Real-time Video Analytics – the *killer app* for edge computing

Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodik, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, Sudipta Sinha

According to a 2015 report by IHS on the installed base for video surveillance equipment, there is a camera installed for every 29 people on the planet, with mature markets like the US having a camera for every eight of its citizens. The report expects the number of cameras to grow by 20% *year-over-year* for the next five years. Video analytics from these cameras are used for traffic control, retail store monitoring, surveillance and security, as well as consumer applications including digital assistants for *real-time* decisions.

Our position is that a *geo-distributed architecture of public clouds, private clusters and edges extending all the way down to compute at the cameras* is the only viable approach that can meet the strict requirements of real-time, large-scale video analytics for the following reasons.

1) **Latency:** Applications require processing the video at *very low latency* because the output of the analytics is used to interact with humans (such as in augmented reality scenarios) or to actuate some other system (such as intersection traffic lights).
2) **Bandwidth:** High-definition video requires large bandwidth (5Mbps or even 25Mbps for 4K video) and streaming large number of video feeds directly to the cloud might be infeasible. When cameras are connected wirelessly, such as inside a car, the available uplink bandwidth is very limited.
3) **Provisioning**: Using compute at the cameras allows for correspondingly lower provisioning (or usage) in the cloud. Also, uninteresting parts of the video can be filtered out, for example, using motion-detection techniques, thus dramatically reducing the bandwidth that needs to be provisioned.

Besides low latency and efficient bandwidth usage, another major consideration for continuous video analytics is the *high compute cost of video processing*. Because of the high data volumes, compute demands, and latency requirements, we believe that cameras represent the most challenging of "things" in Internet-of-Things, and large-scale video analytics may well represent the *killer application* for edge computing. Tapping the potential of the recent dramatic increase in the capabilities of computer vision algorithms is an exciting systems challenge.

We are building a real-time video analytics system with low (resource) cost to produce outputs with high accuracy. While built for generic usage, our initial focus is traffic safety, motivated by the statistic that traffic accidents are among the top ten causes of all fatalities. Our video analytics system runs 24X7 processing *live* camera feeds from traffic intersections at the City of Bellevue (WA). The system produces directional traffic volumes and is on track to identify dangerous conflict patterns towards minimizing traffic deaths. We plan to expand our solution to cities across the US and worldwide.

In this article we summarize our work on real-time video analytics over the past several years. Section 1 covers the key application domains for video analytics. Section 2 describes the geo-distributed infrastructure for video analytics. We describe our video analytics software stack in Section 3. Section 4 provides a high-level overview of the key techniques in the video analytics stack with appropriate references to the details of these techniques. Section 5 finishes with details of our ongoing production deployment.

1 Video Analytics Applications

Video analytics is central to a wide range of future and existing applications ranging from surveillance and self-driving cars, to personal digital assistants and even drone cameras. We explain these applications and highlight their

performance requirements. The state-of-the-art is to deploy expensive custom solutions with hard-coded set of video analytics that typically do not take advantage of the geo-distributed edge infrastructure.

1.1 Vision Zero for Traffic

Traffic related incidents are staggeringly among the top ten causes of deaths worldwide! Recognizing this issue, many cities have adopted the Vision Zero initiative [1] whose stated goal is to eliminate traffic-related deaths. Large cities have hundreds or thousands of traffic cameras installed and our extensive conversations reveal that they are very interested in analyzing video from these cameras for traffic safety and planning.

Detecting "close-calls" between cars, bikers, and pedestrians helps preemptively deploy safety measures. To help install crosswalks, cities would like to detect areas where jaywalkers cross streets. To improve congestion, they would like to detect double-parked cars but exclude first-responders. Counting volumes of cars, pedestrians or bikes feeds into the traffic light controller to appropriately control the durations.

1.2 Self-driving and Smart Cars

A key enabler of self-driving cars is using video streams from its multiple cameras to make low-latency driving decisions. This involves the ability to fuse video feeds from multiple cameras and quickly analyze them. In fact, smart cars, even today, have cameras in them that help trigger safety decisions on brakes and lights.

