Scaling Video Analytics on Constrained Edge Nodes

Christopher Canel [*1] **Thomas Kim** [*1] **Giulio Zhou** [1] **Conglong Li** [1] **Hyeontaek Lim** [1] **David G. Andersen** [1]
Michael Kaminsky [2] **Subramanya R. Dulloor** [3]

Abstract

As video camera deployments continue to grow, the need to process large volumes of real-time data strains wide area network infrastructure. When per-camera bandwidth is limited, it is infeasible for applications such as traffic monitoring and pedestrian tracking to offload high-quality video streams to a datacenter. This paper presents FilterForward, a new edge-to-cloud system that enables datacenter-based applications to process content from thousands of cameras by installing lightweight edge filters that backhaul only relevant video frames. FilterForward introduces fast and expressive per-application "microclassifiers" that share computation to simultaneously detect dozens of events on computationally constrained edge nodes. Only matching events are transmitted to the cloud. Evaluation on two real-world camera feed datasets shows that FilterForward reduces bandwidth use by an order of magnitude while improving computational efficiency and event detection accuracy for challenging video content.

This paper is an extended version of (Canel et al., 2019).

1 Introduction

Video camera deployments in urban areas are ubiquitous: in malls, offices, and homes, and on streets, cars, and people. Almost 100 million networked surveillance cameras were purchased worldwide in 2017 (IHS). Machine learning–based analytics on real-time streams collected by these cameras, such as traffic monitoring, customer tracking, and event detection, promise breakthroughs in efficiency and safety. However, tens of thousands of always-on cameras installed in a modern city collectively generate hundreds of gigabits of data every second, overloading shared network infrastructure. This problem is worse for wirelessly and cellularly–connected nodes and areas outside of infrastructure-rich metropolitan centers (FCC), as they often have more constrained networks (Google Wireless Internet; ITU/UNESCO Broadband Commission for Sustainable Development, 2017). Moreover, the infeasibility of uploading streaming video is at odds with the growing complexity of video analytics applications, which are designed to run in datacenters. This paper addresses the question of how to overcome this network bottleneck and offload large volumes of data from a distributed camera deployment in real time to a datacenter for further processing.

[*]Equal contribution [1]Computer Science Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA [2]Intel Labs, Pittsburgh, Pennsylvania, USA [3]ThoughtSpot, Palo Alto, California, USA. Correspondence to: Christopher Canel <ccanel@cmu.edu>.

Deployment proliferation, combined with increasing camera resolution, necessitates an edge-based filtering approach that is parsimonious with limited bandwidth. We present *FilterForward*, a system that offers the benefits of both edge computing and datacenter-centric approaches to wide-area video processing. Using edge-compute resources collocated with the cameras, FilterForward identifies the video sequences that are most relevant to datacenter applications ("filtering") and offloads only that data for further analysis ("forwarding"). In this way, FilterForward supports near-real-time processing running in datacenters while limiting the use of low-bandwidth wide area network links.

FilterForward is designed for scenarios meeting two key assumptions, which hold for some, though certainly not all, applications. First, relevant events are rare. There is bandwidth to be saved by transmitting only relevant data. Second, datacenter applications require high-quality video data to complete their tasks. This precludes solutions such as heavily compressing streams or reducing their spatial (frame dimensions) or temporal (frame frequency) resolutions.

In the FilterForward model, datacenter applications express interest in specific types of visual content (e.g., "send me sequences containing dogs"). Each application installs on the edge a set of small neural networks called *microclassifiers (MCs)* that perform binary classification on each incoming frame to determine whether an interesting state is occurring. Typically, an interesting state is described in terms of the presence of a certain object. Each MC is trained offline by an application developer. At runtime, frame-level classification results are smoothed to determine the start and end points of

"events" during which the interesting state occurred. Events are re-encoded and streamed to the datacenter.

FilterForward scales to multiple independent applications (e.g., "find dogs and find bicycles") by evaluating many MCs in parallel. Optimizing this multi-tenancy is FilterForward's key contribution. Instead of designing the MCs to operate on raw pixels, FilterForward draws inspiration from modern object detectors and uses a shared base deep neural network (DNN) to extract general features from each frame. All MCs operate on the activations from the base DNN, but they may draw from different layers. This amortizes the expensive task of pixel processing across all of the MCs, allowing FilterForward to execute tens of concurrent MCs using the CPU power available in a small form factor edge node. The base DNN is an expensive per-frame, upfront overhead, but it enables a significant performance improvement once the number of concurrent MCs passes a break-even point.

Our evaluation using two real-world camera feed datasets demonstrates that, for applications meeting FilterForward's requirements (operating with severe bandwidth constraints and requiring high-fidelity data), our architecture uses an order of magnitude less bandwidth than standard compression techniques (Section 4.3). Furthermore, FilterForward is computationally efficient, surpassing the frame rate of existing lightweight filters (Kang et al., 2017) when more than $3-4$ MCs run together and achieving up to $6.1\times$ higher throughput with 50 concurrent MCs (Section 4.4). Finally, MCs are up to $1.3\times$ more accurate than pixel-based DNN filters used in prior work while having up to a $23\times$ lower marginal cost (Section 4.5).

FilterForward is open source at github.com/viscloud/ff.

2 Background and Challenges

This section provides an overview of video analytics before delving into the key challenges introduced by a large-scale camera deployment.

2.1 Video Analytics

Typical video analytics primitives include: *Image classification* categorizes a whole frame based on its most dominant features (e.g., "This is an image of an intersection."). *Object detection* finds interesting objects that may occupy only a small portion of the view and categorizes them (e.g., "This rectangle defines a region containing a car."). *Object tracking* aims to label each object's location across multiple frames (e.g., "This path plots the progress of pedestrian *A* crossing the road."). These and other primitives form the basis of more advanced analyses, such as traffic monitoring, pedestrian action understanding, and hazard detection.

Video analytics workloads entail extensive computation on large amounts of data (e.g., a 1920×1080 pixel stream at 30 frames per second (fps) is $\approx$ 1.5 Gb/s when decompressed). Accomplishing video analytics at scale requires abundant compute, memory, and storage resources, so existing systems often perform this processing in the cloud, using GPUs (Kang et al., 2017; Zhang et al., 2017).

2.2 Edge-to-cloud Challenges

The scenarios that motivate FilterForward include remote "Internet of Things" monitoring and "smart city" deployments of tens or hundreds of thousands of wide-angle, fixed-view cameras. In this section, we describe three key challenges presented by this use case.

