DIVA' performance (speedup) with various bandwidths. Numbers: min/median/max of times (×) of query delay reduction compared to baselines (rows). Averaged on all videos and 9 bandwidths in 0.1MB/s–10MB/s.

vs. "all streaming": network bandwidth saving Compared to streaming all videos (720P 1FPS) at capture time, DIVA saves traffic significantly, as shown in Figure 11. When only as few as 0.005% of video is queried as in our case study (§2), the saving is over three orders of magnitude. Even if all captured videos are queried, DIVA saves more than 10×, as its on-camera operators skip uploading many frames. Among the bandwidth reduction brought by DIVA, only less than 30% attributes to the background subtraction technique. It shows that the disadvantage of "all streaming" is fundamental: streaming optimizations may help save the bandwidth (upmost several times [96]) but cannot offset the waste, as discussed in §2.3.

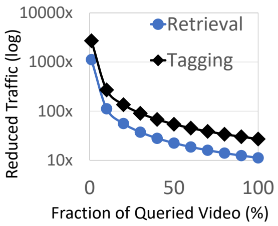


Figure 11: **DIVA significantly reduces network traffic compared to "all streaming".** Results averaged over all videos.

Training & shipping operators For each query, DIVA trains ~40 operators, of which ~10 are on the Pareto frontier. The camera switches among 4–8 operators, which run at diverse speeds (27×–1,000× realtime) and accuracies. DIVA chooses very different operators for different queries. Training one operator typically takes 5–45 seconds on our test platform and requires 5k frames (for bootstrapping) to 15k frames (for stable accuracy). Operators' sizes range from 0.2–15 MB. Sending an operator takes less than ten seconds. Only the delay in training and sending the first operator ($\leq$ 40 seconds) adds to the query delay which is included in Figure 9/10.

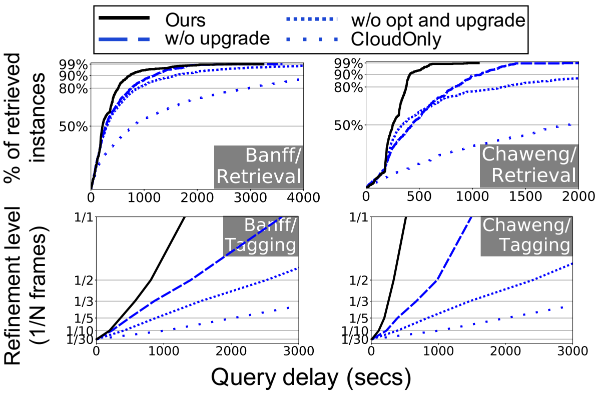
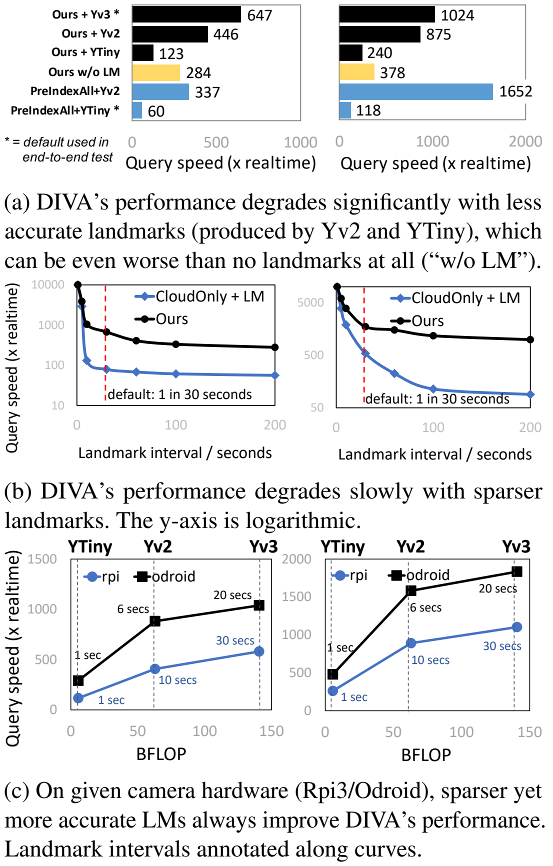


Figure 12: DIVA's both key techniques – optimization with long-term video knowledge (opt) and operator upgrade (upgrade), contribute to performance significantly.