In fact, output from the analytics of traffic camera feeds can also trigger decisions in cars. As we demonstrated in Hannover Masse in 2016, a camera above the traffic light detects poorly visible pedestrians (e.g., hidden between parked cars) and warns approaching self-driving cars using DSRC communication.

1.3 Personal Digital Assistants

Digital assistants with the ability to offer personalized and interactive experiences is an important upcoming technology with potential to dramatically alter our daily activities. These assistants are either part of mobile devices or as standalone hardware in the home (e.g., Amazon Alexa or Jibo). They are equipped with cameras and the key to their personal interaction is recognizing faces, identifying gestures and other activities. Naturally, these analytics have to continuously execute with low latency, either locally on the device or with assistance from the remote cloud.

1.4 Surveillance and Security

Enterprise and government surveillance consists of one of the largest deployment of cameras with millions in the US, UK and China. Police departments are also providing body-worn cameras to law enforcement officers. Analyzing the live video from the body-worn cameras to identify suspicious license plates or people helps alert the officer to call for backup. Automatically detecting sudden motion also helps reacting to any untoward incidents.

Cameras are fast moving into home surveillance too. Companies like Nest or Ring sell home surveillance cameras that identify motion and people towards detecting events like package deliveries, break-ins, and stray animals. Drone cameras area also predicted to significantly aid in surveillance activities.

1.5 Augmented reality

In augmented reality (AR), a user wears goggles that project additional information into the user's field of view, either by projecting a hologram (like in Microsoft's HoloLens) or by recording the view around the user and rendering directly on top of it. To reach its full potential, AR systems need to detect and track objects around the user, recognize people and faces, and perform additional video analytics. However, such analytics cannot be performed directly inside the headset because they require powerful hardware and could overheat. Instead, such analytics could be offloaded to a nearby edge computer (such as a powerful PC in the living room) or to the cloud.

The applications described above have compute requirements ranging from fraction of a core to several CPU cores with some requiring a GPU to run complex DNNs. Requirement on video bandwidths varies from hundreds of kbps

to tens of Mbps. Many applications require latency below a second, while some even as low as tens of milliseconds. A geo-distributed architecture is best suited to cater to the low response times and high-bandwidth requirements.

2 Infrastructure Design and Characteristics

To support the wide range of video analytics scenarios outlined above, the infrastructure for video analytics must embrace the following characteristics intrinsically in its design:

a) **Geo-distribution:** Analytics across cameras, edges, private clusters, public clouds; not just a central location.
b) **Multi-tenancy:** many queries per camera, and queries across many cameras.
c) **Hardware Heterogeneity:** Mix of processing capacities (CPUs, GPUs, FPGAs, ASICs) and networks.

Organizations – cities, police departments, and retail stores – manage large networks of cameras. While in some cases all cameras are in a single building, in other cases they can be distributed across the city, perhaps multiple cities, or even the entire country. These organizations have the desire to execute continuous video analytics on all of their *live* camera streams (e.g. counting cars on *all* intersections).