2.2.1 Limited Bandwidth

Running video analytics by streaming all video to the cloud conflicts with the bandwidth constraints of some deployments, which preclude uploading all camera data. Each camera's uplink bandwidth is limited, both by the physical constraints of modern wide area network infrastructure and the monetary cost of operating a widespread camera deployment. Specifically, we consider large-scale deployments where each camera receives a bandwidth allocation of a few hundred kilobits per second, or less (Public Parking Authority of Pittsburgh, 2018). For comparison, a *low-quality* H.264-encoded 1080p (1920×1080 pixels) stream is approximately 2 Mb/s, an order of magnitude greater than our available uplink bandwidth. Yet, such low-quality data is often insufficient to perform accurate analyses. Modern 4K (3840×2160 pixels) cameras produce up to 30-40 Mb/s, two orders of magnitude beyond the uplink bandwidth, and this gap will only expand as 8K (7680×4320 pixels) cameras become more common. As a concrete example, we built an off-campus deployment where cameras are mounted next to traffic lights at an intersection. The local Internet service provider charges $400 per month for a single 35 Mb/s uplink, creating a strong economic incentive for us to share that bandwidth between as many cameras as possible (currently, eight 4K cameras share each uplink).

This bandwidth gap, exacerbated by the requirement for high-quality data, necessitates an edge-based decision about which frames to send to the datacenter. FilterForward answers this challenge with semantic filtering that uploads only the frames that are relevant to applications.

2.2.2 Real-world Video Streams

In many surveillance deployments, cameras are mounted high on buildings or light posts and fitted with wide-angle lenses that capture broad views of the surrounding area. Interesting objects (e.g., pedestrians, license plates, parcels, animals, etc.) occupy a small portion of the frame. This poses a challenge for the video analytics primitives discussed

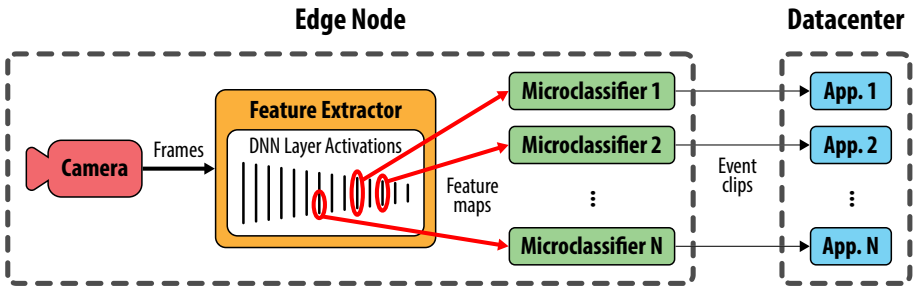


Figure 1. The FilterForward architecture.

in Section 2.1. Image classification maps the entire image to a single category, so an urban viewpoint would always be labeled "street" or "traffic," which is of limited use. Object detection and tracking are designed to pick individual objects out of a frame but often operate on low-resolution images (e.g., as small as 300×300 pixels (Liu et al., 2016)). Aggressive downsampling of a wide-angle image causes details such as license plates and distant people to disappear. To detect these fine-grained details, FilterForward introduces microclassifiers that process high-resolution images on the edge, avoiding quality degradation caused by the decimation required to meet bandwidth constraints.

2.2.3 *Scalable Multi-tenancy*

In real-world deployments, cameras observe scenes containing diverse objects and activities. A single camera may record pedestrians walking down the sidewalk, vehicles stopped at a traffic light, and shoppers entering stores; all while capturing the current weather, the quantity of leaves on the trees, and whether there is snow on the roads. Different applications are simultaneously interested in all of this information, and more. Therefore, any edge-filtering approach must scale to multiple cloud applications focused on disjoint regions of the frame in parallel.

Given edge nodes' limited compute resources, scaling to multiple applications naturally poses a performance challenge. A naïve approach to handling N applications is to run N full DNNs concurrently. However, even relatively lightweight DNNs are costly. In our experience, on a modest Intel® CPU (not GPU), MobileNet (Howard et al., 2017) runs at approximately 15 fps for 512×512 pixel images while consuming more than 1 GB of memory. Even lightweight DNNs have high resource requirements, precluding execution of more than a handful in real time on an edge node. Therefore, to achieve scalability, FilterForward's lightweight MCs simplify per-application processing while the base DNN shares redundant computation between applications.

The rest of this paper describes how FilterForward addresses the above challenges: semantic filtering decimates video streams to meet bandwidth constraints, novel microclassifier architectures detect fine-grained details in wide-angle video, and computation reuse enables scalable multi-tenancy.

3 Designing FilterForward

FilterForward (FF) is a novel video analytics platform that reuses computation to provide highly accurate, multi-tenant video filtering for bandwidth-constrained edge nodes. Purely edge-based approaches constrain applications to the static compute and storage resources of field installations, while datacenter-only analytics necessitate heavily compressing the video for transport. FF offers applications the flexibility of splitting their work between the edge and the cloud, taking advantage of high-fidelity data at the edge to make relevant video sequences available in the cloud.

This section describes the architecture of FF's two main components, the feature extractor and microclassifiers, and explains how they address the three challenges described in Section 2.2: meeting bandwidth constraints, detecting subtle details in real-world video, and supporting many concurrent applications. The system architecture is shown in Figure 1.

3.1 Generating Features

In FF, microclassifiers reuse computation by taking as input *feature maps* produced from the intermediate results (*activations*) of a single reference DNN, which we refer to as the *base DNN*. The component that evaluates the base DNN and produces feature maps is called the *feature extractor*.

As prior work observes (Sharghi et al.; Yeung et al.), activations capture information that humans intuitively desire to extract from images, such as the presence and number of objects in a scene, and outperform handcrafted low-level features (Razavian et al., 2014; Yue-Hei Ng et al., 2015; Babenko & Lempitsky, 2015). The activations of the first layers of a DNN (often simple convolutional filters such as edge detectors) are still visually recognizable. Later activations represent high-level concepts (e.g., "eye," "fur," etc.). Processing feature maps created from these activations has been used successfully for tasks such as object region proposals, segmentation, and tracking (Ren et al., 2015;

Hariharan et al., 2015; Ma et al., 2015; Bertinetto et al., 2016), as well as action classification (Sharma et al., 2015).

For our evaluation, we use the MobileNet (Howard et al., 2017) architecture trained on ImageNet (Russakovsky et al., 2015) as the base DNN. MobileNet offers a balance between accuracy and computational demand that is appropriate for constrained edge nodes. We use the 32-bit (unquantized) version of the network. Picking an appropriate base DNN is, of course, a moving target, and we do not view the selection of a *specific* network as a contribution of this work. Evaluating the robustness of our filtering algorithm to different base DNNs is left for future research.

The feature extractor plays a crucial role in addressing the challenge of detecting small objects in real-world surveillance video (Section 2.2.2). Instead of drastically shrinking incoming frames as is typical in ML-based video analytics, FilterForward examines full-resolution frames. For our evaluation, the full resolution is either 1920×1080 or 2048×850 pixels (Section 4.1), which represent $41.3\times$ and $34.7\times$ increases in total input data, respectively, versus the typical MobileNet input size of 224×224 pixels. By operating on high-resolution frames, small content such as distant pedestrians, make and model–specific automobile details, and faces are captured in greater detail.