(a) DIVA's performance degrades significantly with less accurate landmarks (produced by Yv2 and YTiny), which can be even worse than no landmarks at all ("w/o LM").

(b) DIVA's performance degrades slowly with sparser landmarks. The y-axis is logarithmic.

(c) On given camera hardware (Rpi3/Odroid), sparser yet more accurate LMs always improve DIVA's performance. Landmark intervals annotated along curves.

Figure 13: **Validation of landmark design**. In (a)/(b)/(c): Left – *Retrieval* on Chaweng; Right – *Tagging* on JacksonH.

Subsequent operators are trained and transmitted in parallel to query execution. Their delays are hidden from users.

DIVA elasticity Due to DIVA's design, the computing resources available on low-cost cameras are used efficiently at both capture and query time. Thanks to its elastic execution, it can avoid interference with a camera's surveillance task, notably video encoding and storage. For instance, DIVA can produce denser/sparser landmarks per its CPU time allocated by the camera OS. According to our experiments on Raspberry Pi 3B+, recording video at 720P and 30 FPS only uses less than 2% of CPU time, which is negligible as compared to NN execution. We reserve a small fraction of CPU time to surveillance using *cgroup* and observe no frame drop in the surveillance task and negligible slowdown in NN execution.

8.2 Validation of query execution design

The experiments above show DIVA's substantial advantage over `OptOp`, coming from a combination of two techniques – optimizing queries with long-term video knowledge ("*Long-term opt*", §4) and operator upgrade ("*Upgrade*", §5). We next break down the advantage by incrementally disabling the two techniques in DIVA. Figure 12 shows the results.

Both techniques contribute to significant performance. For instance, disabling *Upgrade* increases the delay of retrieving 90% instances by 2× and that of tagging 1/1 frames by 2×-3×. Further disabling *Long-term Opt* increases the delay of Retrieval by 1.3×-2.1× and that of tagging by 1.6×-3.1×. Both techniques disabled, DIVA still outperforms `CloudOnly` with its single non-optimized operator.

***Upgrade*'s benefit is universal; *Long-term opt*'s benefit is more dependent on queries**, i.e., the skews of the queried object class in videos. For instance, DIVA's benefit is more pronounced on Chaweng, where small bicycles only appear in a region in 1/8 size of the entire frame, than Ashland, where large trains take 4/5 of the frame. With stronger skews in Chaweng, DIVA trains operators that are more accurate and run faster. This also accounts for DIVA's varying (yet substantial) advantages over the alternatives (Figure 9).

8.3 Validation of landmark design

Next, we deviate from the default landmark parameters (Table 3) to validate the choice of sparse-but-sure landmarks.

DIVA hinges on accurate landmarks. As shown in Figure 13(a), modestly inaccurate landmarks (as produced by YOLOv2; 48.1 mAP) increase delays for Q1/Q2 by 45% and 17%. Even less accurate landmarks (by YOLOv3-tiny; 33.1 mAP) increase the delays significantly by 5.3× and 4.3×. Perhaps surprisingly, such inaccurate landmarks can be worse than no landmarks at all ("w/o LM" in Figure 13): when a query starts, a camera randomly uploads unlabeled frames for the cloud to bootstrap operators. *Why inaccurate landmarks hurt so much?* They (1) provide wrong training samples; (2) lead to incorrect observation of spatial skews which further mislead frame cropping; and (3) introduce large errors into initial statistics, making convergence harder.

DIVA tolerates longer landmark intervals. As shown in Figure 13(b), DIVA's *Retrieval* and *Tagging* performance slowly degrade with longer intervals. Even with an infinite interval, i.e., “w/o LM” in Figure 13(a), the slowdown is no more than 3×. On *Counting*, the performance degradation is more pronounced: 5× longer intervals for around 15× slow down. Yet, such degradation is still much smaller than one from inaccurate landmarks (two orders of magnitude). The reason is that, with longer LM intervals DIVA has to upload additional frames in full resolution (∼10× larger than LMs) when a query starts for bootstrapping operators; such a one-time cost, however, is amortized over the full query.