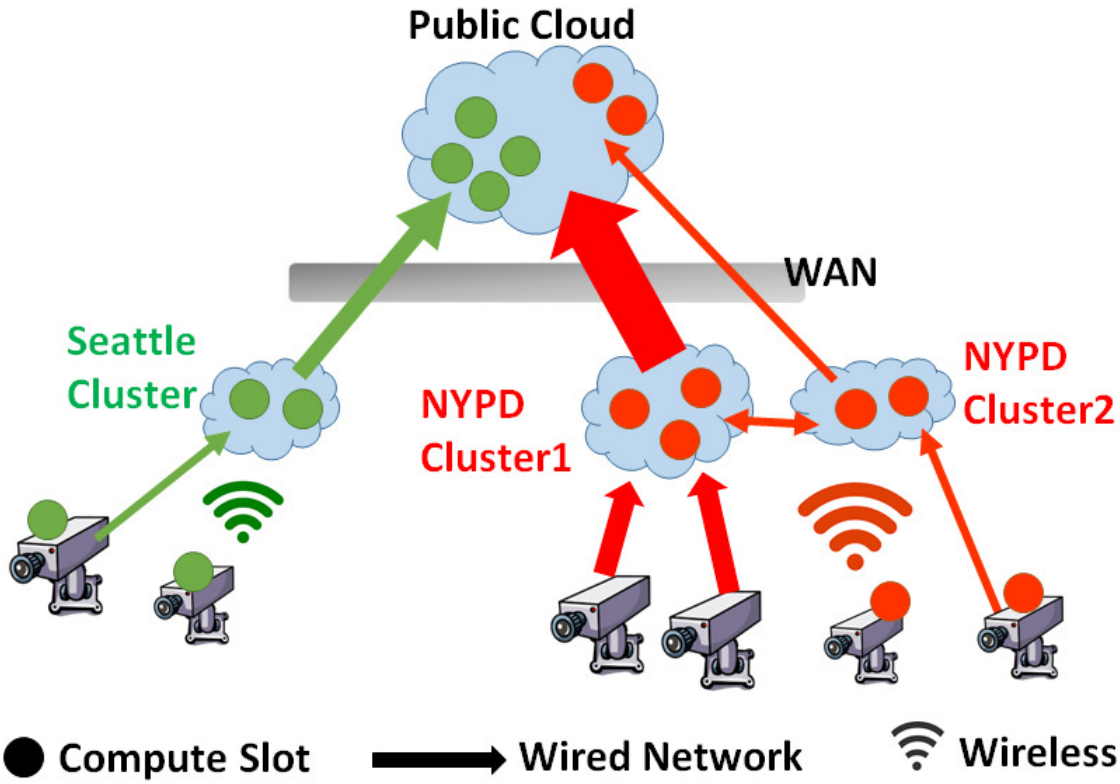

Figure 1: Geo-distributed video analytics infrastructure

Figure 1 illustrates the *hierarchical* geo-distributed infrastructure for video analytics. The infrastructure includes edge and private clusters with heterogeneous hardware to decode video, detect objects, and perform other video analytics tasks. Edge devices may include cameras themselves or processing units placed close to the cameras (e.g., at traffic intersections, in cars, or in buses). Each organization might also manage private clusters (of varying sizes), while also using the processing in public cloud clusters such as Microsoft Azure. The edge, private, and public cloud *clusters* also differ in the type of hardware available. For example, some clusters (including the cameras) might include GPUs. Other types of hardware, such as FPGAs and custom ASICs [5] will be in the public clouds.

The network connecting the cameras, edges, private clusters and the public cloud is a critical resource. The bandwidth required to support a single camera ranges from hundreds of Kb/s for low-resolution video all the way to above 10Mb/s for multi-megapixel cameras. Cameras are connected using a wired or wireless link (Wi-Fi or LTE) to the edge device or the private cluster. In most cases, the uplink bandwidths between the private clusters and cloud are not sufficient to stream a large number of high-resolution videos to the cloud.

While the infrastructure in Figure 1 is general enough to capture all video analytics deployments, we believe that the deployments in practice will vary *across the spectrum* ranging from all the analytics in the cloud (for small deployments of cameras), all analytics in the private clusters (insufficient bandwidths to the cloud or privacy reasons), hybrid of edge/private clusters and cloud (the common case, perhaps after denaturing of videos for privacy [13]), to even all analytics on the edge (or camera). Video analytics sytems should be designed to function across this spectrum of deployments. This is analogical to big data analytics whose original focus was on executing on a single cluster but now is beginning to execute across geo-distributed clusters [2].

3 Video Analytics Software Stack

Figure 2 describes our video analytics stack under development.

