Networked Cameras Are the New Big Data Clusters

Junchen Jiang[1], Yuhao Zhou[1], Ganesh Ananthanarayanan[2], Yuanchao Shu[2], Andrew A. Chien[1]
[1]University of Chicago, [2]Microsoft Research

ABSTRACT

The increasing complexity of deep learning and massive deployment of cameras at the edge have drastically increased the resource demand of edge data analytics. Compared to traditional Internet web applications, such resource demand (in computing, storage and networking) is not limited by millions of human users, but rather the continuous activities of billions of sensors. This paper presents the abstraction of *camera cluster* as an attempt to address this challenge in the context of video analytics. We envision a novel analytics stack that orchestrates the computing resource of massive networked cameras to enable efficient edge video analytics.

CCS CONCEPTS

• **Networks → Application layer protocols**; *Cloud computing*.

KEYWORDS:

Camera Cluster; Video Analytics; Edge

ACM Reference Format:

Junchen Jiang, Yuhao Zhou, Ganesh Ananthanarayanan, Yuanchao Shu, Andrew A. Chien. 2019. Networked Cameras Are the New Big Data Clusters. In *2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo'19), October21, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3349614.3356026

1 INTRODUCTION

With dropping camera prices and increasing accuracy of deep neural networks (DNNs), we see an explosive growth of video-analytics applications [7, 16, 23, 53] and deployments of *large* camera networks (with thousands of cameras) [4, 5, 15]. Meanwhile, the proliferation of *on-camera* compute resource [1, 2, 8, 10, 11] has spurred the prospect of massive video analytics at edge. To deliver these promises, however, we must address the fundamental systems challenge: *How to utilize the on-camera resource in a large camera fleet to run video-analytics applications at scale?*

Existing solutions are insufficient. When processing multiple camera streams [53], current systems (Figure 1(a)) analyze (and optimize the performance of) each video stream *individually*, using only the local resource of the source camera (and cloud/edge servers only for overflow) [25, 34, 43, 51]. As a result, the resource demand grows proportionally with more cameras and more applications.

What's missing, we believe, is not software/hardware resource, but an *abstraction* for a group of cameras to act as a whole, like a compute cluster. This paper envisions transforming a group of networked cameras to a **camera cluster**—a compute cluster (Figure 1 (b)) with an *analytics stack* to provide *efficient resource sharing*, *uniform resource accessing*, and a *unifying abstraction* to perform video analytics applications on many camera feeds. The abstraction of camera cluster would bring principles and benefits, both in performance and programmability.

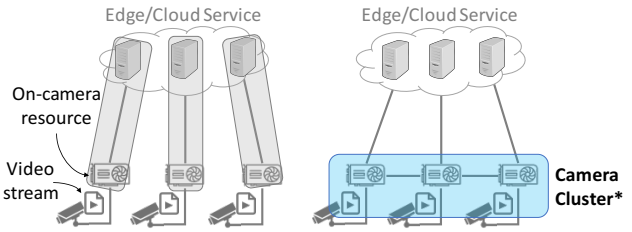
Performance benefits: The camera cluster abstraction offers a systematic way to fully utilize the ever-growing on-camera resource in large camera networks. It opens up new opportunities for optimization. For example, since different cameras' workloads are naturally heterogeneous [39], a camera cluster can spread the workload of one camera to other cameras so that more applications can run with the same on-camera compute resource. Moreover, since applications often use model cascades (*e.g.,* [42, 50]), not all models need to be called for each frame. Thus, instead of loading/unloading DNN models, which can be unwieldy, a camera cluster can leave the models loaded on specific cameras and route the "data" to these locations.

Programmability benefits: Recent years have seen both innovations in low-level accelerators (*e.g.,* [22, 52]), high-level video analytics pipelines [25, 40–42, 50, 51, 55, 60, 61], and applications (*e.g.,* history [36] vs. live [60]), each suited for different use cases. These trends are untenable. Today, applications are compiled directly on low-level libraries [6, 9, 12, 13] that run on a single device (camera/server), so adopting useful techniques across applications will involve significant repetitions and reinventions. In contrast, with a common programming abstraction, camera clusters could simplify application development by allowing applications and system-level optimization to evolve independently.