Create the most accurate landmarks possible Should a camera build denser yet less accurate landmarks or sparser yet more accurate ones? Figure 13(c) suggests the latter is always preferred, because of DIVA’s high sensitivity to landmark accuracy and low sensitivity to long landmark intervals.

DIVA on wimpy/brawny cameras DIVA suits wimpy cameras that can only generate sparse landmarks. Some cameras may have DRAM smaller than a high-accuracy NN (e.g., ∼1 GB for YOLOv3); fortunately, recent orthogonal efforts reduce NN sizes [56]. Wimpier cameras will further disadvantage the alternatives, e.g., `PreIndexAll` will produce even less accurate indexes. On higher-end cameras (a few hundred dollars each [13]) that DIVA is *not* designed for, DIVA still shows benefits, albeit not as pronounced. High-end cameras can afford more computation at capture time. i) They may run `PreIndexAll` with improved index accuracy. In Figure 13(a), running YOLOv2 on all captured frames (`PreIndexAll`+Yv2), DIVA’s performance gain is 1.9× (left) or even 0.6× (right). ii) These cameras may generate denser landmarks and rely on the cloud for the remaining frames. Figure 13(b) shows, with one landmark every 5 seconds, DIVA’s advantage is 1.5×.

9 Related Work

Optimizing video analytics The CV community has studied video analytics for decades, e.g., for online training [83, 84] and active learning [57]. They mostly focus on improving analytics accuracy on short videos [44, 60, 68, 78, 81, 102] while missing opportunities in exploiting long-term knowledge (§4). These techniques alone cannot address the systems challenges we face, e.g., network limit or frame scheduling. A common theme of recent work is to trade accuracy for lower cost: VStore [96] does so for video storage; Pakha *et al.* [70] do so for network transport; Chameleon [55] and VideoStorm [52, 99] do so with video formats. DIVA’s operators as well exploit accuracy/cost tradeoffs. Multiple systems analyze archival videos on servers [58, 62, 73, 80, 96]. DIVA analyzes archival videos on *remote* cameras and embraces new techniques. ML model cascade is commonly used for processing a stream of frames [39, 59, 85]: in processing a frame, it keeps invoking a more expensive operator if the current operator has insufficient confidence. This technique, however, mismatches exploratory analytics, for which DIVA uses one operator to process many frames in one pass and produces inexact yet useful results for all of them.

Edge video analytics To reduce cloud/edge traffic, computation is partitioned, e.g., between cloud/edge [40, 76, 97], edge/-drone [91], and edge/camera [100]. Elf [94] executes counting queries completely on cameras. Most work targets live analytics, processes frames in a streaming fashion and trains NNs ahead of time. DIVA spreads computation between cloud/-cameras but takes a disparate design point (zero streaming) that are inadequate in prior systems. CloudSeg [92] reduces network traffic by uploading low-resolution frames and recovering them via super resolution. DIVA eliminates network traffic at capture time at all.

Online Query Processing Dated back in the 90s, online query processing allows users to see early results and control query execution [49, 50]. It is proven effective in large data analytics, such as MapReduce [43]. DIVA retrofits the idea for video queries and accordingly contributes new techniques, e.g., operator upgrade, to support the online fashion. DIVA could borrow UI designs from existing online query engines.

WAN Analytics To query geo-distributed data, recent proposals range from query placement to data placement [74, 86–88, 90]. JetStream [75] adjusts data quality to meet network bandwidth; AWStream [98] facilitates apps to systematically trade-off analytics accuracy for network bandwidth. Like them, DIVA adapts to network; unlike them, DIVA does so by changing operator upgrade plan, a unique aspect in video analytics. DIVA targets resource-constrained cameras, which are unaddressed in WAN analytics.

10 Conclusions

Zero streaming shifts most compute from capture time to query time. We build DIVA, an analytics engine for querying cold videos on remote, low-cost cameras. At capture time, DIVA builds sparse but sure landmarks; at query time, it refines query results by continuously updating on-camera operators. Our evaluation of three types of queries shows that DIVA can run at more than 100× video realtime under typical wireless network and camera hardware.