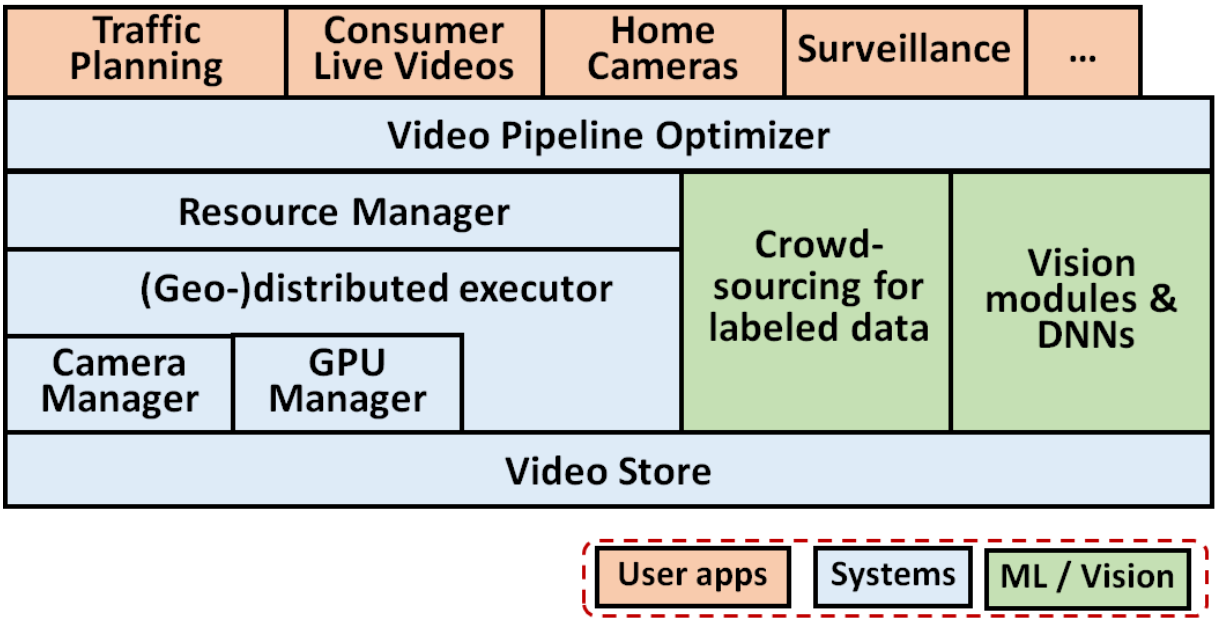

Figure 2: Video Analytics Software Stack

The *video pipeline optimizer* converts high-level video *queries* to video-processing pipelines composed of many *vision modules*; for example, video decoder, followed by object detector and then an object tracker. Each module implements pre-defined interfaces to receive and process events (or data) and send its results downstream. The modules also specify their configuration knobs that we dynamically modify as part of resource management. The modules can internally implement one of our application-level optimizations described in Section 4.4.

The *pipeline optimizer* also estimates the query's *resource-accuracy profile*. Specifically, for the different configurations of knobs and implementations of each vision module in the pipeline, it computes the total resource cost and accuracy of the query. To obtain labeled data required to compute accuracy, the optimizer invokes crowdsourcing. The pipeline and its generated profile is then submitted to the *resource manager.*

The (logically) centralized *global resource manager* (Section 4.1) is responsible for all currently executing query pipelines and their access to all resources -- CPU and GPU compute, network and even pan-tilt-zoom parameters of

steerable cameras. Periodically, the global manager determines the best configuration of each query, the placement of its components across the available clusters, and resource allocation for each vision module. When appropriate, it also merges common modules *across* pipelines processing the same video stream. The pipelines execute through the *geo-distributed executor*, which runs across all the clusters.

We allocate CPU, network, and memory resources to modules using standard OS mechanisms, but we use custom mechanisms for GPUs and steerable cameras. We make execution of DNNs on GPUs dramatically more efficient by running a *DNN execution service* on each machine that handles the all DNN requests on the GPU (Section 4.2). Also, each camera has a *camera manager* that adjusts resolution, frame rate and quality of the video. For steerable cameras, the camera manager *virtualizes* across multiple query pipelines accessing the camera (Section 4.3).