Recent work has shown early promise of camera clusters, such as sharing information across nearby cameras [39, 61]

HotEdgeVideo'19, October 21, 2019, Los Cabos, Mexico

ACM ISBN 978-1-4503-6928-2/19/10…$15.00
https://doi.org/10.1145/3349614.3356026



(a) Current solution (b) "Camera cluster" abstraction

Figure 1: *Current video analytics solutions (left) analyze each video stream independently, whereas a* camera cluster *(right) treats all cameras' resource and their video streams as a compute cluster. *Note that camera cluster is architecturally amenable to include the edge/cloud resource too.*

and merging queries [31, 37, 40, 44]. These techniques, however, have focused on individual cameras or the "edge-cloud" hierarchy, rather than a group of networked cameras. Moreover, they are often tied to specific computer-vision tasks and pipelines. For example, many pipelines save costs by downsizing videos or sampling important frames [41, 46, 58, 59], but doing so precludes other query pipelines that need high-fidelity frames [26, 51]. Finally, without a *holistic* view of an analytics stack, they often address isolated issues and cause new problems elsewhere. For instance, many video pipelines invoke very expensive DNNs only for a small fraction of frames (*e.g.,* [42, 50]), but doing this also creates spiky compute workloads, which makes resource sharing difficult.

The work most closely related to camera clusters is femtocloud [32, 33], which shares with us the high-level idea of organizing edge devices (*e.g.,* smartphones) connected to a wireless access point into a cluster. In particular, it runs generic cloud jobs over edge devices by maximally utilizing the devices' resource. However, it focuses on jobs with predictable resource demands, while video-analytics workloads can be highly dynamic and content-dependent [41, 42]. Moreover, video analytics have unique opportunities, *e.g.,* videos can be re-scaled to trade resource demand for accuracy.

Our goal in this paper, therefore, is to call up the community to tackle the challenges of developing an *analytics stack* for a camera cluster. We first make a case for the "camera cluster" abstraction (§2,3) and discuss the main challenges of building a camera-cluster analytics stack (§4).[1] We believe that work on a camera cluster is not only relevant to video analytics and its problems today, but can significantly impact the future edge computing software stack. If successful, such a common system substrate would facilitate a rich ecosystem of video analytics applications.

[1]Given the wide range of issues involved in designing a whole camera cluster, we focus on the perspectives of its necessity and architecture in this paper, and leave further questions, including privacy/security, energy consumption, and facult tolerance, to future work.

2 A CASE FOR CAMERA CLUSTER

2.1 Recent trends

Video analytics applications (*e.g.,* traffic monitoring and indoor security) are traditionally run on a single or small set of video streams. Moreover, most applications rely on the edge/cloud servers for heavy-lifting analytics (*e.g.,* object detection/classification, re-identification).

However, two recent trends may make this formulation obsolete. First, networked cameras are being deployed *en masse* [4, 5, 18, 21], creating an explosive growth of video streams that require automatic analysis [7, 19]. Second, the cameras are empowered with more *on-board* compute resource [2, 10, 11] and can run complex deep learning models locally (which is often required for privacy compliance [48]).

Thus, the next-generation video analytics applications need to extract real-time insights from *many* cameras [16, 23] by harnessing the increasing on-camera resource. For example, cross-camera identification tracking searches queried identities in video feeds and then tracks them as they move between cameras over time in a large area [53]. These applications can make real impact when deployed at scale; *e.g.,* large cities have thousands of traffic cameras installed to detect "close-calls" between cars, bikers, and pedestrians, which helps preemptively deploy safety measures [17].

2.2 The "camera cluster" abstraction

As shown in Figure 1(a), traditional video applications analyze each camera stream independently using only the resource of the source camera and edge/cloud servers for overflow (and then combine the results if needed).

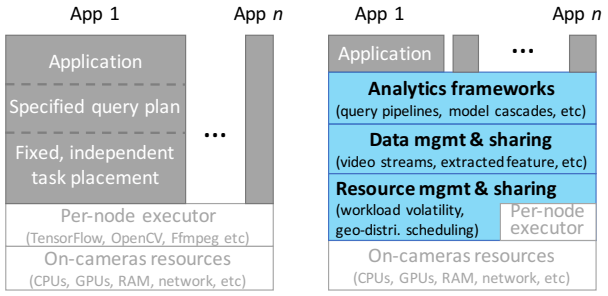
We envision an alternative approach — transforming a group of networked cameras to a compute cluster, called *camera cluster*, which manages the resource sharing and video streams in a camera network with an unifying abstraction (Figure 1(b)). The camera cluster abstraction enables:

1. *Efficient resource sharing* between different applications over an entire camera network.
2. *Flexible data access* to video streams and intermediate data.
3. *Optimized scheduling* for heterogeneous applications.
4. *A common programming interface* for increasingly diverse video analytics pipelines.