Acknowledgments

Mengwei Xu was supported by National Key R&D Program of China under grant number 2020YFB1805500, the Fundamental Research Funds for the Central Universities, and National Natural Science Foundation of China under grant numbers 62032003, 61922017, and 61921003. Tiantu Xu and Felix Xiaozhu Lin were supported in part by NSF awards #1846102 and #1919197. We thank our shepherd, Michael Kozuch, and the anonymous ATC reviewers for their useful suggestions.

References

[1] The european data protection supervisor video-surveillance guidelines. https://edps.europa.eu/sites/edp/files/publication/10-03-17_video-surveillance_guidelines_en.pdf, 2010.

[2] Tufts: Video security university policy. https://publicsafety.tufts.edu/policies/video-security/, 2014.

[3] Wireless cameras slowing router too much. https://community.netgear.com/t5/Nighthawk-WiFi-Routers/Wireless-cameras-slowing-router-too-much/td-p/513047, 2015.

[4] Understanding ip surveillance camera bandwidth. https://www.fortinet.com/content/dam/fortinet/assets/white-papers/wp-ip-surveillance-camera.pdf, 2017.

[5] The zettabyte era: Trends and analysis. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html, 2017.

[6] International trends in video surveillance. https://cms.uitp.org/wp/wp-content/uploads/2020/06/18-07Statistics-Brief-Videosurveillance-web.pdf, 2018.

[7] New case law on retention periods for video surveillance at the workplace. https://www.twobirds.com/en/news/articles/2018/germany/new-case-law-on-retention-periods-for-video-surveillance-at-the-workplace, 2018.

[8] Running yolo detection on raspberry pi. http://raspberrypi4u.blogspot.com/2018/10/raspberry-pi-yolo-real-time-object.html, 2018.

[9] The state of wifi vs mobile network experience as 5g arrives. https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2018-11/state_of_wifi_vs_mobile_opensignal_201811.pdf, 2018.

[10] Video surveillance laws: Video retention requirements by state. https://www.verkada.com/blog/surveillance-laws-video-retention-requirements/, 2018.

[11] Wifi cameras. https://www.security-camera-warehouse.com/ip-camera/wifi-enabled/, 2018.

[12] Background subtraction. https://docs.opencv.org/3.4.0/db/d5c/tutorial_py_bg_subtraction.html, 2019.

[13] Build intelligent ideas with our platform for local ai. https://coral.withgoogle.com/, 2019.

[14] Comcast business internet data plan. https://www.business.org/services/internet/comcast-business-internet-review/, 2019.

[15] Hisilicon ip camera specifications. http://www.hisilicon.com/en/Products/ProductList/Surveillance, 2019.

[16] Wyze camera specifications. https://www.wyze.com/wyze-cam/specs/, 2019.

[17] Wyze camera v2 1080p. https://www.wyze.com/product/wyze-cam-v2/, 2019.

[18] Yi home camera. https://www.amazon.com/YI-Security-Surveillance-Monitor-Android/dp/B01CW4AR9K, 2019.

[19] Youtube live streaming: Ashland. https://www.youtube.com/watch?v=e47XhLmZhFk, 2019.

[20] Youtube live streaming: Banff. https://youtu.be/9HwSNgcdQ7k, 2019.

[21] Youtube live streaming: Boathouse. https://www.youtube.com/watch?v=TXw7CyY0TbU&t=0s, 2019.

[22] Youtube live streaming: Chaweng. https://www.youtube.com/watch?v=tihJ58_qiH0, 2019.

[23] Youtube live streaming: Coralreef. https://youtu.be/WYOe8SfQbac, 2019.

[24] Youtube live streaming: Eagle. https://www.youtube.com/watch?v=Q_OrM8o2k6I, 2019.

[25] Youtube live streaming: Jackson hole. https://youtu.be/2wnU2Kp7quQ, 2019.

[26] Youtube live streaming: Jackson town. https://www.youtube.com/watch?v=1EiC9bvVGnk, 2019.