However, processing drastically more pixels imposes a significant computation overhead, as the work done by each layer of the base DNN increases. Although feature extraction is the most computationally intensive phase of FilterForward, its results are reused by all of the MCs, amortizing the per-frame, upfront overhead once the number of MCs passes a break-even point. This computation reuse is the key to achieving scalable multi-tenancy, a major challenge for real-world surveillance deployments (Section 2.2.3). Ongoing architectural improvements in off-the-shelf feature extraction networks, as well as advances in hardware accelerators (Jouppi et al., 2017; Apple, 2017; intel-movidius; microsoft-project-brainwave), will continue to reduce FilterForward's computational overhead.

We evaluate the base DNN's computation overhead in Section 4.4. Ultimately, the feature maps generated by the base DNN underpin FF's accuracy and scalability achievements.

3.2 Finding Relevant Frames Using Microclassifiers

Microclassifiers are lightweight binary classification neural networks that take as input feature maps extracted by the base DNN and output the probability that a frame is relevant to a particular application. An edge node can run many MCs on a single camera stream, or fewer MCs on several streams.

An application developer chooses an MC architecture (we present several possibilities in Section 3.3) and trains it offline to detect the application's desired content. To deploy an MC, the developer supplies the network weights and architecture specification along with the name of the base DNN layer (and, optionally, a crop thereof) to use as input.

Each microclassifier can pull feature maps from any layer of the base DNN, enabling FilterForward to support different types of tasks (Section 2.2.3). Section 3.4 discusses the layer selection process.

Furthermore, each MC can optionally crop its feature map, thus focusing on a certain portion of the frame. Selecting a static subregion of the field of view (for each MC) helps specialize FilterForward to wide-angle surveillance video (Section 2.2.2) because some applications are only interested in particular regions. One benefit is that this reduces an MC's computation load proportional to the decrease in its input size. Additionally, constraining an MC's spatial scope increases accuracy (for certain applications) for two reasons: (1) The MC must only consider the relevant region of the frame, and (2) by cropping, important objects become more prominent. A key insight is that by cropping feature maps instead of raw pixels, FilterForward retains its ability to simultaneously support MCs interested in different regions, a key scalability requirement. I.e., cropping an MC's feature map to refine its spatial scope is an optional local optimization that each MC performs independently.

Dropping irrelevant frames is crucial to limiting bandwidth use, FilterForward's primary objective (Section 2.2.1). Ideally, an MC will identify all of the frames that an application needs to process in the cloud while rejecting a large fraction of unimportant frames. The redundancy inherent in video provides a safety margin for false negatives. False positives are particularly harmful because they consume upload bandwidth with irrelevant data.

In the background, edge nodes record the original video stream to disk so that datacenter applications can demand-fetch additional video (e.g., context segments surrounding a matched segment) from the edge nodes' local storage.

As discussed in Section 3.1, sharing computation between MCs via the base DNN is FilterForward's solution to the multi-tenancy demands of real-world surveillance deployments (Section 2.2.3). We show in Sections 4.4 and 4.5 that operating on feature maps instead of raw pixels provides microclassifiers with competitive accuracy while reducing marginal compute cost by an order of magnitude.

3.3 Microclassifier Architectures

As discussed in Section 2.2.2, off-the-shelf classifiers and detectors perform poorly on wide-angle surveillance video because the objects of interest are often small. We propose three custom MC architectures, shown in Figure 2, that solve this challenge in different ways. In addition to the base DNN processing full-resolution frames, two important

microclassifier features that help achieve high accuracy on surveillance video are: (1) MCs operate on activations from whichever base DNN layer, and therefore whichever granularity of features, is most appropriate for their task, while (2) optionally cropping away irrelevant regions of the frame. Both of these capabilities help achieve high accuracy on real-world data, as evaluated in Section 4.5.

3.3.1 *Full-frame Object Detector (Figure 2a)*

Modeled after sliding window–style object detectors such as SSD (Liu et al., 2016) and Faster R-CNN (Ren et al., 2015), the full-frame object detector MC applies a small binary classification DNN at each location in a convolutional layer feature map and then aggregates the detections to make a global prediction. This is achieved by using multiple layers of 1×1 convolutions and then applying a `max` operator over the grid of logits (signifying looking for ≥ 1 objects). This model is specifically designed for pattern matching queries, with an implicit assumption of translational invariance (i.e., the model runs the same template matcher everywhere), and is well-suited to processing entire wide-angle frames.

3.3.2 *Localized Binary Classifier (Figure 2b)*

The localized binary classifier MC is a lightweight convolutional neural network (CNN) that processes spatially cropped feature maps. Consisting of two separable convolutions and a fully-connected layer, this architecture is designed to detect prominent objects within a localized region (i.e., like zooming in to a region of the frame).

3.3.3 *Windowed, Localized Binary Classifier (Figure 2c)*

This architecture extends the localized binary classifier MC to incorporate nearby temporal context, improving per-frame accuracy. The user specifies a temporal window of W frames. Given the convolutional feature maps for a symmetric W-sized window centered at frame F, the windowed, localized binary classifier MC first applies a 1×1 convolution to each frame's feature map, then depthwise-concatenates the resulting activations and applies a CNN to predict whether frame F is interesting. This setup allows the MC to pick up on motion cues in the scene, which helps achieve higher accuracies on tasks where objects are constantly moving. The initial single-frame 1×1 convolution significantly reduces the size of the input feature map, making this larger architecture computationally tractable on edge node hardware. As an optimization, the 1×1 convolutions are only computed once, and their outputs are buffered and reused by subsequent windows, eliminating redundant computation.

3.4 Choosing Microclassifier Inputs

Choosing which base DNN layer to use as input to each microclassifier is critical to their accuracies. The layers of a CNN feature hierarchy offer a tradeoff between spatial localization and semantic information. Too late a layer may not be able to observe small details (because they have been subsumed by global semantic classification). Too early a layer could be computationally expensive due to the large size of early layer activations and the amount of processing still required to transform low-level features into a classification.

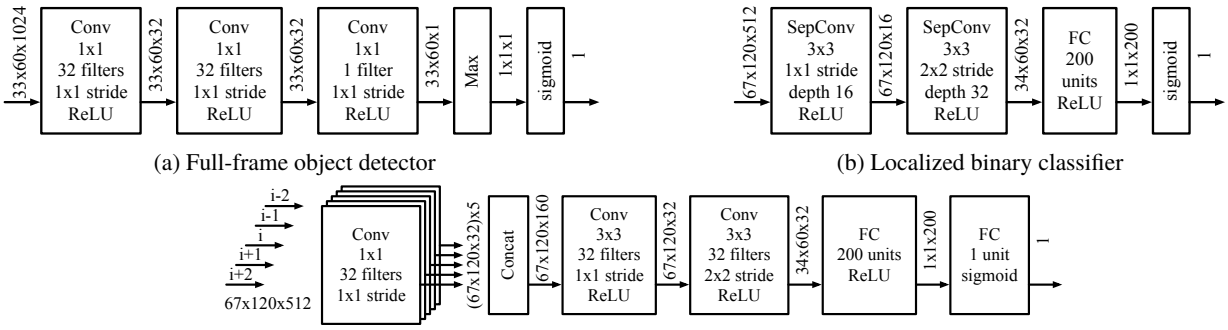
As a baseline, we hand-select a layer, and optionally a crop region, based on two heuristics. First, for the layer, we try to match the typical size of the object class we were detecting. For example, to find pedestrians in a 1920×1080 pixel video where the average height of a human is 40 pixels, we choose the first layer at which a roughly 20:1–50:1 spatial reduction has occurred. In our evaluation, the microclassifiers extract feature maps from the following MobileNet layers: The full-frame object detector uses the penultimate convolutional layer (*conv5_6/sep*) and the localized and windowed, localized binary classifiers use a convolution layer from the middle of the network (*conv4_2/sep*). Their names are specific to the version of MobileNet that we use (cdwat, 2017). Second, we choose the optional crop region based on the region of interest for the application, such as the crosswalks when detecting people (Section 4.1).