Moreover, when videos must be analyzed "locally" by edge devices [48], being able to fully utilize the on-camera resource becomes more necessary. Of course, a camera cluster can also be combined with the edge/cloud resource, though it is not our focus. In this paper, we start with networked cameras operated by the same organization. We leave the question of what cameras should be clustered to future work.

2.3 What's new about camera cluster?

An *analytics stack* over distributed cameras (Figure 2) is the key to unleashing the potential of camera clusters. It helps to put the camera-cluster stack into the perspective of cloud big



(a) Current applications are built independently & holistically

(b) Video analytics stack for camera clusters

Figure 2: *Today's video analytics system vs. the proposed analytics stack (in green) for camera clusters.*

data clusters and edge clusters. Several factors differentiate the camera cluster environment from both settings.

Live videos ≠ Files: Most cloud frameworks (Spark, Mesos, Yarn, Hadoop) are designed for batch jobs whose inputs are stored in distributed storage. Video streams are different in that the videos are neither partitioned (for data locality) nor replicated (for fault tolerance and scalability). Moreover, videos are amenable to compression (frame sampling, image downsizing, etc), while general input data are not.

Live video analytics ≠ Streaming analytics: The cloud frameworks for streaming analytics are unsuitable for live video analytics. In mini-batch streaming (like Spark Streaming), the supported minimum batch sizes are at least a few seconds in practice, whereas many video analytics applications expect an output to be almost at a frame-level (*i.e.,* 33ms per frame for a 30fps video). Other streaming frameworks (like Storm) handle continuous data streams, but those data rates and their compute demands are far lesser than video analytics workloads (complex DNNs on HD video streams).

Vision inference ≠ Big data tasks: A camera cluster must optimize for "model locality" (§3.1). However, most cloud frameworks are optimized for data locality and in-memory caches, all of which assume loading data, rather than loading model and inference, is likely the performance bottleneck.

Camera cluster ≠ Edge/cloud hierarchy: A camera cluster is different from a hierarchy between the edge and cloud. For instance, cross-camera load balancing (§3.1) is not considered in the edge/cloud hierarchy, because the application will resort to the edge/cloud whenever it needs overflow capacity. The task placement in edge/cloud hierarchy needs to handle less resource dynamism, because the heavyweight models can just be loaded in the cloud while the light models are on the edge or camera. Finally, in an edge/cloud hierarchy, the network connectivity between cameras is not fully used, since all traffic flows between individual cameras and edge/cloud servers.

3 BENEFITS OF CAMERA CLUSTERS

Next we present several motivating scenarios where video analytics applications could be improved by a camera cluster. These scenarios are inspired by real applications and existing video pipelines.

3.1 Benefit #1: Saving computing resource

Pipeline merging: Running multiple applications over a shared infrastructure creates many opportunities to save resource consumption by reusing intermediate data. Video analytics is particularly amenable to such data sharing. For instance, two applications analyzing the same video feeds only need to perform video decoding operations or vision feature-extraction once [37, 40]. In addition, history video queries (*e.g.,* "find the peak traffic hours over the last week") can be made very efficient by caching continuous live query results (*e.g.,* congestion monitoring) [36].

Model locality: Different applications sometimes use (or customize) the same set of publicly available vision models. This suggests one can share models, in addition to data, between applications that perform the same tasks. For instance, instead of loading/unloading DNN models frequently, it is better to leave the models loaded on specific cameras and route the "data" to these locations. This can be effective for three reasons. First, loading a large DNN model can be slower than actual inference [38]; *e.g.,* loading a ResNet50 model to GPU takes about 10 seconds, which is 100× longer than using it to classify images (50 images per second) [14]. Second, many video analytics pipelines are cascades (*e.g.,* [25, 42, 50]), so not all models need to be called for each frame. Finally, same pre-loaded DNNs can batch process frames from multiple cameras together to further save computing cost.

3.2 Benefit #2: Resource pooling

Load balancing: With a camera cluster, one can process each video stream using the original camera's resource *and* many other cameras'. This will be particularly useful, when different camera feeds naturally exhibit heterogeneity in their workloads (*e.g.,* cameras tend not to have interesting incidents simultaneously). Even if analyzing each video stream has time-varying workloads, we can still support many applications and many video streams by offloading workload spikes of one camera to many other cameras, without needing to provision more resources per-camera.