[27] Youtube live streaming: Lausanne. https://www.youtube.com/watch?v=7uF7DsUQ9vc, 2019.

[28] Youtube live streaming: Miami. https://www.youtube.com/watch?v=0dctq-YjAdc, 2019.

[29] Youtube live streaming: Mierlo. https://www.youtube.com/watch?v=HbtBgxFkDHU, 2019.

[30] Youtube live streaming: Oxford. https://www.youtube.com/watch?v=St7aTfoIdYQ, 2019.

[31] Youtube live streaming: Shibuya. https://youtu.be/PmrWwYTlAVQ, 2019.

[32] Youtube live streaming: Venice. https://www.youtube.com/watch?v=JqUREqYduHw, 2019.

[33] Youtube live streaming: Whitebay. https://www.youtube.com/watch?v=LXWVYoBluT4, 2019.

[34] Zosi camera. https://www.amazon.com/ZOSI-1280TVL-Security-Weatherproof-Surveillance/dp/B01DF6LJZK, 2019.

[35] Aloÿs Augustin, Jiazi Yi, Thomas Clausen, and William Townsley. A study of lora: Long range & low power networks for the internet of things. Sensors, 16(9):1466, 2016.

[36] David Beymer, Philip McLauchlan, Benjamin Coifman, and Jitendra Malik. A real-time computer vision system for measuring traffic parameters. In Proceedings of IEEE computer society conference on computer vision and pattern recognition (CVPR), pages 495–501, 1997.

[37] T. Blu, P. Dragotti, M. Vetterli, P. Marziliano, and L. Coulot. Sparse sampling of signal innovations. IEEE Signal Processing Magazine, 25(2):31–40, March 2008.

[38] Christian Böhm, Bernhard Braunmüller, Florian Krebs, and Hans-Peter Kriegel. Epsilon grid order: An algorithm for the similarity join on massive high-dimensional data. In ACM SIGMOD Record, volume 30, pages 379–388, 2001.

[39] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 3361–3369, 2015.

[40] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G. Andersen, Michael Kaminsky, and Subramanya R. Dulloor. Scaling video analytics on constrained edge nodes. In Proceedings of the 2nd SysML Conference (SysML), 2019.

[41] Kaushik Chakrabarti, Kriengkrai Porkaew, and Sharad Mehrotra. Efficient query refinement in multimedia databases. In ICDE Conference, January 2000. Poster paper.

[42] Tiffany Yu-Han Chen, Lenin S. Ravindranath, Shuo Deng, Paramvir Victor Bahl, and Hari Balakrishnan. Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. In 13th ACM Conference on Embedded Networked Sensor Systems (SenSys), November 2015.

[43] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy, and Russell Sears. Mapreduce online. In Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI), pages 21–21, 2010.

[44] Boyuan Feng, Kun Wan, Shu Yang, and Yufei Ding. SECS: efficient deep stream processing via class skew dichotomy. CoRR, abs/1809.06691, 2018.

[45] Ziqiang Feng, Shilpa George, Jan Harkes, Padmanabhan Pillai, Roberta Klatzky, and Mahadev Satyanarayanan. Eureka: Edge-based discovery of training data for machine learning. IEEE Internet Computing, PP:1–1, 01 2019.

[46] Ziqiang Feng, Junjue Wang, Jan Harkes, Padmanabhan Pillai, and Mahadev Satyanarayanan. Eva: An efficient system for exploratory video analysis. SysML, 2018.

[47] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. Mp-dash: Adaptive video streaming over preference-aware multipath. In Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT), pages 129–143, New York, NY, USA, 2016. ACM.

[48] Michiel Hazewinkel. Encyclopaedia of Mathematics. Springer Netherlands, 1988.

[49] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas. Interactive data analysis: the control project. Computer, 32(8):51–59, Aug 1999.

[50] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD), pages 171–182, New York, NY, USA, 1997. ACM.

[51] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. Focus: Querying large video datasets with low latency and low cost. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA, 2018. USENIX Association.

[52] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodík, Leana Golubchik, Minlan Yu, Victor Bahl, and Matthai Philipose. Videoedge: Processing camera streams using hierarchical clusters. In 2018 IEEE/ACM Symposium on Edge Computing (SEC), 2018.