In the prototype version of FF, each MC pulls features from a single base DNN layer and we constrain the feature crops to be rectangular. Combining features from multiple layers and experimenting with free-form and discontiguous crop regions, as well as automating the selection of these parameters, are interesting challenges for future work.

3.5 From Per-frame Classifications to Events

A microclassifier outputs binary per-frame classifications (i.e., is this frame relevant or not?), which FilterForward then smooths into event detections. First, each MC's results for N consecutive frames are accumulated into a window. Then, to mask spurious misclassifications, we apply K-Voting to this window, treating the middle frame as a detection if at least K of the N frames in the window are positive detections. For our evaluation, we conservatively set $N = 5$ and $K = 2$, which provides fairly aggressive false negative mitigation at the expense of potential false positives. The resulting smoothed, per-frame labels are fed into a transition detector that considers each contiguous segment of positively-classified frames to be a unique event. Each event is assigned an MC-specific, monotonically increasing, unique ID, which is stored in each frame's metadata. These IDs are used by applications to determine the event boundaries.

A single frame may be classified as part of an interesting event by multiple MCs. For example, if frame F is part of event X for MC A and event Y for MC B, then F's internal metadata will contain the mapping $(A \rightarrow X; B \rightarrow Y)$, indicating that it is part of multiple events. As for the frames themselves, they



(a) Full-frame object detector

(b) Localized binary classifier

(c) Windowed, localized binary classifier

Figure 2. Three microclassifier architectures. The dimensions quoted here correspond to a 1920 × 1080 pixel video (no spatial cropping).

are re-encoded using H.264 at a user-configured bitrate and streamed back to the datacenter. The application developer specifies a bitrate that is sufficiently high for their tasks (the implications of this parameter are discussed in Section 4.3).

4 Evaluation

This section evaluates how FilterForward addresses the three challenges in Section 2: limited bandwidth, real-world video streams, and scalable multi-tenancy. We begin by defining our datasets and accuracy metric, then demonstrate that, for applications looking for rare events, FilterForward significantly reduces bandwidth use. We show that FilterForward achieves a high frame rate on commodity hardware while maintaining high accuracy on two event detection tasks despite having a lower marginal cost than existing techniques.

4.1 Real-world Datasets

We evaluate using two datasets (Figure 3) showing scenes that are representative of the real-world surveillance deployments that FilterForward targets. The first dataset consists of video captured from a traffic camera deployment in Jackson Hole, Wyoming (the *Jackson* dataset). We collected two six-hour videos from two consecutive days, between 10 AM and 4 PM. Then, we annotated the twelve hours of data with labels for when pedestrians appear in the crosswalks (the *Pedestrian* task). This task allows us to demonstrate the spatial selectivity of our microclassifiers in a way that is hopefully relevant to future traffic monitoring applications. E.g., combined with a simple traffic light classifier, a user could craft composite queries to detect jaywalkers.

In addition, we collected a second dataset from a higher-quality camera in our own urban deployment, consisting of two three-hour videos of a city street (the *Roadway* dataset) captured back-to-back during the middle of the day. We annotated the six hours of data with labels for when passing pedestrians are wearing red articles of clothing or carrying red parcels (the *People with red* task). For both datasets, the first video is used for training and the second for testing.

(a) The *Jackson* and *Roadway* datasets, respectively.

(b) Dataset details.

Attribute	*Jackson*	*Roadway*
Resolution	1920 × 1080 pixels	2048 × 850 pixels
Frame rate	15 fps	15 fps
Frames	600,000	324,009
Task	*Pedestrian*	*People with red*
Event frames	95,238	71,296
Unique events	506	326

(c) Rectangular pixel regions that correspond to the tasks' optional spatial crops. Note that in FilterForward, the feature maps are cropped, not the raw pixels.

Task	Upper left corner	Lower right corner
Pedestrian	(0, 539)	(1919, 1079)
People with red	(0, 315)	(2047, 819)

Figure 3. Real-world evaluation videos and tasks.

Table 3c details the pixel regions corresponding to these tasks' optional spatial crops. For the *Pedestrian* task, we select the bottom half of the frame, as the trees and sky are unnecessary. For the *People with red* task, we select the street and sidewalk area (59% of the frame). FilterForward crops the feature maps produced by the base DNN, not the original pixels, so the coordinates in Table 3c are rescaled based on the dimensions of the feature maps. Whether these crops are in effect is described below on a per-experiment basis. The base DNN intermediate layers from which the MCs extract features are described in Section 3.4.

4.2 Defining Event F1 Score

Most classification metrics operate on a per-frame basis. Because FilterForward is event-centric, we adopt a modified recall metric from recent work that is designed for events that span multiple frames (Lee et al., 2018). For an event i, the resulting $EventRecall_i$ metric weighs two success measures: $Existence_i$ rewards detecting at least one frame from the event, and $Overlap_i$ rewards detecting an increasing fraction of the frames from the event. Below, R_i and P_i are the ground truth and predicted event ranges, respectively.

$$Existence_i = \begin{cases} 1 & \text{if detect any frame in event } i \\ 0 & \text{otherwise} \end{cases}$$

$$Overlap_i = \sum_j \frac{|Intersect(R_i, P_i)|}{|R_i|}$$

$$EventRecall_i = \alpha \times Existence_i + \beta \times Overlap_i$$

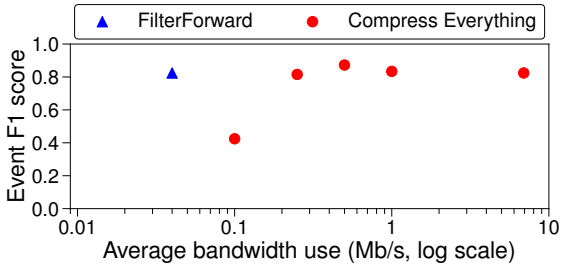
We choose $\alpha = 0.9$ and $\beta = 0.1$ to place greater importance on detecting at least one frame in each event. For real-time event detection in a surveillance setting, we believe that not missing events is more important than capturing all frames in an event. If an application receives at least one frame from an event, then it can demand-fetch additional frames while prioritizing between events.