Ability to run more complex models: By pooling together the ever-growing resource on cameras (and the edge/cloud), a camera cluster can run more complex and more accurate models than today's solutions. One possible implementation is to split a video stream to frame groups and process them in parallel by multiple cameras. This is particularly useful for processing history videos, which is not limited by the fixed frame arrival rate.

3.3 Benefit #3: Improving analytics quality

Sharing insights across cameras: As a camera cluster offers direct access to the video streams of other cameras, one can improve video analytics quality of one video stream by leveraging insights from other video streams, even though they are *not* directly queried [41]. For instance, to find the best model for analyzing camera A, one can look at camera B that has similar video characteristics (*e.g.,* similar object class distributions camera angles, or lighting conditions), and if we know which model is accurate for B, that model would probably produce high accuracy for A.

Online model update: Vision models are more accurate when the test video streams are similar to the training data (similar lighting or camera angle). Thus, some query pipelines update the vision models with locally captured images via transfer learning (*e.g.,* [42, 50]). Today, they use cloud resource to update the models, but this can bring privacy complications [48] and extra delays. A promising alternative is to leverage all on-camera resource to update model, without resorting to cloud.

3.4 Benefit #4: Hiding low-level intricacies

The camera cluster abstraction also hides the intricacies of underlying hardware and simplifies video analytics application development, allowing applications and implementation to evolve independently. This is particularly needed, given the growing diversity in hardware and software systems (*e.g.,* [25, 36, 40–42, 50, 55, 60, 61]), each suited for different use cases. Ideally, application developers need a common interface to access video streams and compute/network resource, without worrying about resource contention, bandwidth sharing and fault tolerance, etc. Unfortunately, many applications today are built independently and holistically with no such common interface, leading to substantial reinventions and repetitions when applying useful techniques (*e.g.,* [34, 37, 41, 58]) across different applications.

4 CAMERA CLUSTER ARCHITECTURE

While some of the aforementioned benefits have been studied in specific contexts (*e.g.,* [37, 39–41, 44, 61]), there has been few systematic effort to design a camera cluster to explore *all* benefits simultaneously. Here, we focus on three functionalities essential to a camera cluster.

4.1 Resource sharing

A camera cluster will be shared by multiple applications, each of which has many users. These applications can vary in time scales (*e.g.,* realtime vs historical), accuracy requirements, and resource demands (*e.g.,* DNN-based object detection needs orders of magnitude more computation than object tracking). Meanwhile, there are more video analytics users; *e.g.,* a crossroad traffic camera might be used by three uses: monitoring congestion, spotting traffic-light violation, and identifying a specific car. Thus, a camera cluster needs efficiently resource sharing (GPU cycles, memory, network) between users of same application and between applications.

Today, applications are built independently, and users directly configure where the analytics should run and reserve resource potentially for indefinitely period of time. The only option for sharing resources between applications is at the level of a single camera or server [28, 54]. This approach is inefficient to achieve the performance benefits outlined in §3. In contrast, a more efficient resource sharing mechanism for camera clusters should address four challenges.

Heterogeneity-aware scheduling: A better scheduling algorithm should explore the inherent heterogeneity between the resource/accuracy tradeoffs of different applications. This has led to substantial resource savings in edge/cloud servers (*e.g.,* [37]) but it remains unclear how to achieve similar savings over a large cluster of distributed cameras that are loosely connected and have heterogeneous resources.

Sharing network resource: Network connectivities between cameras (*e.g.,* WiFi or LTE) can be a limiting factor in a camera cluster, and the network loads between cameras can also be dynamic as we stream videos across cameras. So we need to reduce the amount of communication (*e.g.,* by exploring "video locality" [61]). The network resource must also be shared properly among applications. Current solutions [26, 46, 58] are insufficient as they assume the network is used by only one video stream.

Workload volatility: Resource demand of video analytics can be highly volatile. For instance, some video analytics pipelines invoke expensive models only for a small number of frames where interesting incidents occur (*e.g.,* [25, 42, 50]). This creates sudden spikes in resource demand that are hard to predict. In contrast, cloud scheduling algorithms (for minimizing job completion time) rely on the tasks having deterministic resource demands [29, 30].