[53] Ihab F Ilyas, Rahul Shah, Walid G Aref, Jeffrey Scott Vitter, and Ahmed K Elmagarmid. Rank-aware query optimization. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data (ICMD), pages 203–214, 2004.

[54] Samvit Jain, Junchen Jiang, Yuanchao Shu, Ganesh Ananthanarayanan, and Joseph Gonzalez. Rexcam: Resource-efficient, cross-camera video analytics at enterprise scale. CoRR, abs/1811.01268, 2018.

[55] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: Scalable adaptation of video analytics. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM), pages 253–266, New York, NY, USA, 2018. ACM.

[56] Tian Jin and Seokin Hong. Split-cnn: Splitting window-based operations in convolutional neural networks for memory system optimization. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 835–847, 2019.

[57] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. Fine-tuning deep neural networks in continuous learning scenarios. In Chu-Song Chen, Jiwen Lu, and Kai-Kuang Ma, editors, Computer Vision – ACCV 2016 Workshops, pages 588–605, Cham, 2017. Springer International Publishing.

[58] Daniel Kang, Peter Bailis, and Matei Zaharia. Blazeit: Fast exploratory video queries using neural networks. arXiv preprint arXiv:1805.01046, 2018.

[59] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: Optimizing neural network queries over video at scale. Proc. VLDB Endow., 10(11):1586–1597, August 2017.

[60] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 817–825, June 2016.

[61] Nick Koudas and Kenneth C Sevcik. High dimensional similarity joins: Algorithms and performance evaluation. IEEE Transactions on Knowledge and Data Engineering (TKDE), 12(1):3–18, 2000.

[62] Sanjay Krishnan, Adam Dziedzic, and Aaron J. Elmore. Deeplens: Towards a visual data management system. In CIDR 2019, 9th Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings, 2019.

[63] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems (NIPS), pages 1097–1105. Curran Associates, Inc., 2012.

[64] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. International Journal of Computer Vision (IJCV), Aug 2019.

[65] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. Reducto: On-camera filtering for resource-efficient real-time video analytics. In Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM), pages 359–376, 2020.

[66] Mike Liao. Benchmarking hardware for cnn inference in 2018. https://towardsdatascience.com/benchmarking-hardware-for-cnn-inference-in-2018-1d58268de12a, 2018.

[67] Alan J Lipton, Peter L Venetianer, Niels Haering, Paul C Brewer, Weihong Yin, Zhong Zhang, Li Yu, Yongtong Hu, Gary W Myers, Andrew J Chosak, et al. Video analytics for retail business process monitoring, 2015. US Patent 9,158,975.

[68] Mason Liu and Menglong Zhu. Mobile video object detection with temporally-aware feature maps. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[69] NIST. The spectrum crunch. https://www.nist.gov/topics/advanced-communications/spectrum-crunch, 2019.

[70] Chrisma Pakha, Aakanksha Chowdhery, and Junchen Jiang. Reinventing video streaming for distributed vision analytics. In 10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18), Boston, MA, 2018. USENIX Association.

[71] Niketan Pansare, Vinayak R Borkar, Chris Jermaine, and Tyson Condie. Online aggregation for large mapreduce jobs. Proceedings of the VLDB Endowment, 4(11):1135–1145, 2011.

[72] Ziv Paz. Innovation in surveillance: What's changing at the edge, core and cloud? https://blog.westerndigital.com/innovation-surveillance-edge-core-cloud/, year = 2018.

[73] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. Scanner: Efficient video analysis at scale. ACM Trans. Graph., 37(4):138:1–138:13, July 2018.

[74] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. Low latency geo-distributed data analytics. SIGCOMM Comput. Commun. Rev., 45(4):421–434, August 2015.

[75] Ariel Rabkin, Matvey Arye, Siddhartha Sen, Vivek S. Pai, and Michael J. Freedman. Aggregation and degradation in jetstream: Streaming analytics in the wide area. In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), pages 275–288, Seattle, WA, 2014. USENIX Association.

[76] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In IEEE Conference on Computer Communications (INFOCOM), pages 1421–1429, April 2018.