Video storage is a key component of our stack and we intend to build upon ideas from prior work like GigaSight [12] for faster cataloging and retrieval of videos.

4 Core Technologies

In this section, we describe the important technical features that enable us to meet our goals of real-time, high accuracy, low cost execution of video analytics.

4.1 Resource-Accuracy Profiles for Scheduling

Video analytics can have *very high resource demands*. Tracking objects in video is a core primitive for many scenarios, but the best tracker in the VOT Challenge 2015 processes only 1 frame/second on an 8-core machine while most cameras stream data at 30 frames/second. Some of the most accurate DNNs for object recognition, another core primitive, require 30GFlops to process a single frame [3]. To reduce high processing costs and data rates, we propose *approximation* as a first-order abstraction in our system – we carefully adjust *configuration* of each query and the amount of allocated resources to control the *accuracy* of query output.

A key property of video analytics is their *resource-accuracy* relationship. Vision algorithms typically contain various parameters, e.g., frame resolution, frame sampling, and internal algorithmic parameters. In general, processing each frame in the video stream (without sampling) and at the highest resolution results in the highest accuracy but also increases resource demand. Other examples include which specific DNN model to execute (Section 4.2). The combination of these parameter *configurations* determines the *accuracy* of the produced output as well as its *resource demand*. We abstract all the application-specific parameters into the resource-accuracy profile.

Figure 3 shows an example resource-accuracy profile of the canonical object tracker processing a video whose incoming rate is 30 frames/second. We profile 300 configurations of parameters (e.g., frame sampling, resolution, and implementation choices) and compare it to the ground truth of the tracks obtained using crowdsourcing. Note the vast spread in accuracy as well as the CPU demand and data-rates (representing network demands). For each of the configurations, if the allocated resource is less than the demand, the analytics cannot keep up with the incoming rate of the video stream.

Generating the resource-accuracy profile is challenging. Unlike SQL queries, there are no well-known analytical models to capture the resource-accuracy relations as they often depend on the *specific camera view*. Figure 3 shows an example of how different implementations for object detection are better suited for different cameras. Likewise, reducing video resolution may not reduce the accuracy of license plate reader if the camera is zoomed in enough, but will impact accuracy otherwise. Therefore, we need to generate the profile using a *labeled dataset of ground truths* for each video query pipeline for each camera. However, exhaustively running through all the configurations can be prohibitive. For instance, generating the profile for a license plate query consumed 20 CPU *days*.