Use of virtualization: Finally, we need to decouple *which* camera feeds are being analyzed from *where* analytics happens. Containers (*e.g.,* Docker) can be a promising approach to this goal. Edge products like Azure Data Box Edge [3] already support containerized deployments. Aspects discussed earlier like pipeline merging and load balancing will have to be carefully designed to ensure an acceptable overhead of shipping containers. Techniques that "pre-warm" the containers will be crucial towards dynamic spin-up of containers in a camera cluster.

4.2 Data access and sharing

Efficient data sharing is also needed for the performance optimization described in §3. In particular, we need a common data-accessing API to share three types of data.

Video data: The API should support both *streamed* videos (read with low latency and in high resolution, but no need to be saved for a long time) and *archived* videos (read in batch

and typically in compressed formats). It should also express key video properties (video color space, frame rate, quantization parameters, etc) that can heavily influence video analytics performance [60]. It should also allow access to *any* video streams that meet a given criteria (*e.g.,* video streams that have non-overlapping views [39], or video streams where a given identity is detected [53]). Unfortunately, today's interface — accessing videos by directly specifying the video source — is inefficient to meet these goals. Moreover, camera network operators today often store archived data in cloud [36], which forces users to rewrite programs to switch between live video analytics and history video analytics.

Intermediate data: For different applications to share their intermediate data, the data-accessing API should express not only the original video stream, but the vision models that generate the data as well. Suppose one application wants to count the number of buses on a street, and the other application wants to monitor the speed of buses on the street. Naturally, they can share the detected "bus" instances. But for such sharing to be transparent (*i.e.,* same output with or without sharing), we must ensure the "bus" instances are found by the same "bus-detection" model. One option is to have the storage system remember how to recompute each intermediate data, in much the same way that a MapReduce system knows how to re-run a map task if it loses its output.

Model reference: Finally, to enable "model locality" (load a DNN model to GPU memory once and route data to it), we must allow different applications to access the same vision models and routines. One idea is to create a unique fingerprint for each model, so a preloaded model can be reused by another model if it provides the same fingerprint. However, this means any bit-level change can result in different fingerprints, which can be too stringent, since even training the same DNN architecture on the same dataset can yield slightly different models. An alternative is to imbue vision-related semantics in the API, such as in [45].

4.3 Programming frameworks and interface

As mentioned in §3.4, camera clusters provide an abstraction that separate applications from hardware intricacies. To realize this benefit, we need a programming framework to support and optimize an increasing number of diverse analytics pipelines, from monolithic models [41, 60] to various model cascades [40, 42, 50], and other complex structures (*e.g.,* [25, 46]). We envision that a camera cluster exposes a declarative interface with symbolic APIs to allow programmers to define an analytics pipelines as static data flow (similar to the popular deep learning frameworks [20, 24]). This can greatly facilitate performance optimization.

Moreover, many pipelines also need to adapt their configurations or vision models (§3.3) to cope with the dynamic video content (*e.g.,* [25, 41, 46, 50, 58, 60]). These pipelines can benefit from a *real-time profiling* API, where the intermediate data can be used to profile the performance of different configurations (or counterfactually evaluate possible changes) and then apply the changes to the pipelines [41].

Finally, many video analytics applications use complex pipeline of multiple models to analyze a video stream. Like other machine learning applications, these models can have errors (both false positives and false negatives), so many applications often involve human inspection or developers may need to debug which model causes some error (*e.g.,* object being missing or necessary features not extracted).

5 RELATED WORK

Applications over camera networks: Analyzing real-time video feeds from a large camera network is a popular topic in vision and edge computing communities. A wide range of applications, multi-camera tracking, multi-camera activity monitoring, and object reidentification, are being studied and optimized with novel vision algorithms (see [53] for a comprehensive review). These promises are gaining momentum in real world too with well-funded initiatives taking root (*e.g.,* [17]). This work sheds light on the need for and the challenges of a system stack to support these applications.

Video analytics pipelines: There is an increasing number of video-analytics pipelines that optimize for high accuracy and low resource consumption [25, 34, 36, 37, 40–42, 46, 50, 51, 58, 60, 61]. These pipelines, however, are independently built on low-level executors running on a single camera/server, and mostly for processing a single camera stream. So the applications have to handle all the intricacies of hardware, communication, fault tolerance, etc. In contrast, a camera-cluster analytics stack will hide these intricacies from applications while enabling efficient resource sharing and data sharing between applications for better performance.