On the other hand, we retain the standard definition of *precision*: the fraction of predicted frames that are true positives (i.e., $\frac{\text{\# correctly detected}}{\text{total \# detected}}$). For FilterForward, precision determines what fraction of bandwidth is used to send relevant frames. A precision of 1.0 means that all bandwidth is spent sending useful true positive frames. We combine standard precision with our modified definition of event recall to calculate an *event F1 score*—the harmonic mean of precision and recall—which is used throughout this evaluation. An intuitive way to conceptualize event F1 score is as a measure of end-to-end event detection accuracy.

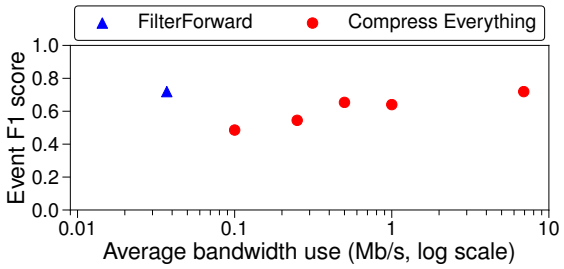
4.3 Saving Wide Area Bandwidth

First, we demonstrate that FilterForward achieves its primary objective, conserving edge-to-cloud bandwidth (Section 2.2.1). Specifically, filtering on the edge with FF uses 6.3 – 13× less bandwidth than heavily compressing and uploading the full stream.

Figure 4 relates average bandwidth use and event F1 score for two MC architectures on the *Roadway* dataset's *People with red* task. In this figure, we evaluate two techniques for uploading video from the edge: "FilterForward" corresponds to running FF on the edge, on the original stream, and then compressing the selected frames for upload; "Compress everything" represents uploading the *entire* stream, compressed to a low bitrate, then running FF in the cloud.



(a) Full-frame object detector MC



(b) Localized binary classifier MC

Figure 4. Bandwidth use on the *Roadway* dataset's *People with red* task for two strategies for offloading data: (1) compressing the video using H.264 and sending all frames; and (2) FilterForward, where only relevant sequences are sent.

Running FilterForward on both the edge stream and the cloud stream allows us to simultaneously analyze its bandwidth and accuracy benefits. We do not evaluate other simple bandwidth-saving techniques beyond full-stream compression because they are comparatively ineffective: (1) Reducing the resolution is infeasible because doing so decimates small details too aggressively; (2) Temporal sampling is nonviable because dropping a few frames does not provide proportional bandwidth savings (video compresses well, so each frame does not add much overhead) and arbitrarily dropping many frames can obscure short events.

Filtering on the edge using the full-frame object detector (Figure 4a) and localized binary classifier (Figure 4b) MCs **reduces bandwidth use by 6.3× and 13×**, respectively, compared to uploading the full stream. Matched frames are re-encoded to 250 Kb/s and 500 Kb/s [1], respectively, and uploaded. These bitrates are chosen as sufficiently good quality for the combination of task and MC. However, it is important to note that FF's bandwidth savings is independent of the selected upload bitrate. Whatever upload bitrate the application developer chooses for their task, FF allows

[1]These values are used as the target bitrates for H.264 compression. Because matched frames are bursty, the average bitrate of the uploaded stream is lower. I.e., there are periods where nothing is uploaded and then there are periods where matched frames are uploaded at these qualities. Furthermore, in practice, regardless of the bitrate used by the compression algorithm, the upload will be throttled to the maximum bandwidth of the network connection.

them to utilize that bandwidth more efficiently by dropping irrelevant frames. I.e., instead of distributing the available bandwidth uniformly across all frames, FF allows the user to concentrate their limited bandwidth resources on the frames that matter most, thus delivering those frames to the datacenter at the highest possible quality.

Of course, the number of frames that FF drops depends on the rarity of the events that the MCs are searching for. The lower the aggregate detection frequency, the more bandwidth FF will save. In Figure 4, the localized binary classifier MC saves more bandwidth (i.e., drops more frames) than the full-frame object detector MC because it experiences more false negatives (i.e., it misses some events).

In terms of accuracy, compared to heavily compressing the full stream to a bandwidth similar to that used by FF, the full-frame object detector and localized binary classifier MCs **increase the event F1 score by 1.5× and 1.9×**, respectively. This demonstrates the value of processing high-fidelity data: When using similar amounts of bandwidth, FF achieves much higher accuracy than uploading the full stream because filtering the original data on the edge gives FF access to fine-grained details that compression destroys. In effect, FF combines the accuracy of sending the original video with the bandwidth savings of heavy compression.

4.4 End-to-end Performance Scalability

FilterForward embraces performance scalability as a first-class design objective (Section 2.2.3). To demonstrate microclassifiers' low marginal cost, we compare to two alternative filtering techniques: (1) naïvely running multiple instances of a full DNN (MobileNet), and (2) training specialized pixel-level classifiers. The specialized classifiers, referred to as *discrete classifiers (DCs)* because they process raw pixels, are similar to techniques used in NoScope (Kang et al., 2017) (discussed further in Section 5.2.1). A DC is faster than a general-purpose image classification DNN like MobileNet but more expensive than an MC. Section 4.5 offers a more detailed cost and accuracy comparison with DCs. FilterForward, the full DNNs, and the DCs all operate on full resolution frames, which for these experiments are 1920×1080 pixels.

We constructed several DCs with between 100 million and 2.5 billion multiply-adds, varying the number of convolutional layers ($2-4$), the number of kernels ($16-64$), the stride length ($1-3$), the number of pooling layers ($0-2$), and the type of convolutions (standard or separable). We fixed the kernel size to 3. We report results for a representative example from the Pareto frontier of accuracy and cost.

All performance experiments are conducted on a desktop computer with a quad-core Intel® Core™ i7-6700K CPU and 32 GB of RAM, using only the CPU (not the integrated or discrete GPUs). In our experience, this CPU is representative of an edge node mounted on a light post, but we expect future deployments to also contain GPUs or DNN hardware accelerators. We execute the base DNN using a version of the Caffe deep learning framework (caffe) that has been optimized for Intel CPUs (Intel) and uses the Intel Math Kernel Library for Deep Neural Networks (intel-mkl-dnn). We execute the MCs and DCs using TensorFlow (Abadi et al., 2016). We set the neural network batch size independently for the full DNNs, DCs, and MCs based on a short parameter sweep. To reduce CPU contention, we use end-to-end flow control to guarantee that, for FilterForward, the base DNN and MCs are executed in phases (not pipelined) so that Caffe and TensorFlow do not compete for cores. Evaluation videos are H.264-compressed, reside on disk, and are always written back to disk (to simulate archiving the full stream for lazy upload) so all performance experiments implicitly contain disk reads/writes and H.264 decoding. At no time are disk accesses or decoding the bottleneck. Because our testbed software stack is not heavily optimized, the magnitude of our performance measurements matters less than the trends in how the different architectures scale.

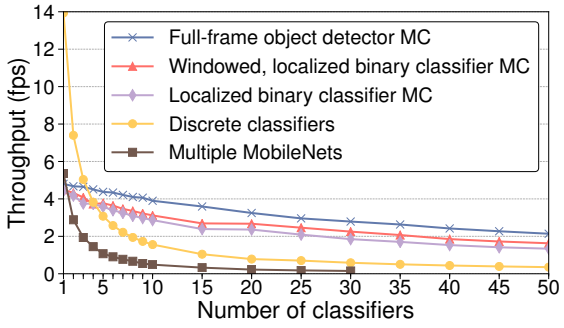


Figure 5. Throughput (in frames per second) of the three MC architectures compared to full DNNs and discrete classifiers. FilterForward amortizes the cost of the base DNN when running 4 or more concurrent MCs.