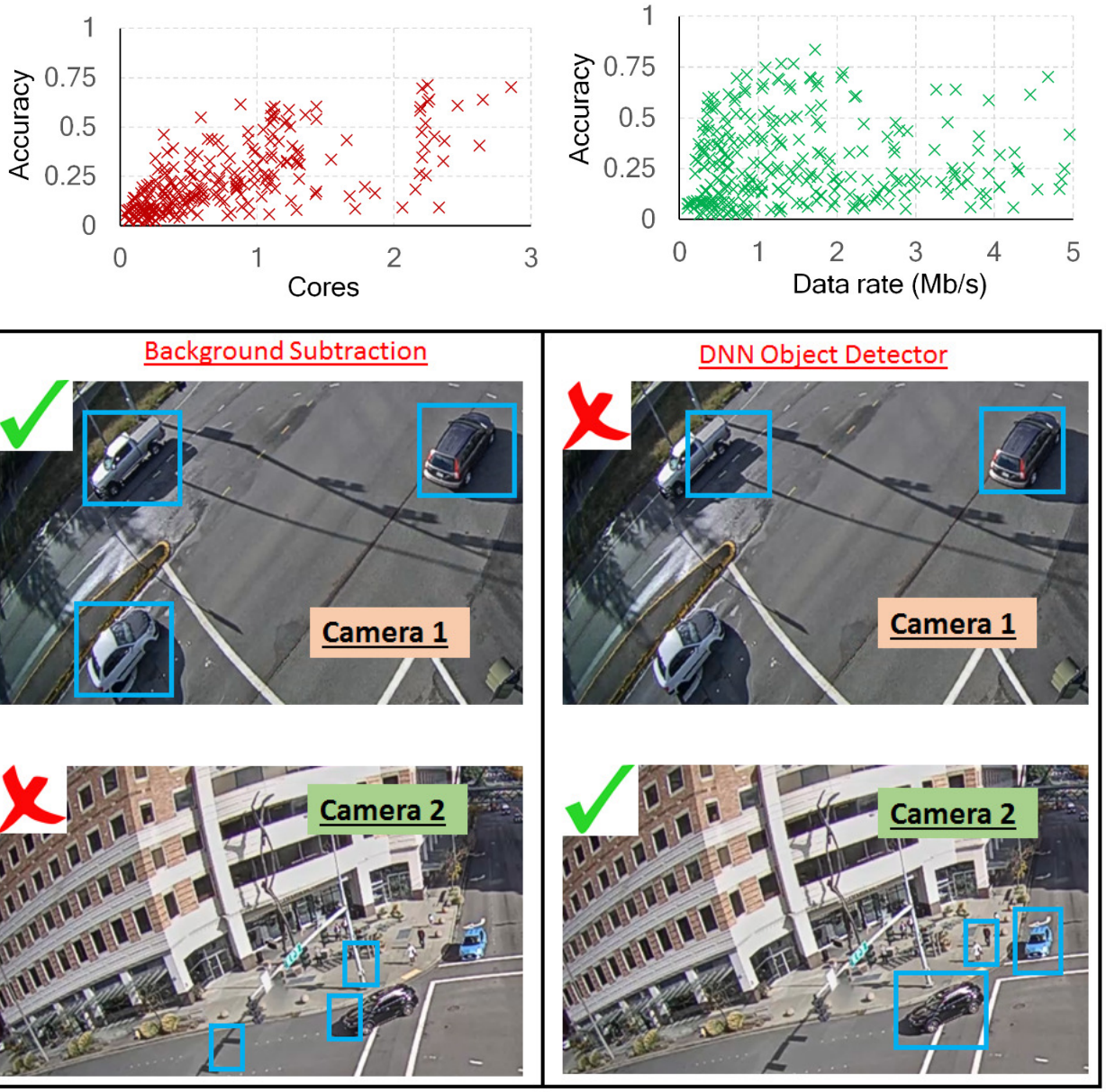

Figure 3: Resource-Accuracy. The graphs on top plot the variation in accuracy for different configurations along with their resource consumptions. The images show the Impact of the implementation choice for object detection (background subtraction vs. DNN) on different cameras.

Profile generation: VideoStorm [4] selectively explores only the promising configurations and avoids wasting CPU cycles exploring the accuracy of configurations that have really low accuracy or high resource demand. In profiling queries that are a pipeline of vision modules, it caches the intermediate results of modules and avoids re-running those modules. Finally, to avoid an explosion of cache space, it profiles configurations with overlapping modules (or "prefixes") together thus saving on both CPU cycles as well as caching.

Scheduling for Accuracy: The choice of configurations is fundamental to resource management because that dictates the resource demands. In a multi-tenant cluster with many video queries, we *schedule for accuracy*, instead of the classic approach of scheduling for fairness that is adopted by modern cluster schedulers like Apache Yarn. We have built VideoStorm to schedule thousands of video queries according to their resource-accuracy profiles as well

as tolerance to *lag* in processing, i.e., the time between arrival and processing of the frames [4]. VideoStorm uses the Model-Predictive-Control and preferentially allocates resources to those queries that provide a higher accuracy for the same quantum of allocated resources. Our system also decides on the *placement* of the modules in the pipeline: execute *each* of them at the camera, private cluster or cloud. This involves considering capacities of multiple resources (compute, network) at all the locations.