Edge data analytics systems: There have been several attempts at providing system abstractions of edge compute resource (see [47] for a review). Some are general-purpose solutions (*e.g.,* [32, 33] seek to build a generic cloud service at edge) or focus on edge servers (*e.g.,* [54]), so they are not suitable to explore the opportunities unique to camera-cluster and video analytics. Some more recent work has examined separate issues in edge video analytics in isolation (*e.g.,* video data storage [49, 56], model management [27, 38], and sensor network [21]), but they lack an integrative effort towards a holistic analytics stack, as big data framework in cloud.

ACKNOWLEDGEMENT

This work was supported in part by CNS-1901466, CERES Research Center, and a Google Faculty Research Award.

REFERENCES

[1] 'ai camera' combines machine vision, deep learning. https://www.enterprisetech.com/2018/10/19/ai-camera-combines-machine-vision-deep-learning/. Accessed: 2019-3-3.

[2] Amazon deeplens cameras. https://aws.amazon.com/deeplens/. Accessed: 2019-2-13.

[3] Azure data box edge. https://docs.microsoft.com/en-us/azure/databox-online/data-box-edge-overview. Accessed: 2019-3-4.

[4] British transport police: Cctv. http://www.btp.police.uk/advice_and_information/safety_on_and_near_the_railway/cctv.aspx. Accessed: 2018-10-27.

[5] Can 30,000 cameras help solve chicago's crime problem? https://www.nytimes.com/2018/05/26/us/chicago-police-surveillance.html. Accessed: 2018-10-27.

[6] Ffmpeg. https://www.ffmpeg.org/. Accessed: 2019-2-13.

[7] Humans can't watch all the surveillance cameras out there, so computers are. https://slate.com/technology/2019/06/video-surveillance-analytics-software-artificial-intelligence-dangerous.html. Accessed: 2019-3-4.

[8] Introducing intel vision accelerator design products. https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/vision-accelerator-design-product-brief.pdf. Accessed: 2019-3-3.

[9] Mxnet: A scalable deep learning framework. https://mxnet.apache.org/. Accessed: 2019-2-13.

[10] New nvidia deepstream sdk 3.0 removes boundaries of video analytics. https://news.developer.nvidia.com/new-nvidia-deepstream-sdk-3-0-removes-boundaries-of-video-analytics/. Accessed: 2019-2-20.

[11] Nvidia jetson systems. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems-dev-kits-modules/. Accessed: 2019-2-13.

[12] Opencv library. https://opencv.org/. Accessed: 2019-2-13.

[13] Tensorflow. https://www.tensorflow.org/. Accessed: 2019-2-13.

[14] Tensorflow image classification benchmarks. https://www.tensorflow.org/guide/performance/benchmarks. Accessed: 2019-3-4.

[15] Video analytics market size worth $9.4 billion by 2025 | cagr: 22.8%. https://www.grandviewresearch.com/press-release/global-video-analytics-market. Accessed: 2019-2-13.

[16] Video surveillance: How technology and the cloud is disrupting the market. https://cdn.ihs.com/www/pdf/IHS-Markit-Technology-Video-surveillance.pdf. Accessed: 2019-2-13.

[17] The vision zero initiative. http://www.visionzeroinitiative.com/. Accessed: 2019-3-1.

[18] Market for small IP camera installations expected to surge. http://www.securityinfowatch.com/article/10731727/, 2012.

[19] Data generated by new surveillance cameras to increase exponentially in the coming years. http://www.securityinfowatch.com/news/12160483/, 2016.

[20] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[21] K. Abas, C. Porto, and K. Obraczka. Wireless smart camera networks for the surveillance of public spaces. *Computer*, 47(5):37–44, 2014.

[22] J. Albericio, A. Delmás, P. Judd, S. Sharify, G. O'Leary, R. Genov, and A. Moshovos. Bit-pragmatic deep neural network computing. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 382–394. ACM, 2017.

[23] G. Ananthanarayanan, V. Bahl, P. BodÃŋk, K. Chintalapudi, M. Philipose, L. R. Sivalingam, and S. Sinha. Real-time video analytics âĂŞ the killer app for edge computing. *IEEE Computer*, October 2017.

[24] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[25] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168. ACM, 2015.

[26] S. P. Chinchali, E. Cidon, E. Pergament, T. Chu, and S. Katti. Neural networks meet physical networks: Distributed inference between edge devices and the cloud. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pages 50–56. ACM, 2018.