Figure 5 compares the filtering throughput of FilterForward's three MC architectures to that of multiple full MobileNet DNNs and NoScope-style discrete classifiers. With only a single classifier, FilterForward processes frames at $0.32-0.34\times$ the speed of the DCs and $0.83-0.90\times$ the speed of multiple MobileNets. By 20 classifiers, this has risen to $3.0-4.1\times$ faster than the DCs. By 50 classifiers, FilterForward has up to **6.1× higher throughput**. Intuitively, with only a single filtering task, using MobileNet directly yields higher throughput than FilterForward because of feature extraction and data movement overheads in the latter. By two classifiers, these overheads do not matter. On the other hand, the DCs have higher throughput when the number of classifiers is low because they do not pay the overhead of running an expensive full DNN. However, since each DC must compute the full translation from pixels to a

decision, they perform redundant work. By sharing compute and amortizing the cost of its base DNN, FilterForward is **faster with more than 3 - 4 classifiers**. Running multiple MobileNets, while straightforward, is never optimal from a throughput perspective and runs out of memory beyond 30 classifiers. Ultimately, for use cases with more than a handful of filtering tasks, FilterForward offers much higher throughput than existing techniques.

To further understand FilterForward's throughput scalability, for each frame we measure the time taken by the base DNN and MCs. Figure 6 shows this breakdown for our three proposed microclassifier architectures. With few queries, the base DNN's execution time dominates, as expected. The total execution time grows only modestly as we add dozens of concurrent MCs. Depending on the microclassifier, the base DNN's CPU time is equivalent to that of 15 - 40 MCs.

4.5 Microclassifier Cost and Accuracy

Finally, we demonstrate that because they operate on feature maps, FilterForward's microclassifiers have substantially lower marginal compute cost, yet higher accuracy on real-world datasets, than discrete classifiers (Section 2.2.2). We use the same discrete classifier architecture as in Section 4.4, and trained the MCs and DCs on 0.5 epochs of data, using spatial crops (Table 3b) for the applicable MCs and the *Roadway* dataset's DC (the *Jackson* dataset's DC's accuracy did not benefit from a spatial crop).

Multiply-adds are a good proxy for the compute cost of a DNN model (Howard et al., 2017). Given a feature map of size $H \times W$ and depth M, the number of multiply-adds in a fully-connected layer with N hidden units is: $N \times H \times W \times M$. For the same feature maps, the number of multiply-adds in a convolutional layer with F filters of size $K \times K$ and a stride of S is: $\frac{H}{S} \times \frac{W}{S} \times M \times K^2 \times F$. The cost of a convolutional layer can be reduced using separable or "factored" convolutions (kernels are split into depthwise followed by pointwise convolutions) with some accuracy penalty. The number of multiply-adds in a separable convolutional layer with the same parameters is: $\frac{H}{S} \times \frac{W}{S} \times M \times (K^2 + F)$. Recall that FilterForward ingests full resolution video, so in our experiments, the number of multiply-adds is much greater than for typical input sizes, such as 224×224 pixels.

Figure 7 compares two microclassifiers' accuracy (event F1 score) and marginal compute cost (number of multiply-adds) to those of the discrete classifiers for both of our datasets. Compared to the DCs, FilterForward's MCs are up to 1.3× as accurate while being 23× cheaper on the *Jackson* dataset and 1.1× as accurate and 11× cheaper on the *Roadway* dataset. This order of magnitude savings in marginal compute cost results from the MCs operating on feature maps instead of pixels, meaning that they have less work to do (translating features to a classification is simpler than doing so for pixels). Of course, the tradeoff is that the MCs must pay the upfront overhead of the base DNN.

We believe that, in Figure 7, the slight accuracy improvements compared to the DCs are a byproduct of using a more complex network for the pixel processing. The DCs must walk a fine line between accuracy and cost: To remain lightweight compared to full multi-class DNNs like MobileNet, they sacrifice complexity. By amortizing the pixel processing across all its MCs, FF allows users to run a more powerful feature extraction network, and therefore extract higher-quality features that are a better basis for additional analysis, than would be possible using the DCs.

5 Related Work

This section outlines related work that applies to the three challenges raised in Section 2.2 (limited bandwidth, real-world video streams, and scalable multi-tenancy)

5.1 Conventional Machine Learning Approaches

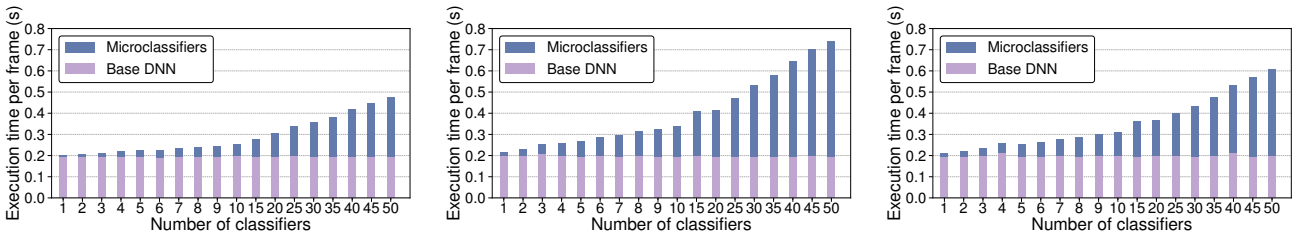
Conventional ML techniques for reusing computation improve scalability, but their rigidity sacrifices accuracy.

Transfer learning accelerates multi-application training and inference by leveraging the observation that DNNs trained for image classification and object detection identify general features that transfer well to specialized tasks (Donahue et al., 2014; Yosinski et al., 2014). During inference, transfer learning shares computation by running one base DNN to completion and extracting its last layer's activations as a feature vector, which is then used by multiple specialized classifiers (one per application) (Pakha et al., 2018). Recent work allows application-specific DNNs to share multiple layers with the base DNN (Jiang et al., 2018a), similar to how our microclassifiers can pull from any layer of FilterForward's base DNN. However, conventional transfer learning suffers from poor accuracy for small objects because it retains the original DNN architecture for the retrained layer(s). Even though these approaches are computationally efficient, they are not tailored to real-world video streams.

Multi-task learning (Caruana, 1998) offers an efficient way to share computation across models, but all models must be retrained when new tasks are added. This retraining overhead makes multi-task learning unsuited to real-world deployments, where tasks are frequently added and removed.