[77] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[78] Venkatesh Saligrama and Zhu Chen. Video anomaly detection based on local statistical aggregates. 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2112–2119, 2012.

[79] Cosma Rohilla Shalizi. Advanced data analysis from an elementary point of view. http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/ADAfaEPoV.pdf, year = 2019.

[80] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. Nexus: a gpu cluster engine for accelerating dnn-based video analysis. In Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP), pages 322–337, 2019.

[81] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy. Fast video classification via adaptive cascading of deep models. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[82] Honghui Shi. Geometry-aware traffic flow analysis by detection and tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pages 116–120, 2018.

[83] Ervin Teng, João Diogo Falcão, and Bob Iannucci. Clickbait: Click-based accelerated incremental training of convolutional neural networks. CoRR, abs/1709.05021, 2017.

[84] Ervin Teng, Rui Huang, and Bob Iannucci. Clickbait-v2: Training an object detector in real-time. CoRR, abs/1803.10358, 2018.

[85] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition (CVPR), 1:511–518, 2001.

[86] Raajay Viswanathan, Ganesh Ananthanarayanan, and Aditya Akella. CLARINET: Wan-aware optimization for analytics queries. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pages 435–450, Savannah, GA, 2016. USENIX Association.

[87] Ashish Vulimiri, Carlo Curino, P. Brighten Godfrey, Thomas Jungblut, Jitu Padhye, and George Varghese. Global analytics in the face of bandwidth and regulatory constraints. In 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), pages 323–336, Oakland, CA, 2015. USENIX Association.

[88] Ashish Vulimiri, Carlo Curino, Philip Brighten Godfrey, Thomas Jungblut, Konstantinos Karanasos, Jitendra Padhye, and George Varghese. Wanalytics: Geo-distributed analytics for a data intensive world. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD), pages 1087–1092, New York, NY, USA, 2015. ACM.

[89] Haizhong Wang, Kimberly Rudy, Jia Li, and Daiheng Ni. Calculation of traffic flow breakdown probability to optimize link throughput. Applied Mathematical Modelling, 34(11):3376 – 3389, 2010.

[90] Hao Wang and Baochun Li. Lube: Mitigating bottlenecks in wide area data analytics. In 9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17), Santa Clara, CA, 2017. USENIX Association.

[91] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In 2018 IEEE/ACM Symposium on Edge Computing (SEC), pages 159–173, 2018.

[92] Yiding Wang, Weiyan Wang, Junxue Zhang, Junchen Jiang, and Kai Chen. Bridging the edge-cloud barrier for real-time advanced vision analytics. In 11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19), 2019.

[93] Slate William Saletan. The case for mass surveillance. https://www.delcotimes.com/news/the-case-for-mass-surveillance/article_61a27a3c-8e8b-54e3-b048-e44682b6a024.html, 2013.

[94] Mengwei Xu, Xiwen Zhang, Yunxin Liu, Gang Huang, Xuanzhe Liu, and Felix Xiaozhu Lin. Approximate query service on autonomous iot cameras. In Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services, pages 191–205, 2020.

[95] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. Deepcache: Principled cache for mobile deep vision. In Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, pages 129–144, 2018.

[96] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. Vstore: A data store for analytics on large videos. In Proceedings of the Fourteenth EuroSys Conference 2019 (EuroSys), pages 16:1–16:17, New York, NY, USA, 2019. ACM.

[97] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li. Lavea: Latency-aware video analytics on edge computing platform. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pages 2573–2574, June 2017.

[98] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A. Lee. Awstream: Adaptive wide-area streaming analytics. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM), pages 236–252, New York, NY, USA, 2018. ACM.

[99] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), pages 377–392, Boston, MA, 2017. USENIX Association.

[100] Tan Zhang, Aakanksha Chowdhery, Paramvir (Victor) Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom), pages 426–438, New York, NY, USA, 2015. ACM.

[101] Hongwei Zhu, Farzin Aghdasi, Greg M Millar, and Stephen J Mitchell. Online learning method for people detection and counting for retail stores, 2017. US Patent 9,639,747.

[102] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7210–7218. IEEE Computer Society, 2018.