[27] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan. Cachier: Edge-caching for recognition applications. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 276–286. IEEE, 2017.

[28] B. Fang, X. Zeng, and M. Zhang. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 115–127. ACM, 2018.

[29] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Nsdi*, volume 11, pages 24–24, 2011.

[30] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella. Multi-resource packing for cluster schedulers. *ACM SIGCOMM Computer Communication Review*, 44(4):455–466, 2015.

[31] P. Guo and W. Hu. Potluck: Cross-application approximate deduplication for computation-intensive mobile applications. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 271–284. ACM, 2018.

[32] K. Habak, M. Ammar, K. A. Harras, and E. Zegura. Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *2015 IEEE 8th international conference on cloud computing*, pages 9–16. IEEE, 2015.

[33] K. Habak, E. W. Zegura, M. Ammar, and K. A. Harras. Workload management for dynamic mobile device clusters in edge femtoclouds. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, page 6. ACM, 2017.

[34] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 123–136. ACM, 2016.

[35] J. Hauswald, M. A. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang, et al. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In *ACM SIGPLAN Notices*, volume 50, pages 223–238. ACM, 2015.

[36] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 269–286, 2018.

[37] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose. Videoedge: Processing camera streams using hierarchical clusters. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 115–131. IEEE, 2018.

[38] L. N. Huynh, Y. Lee, and R. K. Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 82–95. ACM, 2017.

[39] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. E. Gonzalez. Scaling Video Analytics Systems to Large Camera Deployments. In *ACM HotMobile*, 2019.

[40] A. H. Jiang, D. L.-K. Wong, C. Canel, L. Tang, I. Misra, M. Kaminsky, M. A. Kozuch, P. Pillai, D. G. Andersen, and G. R. Ganger. Mainstream: Dynamic stem-sharing for multi-tenant video processing. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 29–42, 2018.

[41] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 253–266. ACM, 2018.

[42] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597, 2017.

[43] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *Acm Sigplan Notices*, 52(4):615–629, 2017.

[44] R. LiKamWa and L. Zhong. Starfish: Efficient concurrency support for computer vision applications. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 213–226. ACM, 2015.

[45] Y. Lu, A. Chowdhery, and S. Kandula. Optasia: A relational platform for efficient large-scale video analytics. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 57–70. ACM, 2016.

[46] C. Pakha, A. Chowdhery, and J. Jiang. Reinventing video streaming for distributed vision analytics. In *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*, 2018.

[47] J. Pan and J. McElhannon. Future edge cloud and edge computing for internet of things applications. *IEEE Internet of Things Journal*, 5(1):439–449, 2018.

[48] Q. M. Rajpoot and C. D. Jensen. Video surveillance: Privacy issues and legal compliance. In *Promoting Social Change and Democracy through Information Technology*, pages 69–92. IGI global, 2015.

[49] A. Ravindran and A. George. An edge datastore architecture for latency-critical distributed machine vision applications. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[50] H. Shen, S. Han, M. Philipose, and A. Krishnamurthy. Fast video classification via adaptive cascading of deep models. *arXiv preprint*, 2017.

[51] S. Teerapittayanon, B. McDanel, and H. Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 328–339. IEEE, 2017.

[52] F. Tu, W. Wu, S. Yin, L. Liu, and S. Wei. Rana: towards efficient neural acceleration with refresh-optimized embedded dram. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, pages 340–352. IEEE Press, 2018.

[53] X. Wang. Intelligent multi-camera video surveillance: A review. *Pattern recognition letters*, 34(1):3–19, 2013.

[54] D. Willis, A. Dasgupta, and S. Banerjee. Paradrop: a multi-tenant platform to dynamically install third party services on wireless gateways. In *Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture*, pages 43–48. ACM, 2014.

[55] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 129–144. ACM, 2018.

[56] T. Xu, L. M. Botelho, and F. X. Lin. Vstore: A data store for analytics on large videos. In *Proceedings of the Fourtheenth European Conference on Computer Systems*, 2019.

[57] M. Zaharia, B. Hindman, A. Konwinski, A. Ghodsi, A. D. Joesph, R. Katz, S. Shenker, and I. Stoica. The datacenter needs an operating system. In *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*, pages 17–17. USENIX Association, 2011.

[58] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee. Awstream: Adaptive wide-area streaming analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 236–252. ACM, 2018.

[59] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2718–2726, 2016.

[60] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In *NSDI*, volume 9, page 1, 2017.

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