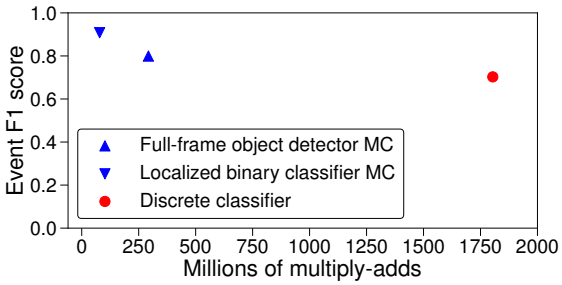
5.2 Filtering-based Approaches

Filtering video by dropping irrelevant frames reduces computation and transmission load (Kang et al., 2017; Pakha et al., 2018; Wang et al., 2018). One method of filtering is to use a cascade of progressively more accurate and expensive detectors, stopping execution at the cheapest model that produces

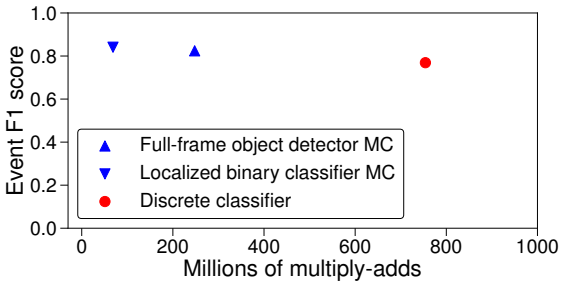


(a) Full-frame binary classifier (b) Localized binary classifier (c) Windowed, localized binary classifier

Figure 6. Execution time (in seconds) breakdown of FilterForward's main components for the three microclassifier architectures. FilterForward pays the upfront cost of evaluating the base DNN, but then reaps the resulting benefit of each additional MC being cheap.



(a) *Jackson* dataset, *Pedestrian* task.



(b) *Roadway* dataset, *People with red* task.

Figure 7. Number of multiply-adds versus event F1 score for microclassifiers and discrete classifiers. MCs have a much lower marginal cost than DCs, yet achieve higher accuracy.

a high confidence prediction. This is a common technique for optimizing the "fast path" where most frames can be discarded near the beginning of the cascade. Early work in this field includes (Viola & Jones, 2001), which introduces a detector cascade based on traditional computer vision features and includes an attention mechanism to prune the feature space and improve throughput. FilterForward builds on this idea but specializes to the task of detecting small objects in surveillance video. Similar to the aforementioned attention mechanism, FF includes an optional optimization where a microclassifier can spatially crop its feature map to focus on a particular region (to improve accuracy) and reduce model complexity (to save computation).

5.2.1 *Saving Compute During Bulk Analytics*

Recent work has applied filter cascades to reduce computation load during bulk video analytics. NoScope (Kang et al., 2017) drops frames whose pixel-level differences from a reference image or previous frame do not meet a threshold, before feeding them into the cascade. NoScope first evaluates cheap, task-specific, pixel-level CNNs (e.g., a custom "Shetland pony" binary classifier), which we refer to as *discrete classifiers*, and only applies an expensive CNN (e.g., YOLO9000 (Redmon & Farhadi, 2016)) when the confidence of the cheap CNN is below a threshold. Discrete classifiers are similar to our MCs, except that they operate on raw pixels. In FilterForward, the base DNN amortizes pixel processing across all MCs, reducing the marginal cost of each classifier without sacrificing accuracy. We compare the throughput and accuracy of our MCs to NoScope's discrete classifiers in Sections 4.4 and 4.5.

Previous systems were often evaluated on highly-curated datasets, where video processing was orchestrated to be easier. For example, NoScope (Kang et al., 2017) was evaluated on video that has been cropped to a narrow region of interest (objects typically occupy the majority of the frame). Enlarging objects in this way makes analysis both easier (because objects are more prominent) and cheaper (because the DNN input resolution can be reduced). However, modifying the data in this way diverges from our goal of processing wide-angle surveillance video. FilterForward supports a similar cropping technique, but this is not crucial to its design.

Focus (Hsieh et al., 2018) divides processing between ingest time and query time, using cheap CNNs and clustering to build an approximate index up front that dramatically accelerates offline queries. Focus's ingest CNN is conceptually similar to FilterForward's base DNN—they both generate semantic information about each frame that is used for future processing—but our use of feature maps instead of top classes is more general. The notion of storing per-frame metadata in an index is applicable to FilterForward and an interesting direction for future work.

Both NoScope and Focus assume that it is possible to stream all video to a resource-rich datacenter. This is not

fundamental to their algorithms, but pushing components of either system to the edge would introduce additional compute constraints. A basic premise of FilterForward is that uploading all video is infeasible, so our design builds off computation sharing that enables an edge node to support many concurrent applications.

5.2.2 *Saving Bandwidth on Constrained Edge Nodes*

Similar to FilterForward, others have approached the challenges of running ML workloads on edge-generated video in real time. Both (Pakha et al., 2018) and (Wang et al., 2018) push computation to the edge to determine which frames are "uninteresting" to heavyweight analytics in the cloud.

(Pakha et al., 2018) uses sampling and superposition coding to send frames only when relevant objects appear and then using the lowest possible quality. While the work displays impressive bandwidth savings, the iterative communication between the edge and the cloud limits its throughput.

(Wang et al., 2018) examines the heavily bandwidth-constrained use case of offloading video in real time from a swarm of autonomous drones using the 4G LTE cellular network. Similar to FilterForward, this system uses lightweight DNNs (e.g., MobileNet) running on the edge (here, on the drones) combined with lightweight classifiers (they use support-vector machines (SVMs)) to give an early indication of whether a frame is interesting. These SVMs are similar in principle to our microclassifiers, but always operate on activations extracted from the base DNN's final pooling layer and are much shallower than MCs, meaning that they have a lower capacity to learn and inferior accuracy.

Additionally, both of these systems are not optimized for multi-tenant environments. FilterForward is designed with query scalability as a first-class concern, and can run dozens of concurrent microclassifiers.

Both (Pakha et al., 2018) and (Wang et al., 2018) focus on streams where the camera is moving, whereas FilterForward considers stationary surveillance cameras. Operating on streams with less global motion gives FilterForward an advantage because it is easier to train classifiers for these streams, and the larger proportion of unchanging pixels makes such streams more compressible.

5.3 Resource Scheduling for Video Pipelines

Resource management is crucial for practical video analytics because applications often impose the conflicting goals of maximizing their overall benefit and meeting performance constraints. For instance, VideoStorm (Zhang et al., 2017) adjusts query quality to maximize a combined utility, using efficient scheduling that leverages offline quality and resource profiles. LAVEA (Yi et al., 2017) places tasks across edge nodes and clients (e.g., mobile phones) to minimize the latency of video analytics. DeepDecision (Ran et al., 2018) expresses resource scheduling in video processing as a combinatorial optimization problem. Chameleon (Jiang et al., 2018b) dynamically adjusts a video processing pipeline's hyperparameters as the content in the scene changes, using temporal and spatial (i.e., across nearby cameras) correlations to prune the optimization search space.

Much of this scheduling work is complementary to FilterForward, which shares a similar motivation of balancing accuracy and throughput, but focuses on edge nodes with constrained network bandwidth. Unlike prior scheduling work that adjusts only general knobs such as video bitrate, resolution, and choice of DNN model, FilterForward's computation sharing directly improves the computational efficiency of multiple filters running on the same edge node.

6 Conclusion

Scaling real-time, wide-area video analytics poses a challenge for bandwidth-limited, compute-constrained camera deployments. This paper presents FilterForward, a new filtering architecture for the edge that uses lightweight, per-application microclassifiers to identify relevant video segments to upload. We show that FF reduces bandwidth use by an order of magnitude without sacrificing accuracy while scaling to as much as 6.1× higher throughput than existing approaches. However, even though this paper describes FilterForward in terms of saving bandwidth on the edge, we believe that our scalable early-discard algorithm is a viable method of eliding unnecessary computation in ML-based cloud analytics as well. We believe that FilterForward's computation sharing and hybrid edge-to-cloud design transcend video processing and provide useful building blocks for ML applications in constrained environments.

FilterForward is open source at github.com/viscloud/ff.

Acknowledgments

We appreciate the insights of the SysML '19 program committee and our colleagues at Carnegie Mellon University and Intel Labs. This work was supported by Intel via the Intel Science and Technology Center for Visual Cloud Systems (ISTC-VCS).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: A system for large-scale machine learning. In *Proc. 12th USENIX OSDI*, Savannah, GA, November 2016.

Apple. The future is here: iPhone X. https://www.apple.com/newsroom/2017/09/the-future-is-here-iphone-x/, 2017.

Babenko, A. and Lempitsky, V. Aggregating local deep features for image retrieval. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, pp. 850–865. Springer, 2016.

caffe. Caffe. http://caffe.berkeleyvision.org/, 2017.

Canel, C., Kim, T., Zhou, G., Li, C., Lim, H., Andersen, D. G., Kaminsky, M., and Dulloor, S. R. Scaling video analytics on constrained edge nodes. In *Proceedings of the 2nd SysML Conference (SysML '19)*, Palo Alto, CA, 2019.

Caruana, R. Multitask learning. In *Learning to learn*, pp. 95–133. Springer, 1998.

cdwat. MobileNet-Caffe. https://github.com/cdwat/MobileNet-Caffe, 2017.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pp. 647–655, 2014.

FCC. 2016 BROADBAND PROGRESS REPORT. 2016.

Google Wireless Internet. Google's Excellent Plan To Bring Wireless Internet To Developing Countries. https://www.forbes.com/sites/timworstall/2013/05/25/googles-excellent-plan-to-bring-wireless-internet-to-developing-countries/, 2013.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 447–456, 2015.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

Hsieh, K., Ananthanarayanan, G., Bodik, P., Venkataraman, S., Bahl, P., Philipose, M., Gibbons, P. B., and Mutlu, O. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 269–286, Carlsbad, CA, 2018. USENIX Association. ISBN 978-1-931971-47-8. URL https://www.usenix.org/conference/osdi18/presentation/hsieh.

IHS. Top Video Surveillance Trends for 2017. https://cdn.ihs.com/www/pdf/TEC-Video-Surveillance-Trends.pdf, 2017.

Intel. Intel distribution of caffe. https://github.com/intel/caffe.

intel-mkl-dnn. Intel Math Kernel Library for Deep Neural Networks. https://01.org/mkl-dnn, 2018.

intel-movidius. Intel Movidius Neural Compute Stick. https://developer.movidius.com/, 2018.

ITU/UNESCO Broadband Commission for Sustainable Development. The State of Broadband: Broadband catalyzing sustainable development. 2017.

Jiang, A., Wong, D. L.-K., Canel, C., Misra, I., Kaminsky, M., Kozuch, M., Pillai, P., Andersen, D. G., and Ganger, G. R. Mainstream: Dynamic stem-sharing for multi-tenant video processing. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018a.

Jiang, J., Ananthanarayanan, G., Bodik, P., Sen, S., and Stoica, I. Chameleon: Scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pp. 253–266, New York, NY, USA, 2018b. ACM. ISBN 978-1-4503-5567-4. doi: 10.1145/3230543.3230574. URL http://doi.acm.org/10.1145/3230543.3230574.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., et al. In-datacenter performance analysis of a tensor processing unit. *CoRR*, abs/1704.04760, 2017. URL http://arxiv.org/abs/1704.04760.

Kang, D., Emmons, J., Abuzaid, F., Bailis, P., and Zaharia, M. Noscope: Optimizing deep cnn-based queries over video streams at scale. *PVLDB*, 10(11):1586–1597, 2017.

Lee, T. J., Gottschlich, J., Tatbul, N., Metcalf, E., and Zdonik, S. Precision and recall for range-based anomaly detection. In *Proc. SysML Conference*, Stanford, CA, February 2018.

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

Ma, C., Huang, J.-B., Yang, X., and Yang, M.-H. Hierarchical convolutional features for visual tracking. In

Proceedings of the IEEE International Conference on Computer Vision, pp. 3074–3082, 2015.

microsoft-project-brainwave. Microsoft unveils Project Brainwave for real-time AI. https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/, 2018.

Pakha, C., Chowdhery, A., and Jiang, J. Reinventing video streaming for distributed vision analytics. In *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*, Boston, MA, 2018. USENIX Association. URL https://www.usenix.org/conference/hotcloud18/presentation/pakha.

Public Parking Authority of Pittsburgh. Rfp for unified security camera system. http://apps.pittsburghpa.gov/redtail/images/3780_RFP_FOR_UNIFIED_SECURITY_CAMERA_SYSTEM_9.20.18.pdf, 2018.

Ran, X., Chen, H., Zhu, X., Liu, Z., and Chen, J. DeepDecision: A mobile deep learning framework for edge video analytics. In *Proc. INFOCOM*, 2018.

Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, 2014.

Redmon, J. and Farhadi, A. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. URL http://arxiv.org/abs/1612.08242.

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

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Sharghi, A., Laurel, J. S., and Gong, B. Query-focused video summarization: Dataset, evaluation, and A memory network based approach. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, CVPR 2017.

Sharma, S., Kiros, R., and Salakhutdinov, R. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015.

Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, 2001.

Wang, J., Feng, Z., Chen, Z., George, S., Bala, M., Pillai, P., Yang, S.-W., and Satyanarayanan, M. Bandwidth-efficient live video analytics for drones via edge computing. In *Proceedings of the Third ACM/IEEE Symposium on Edge Computing (SEC 2018)*, Bellevue, WA, 2018.

Yeung, S., Russakovsky, O., Mori, G., and Fei-Fei, L. End-to-end learning of action detection from frame glimpses in videos. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, CVPR 2016.

Yi, S., Hao, Z., Zhang, Q., Zhang, Q., Shi, W., and Li, Q. LAVEA: Latency-aware video analytics on edge computing platform. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017.

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

Yue-Hei Ng, J., Yang, F., and Davis, L. S. Exploiting local features from deep networks for image retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

Zhang, H., Ananthanarayanan, G., Bodik, P., Philipose, M., Bahl, P., and Freedman, M. J. Live video analytics at scale with approximation and delay-tolerance. In *Proc. 14th USENIX NSDI*, Boston, MA, March 2017.