Evaluation using real-world video analytics on live camera streams (HD streams) from traffic intersections and indoor surveillance cameras shows that our system: (1) achieves 80% better average accuracy of queries than the competing fair scheduler, and (2) consumes 3.5x lesser CPU cycles for profile generation. Detailed evaluation is available in [4].

4.2 Efficient Deep Neural Networks execution on GPUs

For many vision tasks today, executing Deep Neural Networks (DNNs) on GPUs is vital. The simplest model for executing a DNN via a GPU is for an application to set up and execute the corresponding operations on the GPU by using memory allocation and matrix computation functions provided by a library. Each invocation of the library leads to one or more distinct instances of computational kernels being lined up for execution by the GPU runtime. This simple approach, however, is inadequate in the common setting when several applications seek to perform possibly distinct DNN computations on incoming video.

The complication comes from several sources. First, loading weight matrices corresponding to DNNs into GPU memory is a heavyweight operation and the models might be large enough (e.g., 10-100MB) that they cannot all be preloaded into memory. One thus has to consider when to load and evict individual models. Second, it is significantly more efficient to execute models in batch mode, where the same model is executed in parallel on several inputs, than if a series of heterogeneous models are presented for execution to the GPU. Also, many applications share the models and inputs they execute on. Therefore, it is beneficial to combine models across multiple applications and reduce resource demand. Finally, some models may simply be too large to execute on the edge (e.g., the VGG16 model requires 30GFLOPs) and are best executed in the cloud, if latency permits. We implement the above optimizations in a *DNN Execution Service*, which runs on each machine with a GPU. All DNN requests run against this service which reconciles these complexities across multiple applications.

Approximate scheduling (Section 4.1) relies on the fact that for each DNN, we can apply a set of optimizations to produce DNN variants with differing accuracy and resource use. The machine learning community has proposed several optimizations such as replacing matrices with smaller factors [9], limiting the number of bits used to represent model parameters [11] and reducing the architectural complexity of models in systematic ways [10]. In addition, we have proposed new optimizations such as *specialization*, which replaces large, slow models useful for classifying many classes (e.g., thousands of objects), with much smaller and faster versions specialized for the classes observed at runtime (e.g., under ten objects). These optimizations often reduce resource requirements significantly: the factorized VGG16 model consumes 4.9x fewer cycles at 0.5% accuracy loss.

Our system, MCDNN, gathers these optimizations into a single "optimizing compiler" for DNNs [3]. MCDNN selects the appropriate DNN variant and its placement to optimize the accuracy, latency, and cost of the *whole query pipeline* (not just the DNN). We believe similar design considerations apply to emerging accelerators beyond the GPU such as FPGAs and custom ASICs [5]. While we have omitted the details in the interest of space, please refer to [3] for a full description.

4.3 Virtualizing Steerable Cameras

Cameras are often electronically steerable (pan, tilt, zoom or “PTZ”) and have to support multiple applications simultaneously like amber alert scanning based on license plate recognition and traffic volume monitoring. The primary challenge in supporting multiple such applications concurrently is that the view and image requirements of the applications differ. Allowing applications to directly steer the cameras inevitably leads to *conflicts*. If the license plate recognizer zooms in to get a better view of the text, the traffic volume monitor loses the larger picture and cannot count all the cars in the intersection.

Our solution *virtualizes* the camera hardware. With virtualization, we break the one-to-one binding between the camera and the application. The application binds itself to a virtual instance of the camera and specifies its view requirements, e.g., orientation, resolution, zoom. Our system does its best to provide the most recent view that meets the applications' requirements. The virtual camera abstraction makes steering changes transparent to applications. The insight that led to this design came from observing that temporarily steering the camera away can be masked by replaying the image from the previous view and the impact of incorrect representation of the actual scene is minimized when no significant change is expected in the view during that time. To know when to steer, our software learns the mobility patterns in its view and predicts when motion or change is likely to occur in each of the view it is managing. Traffic systems, for example, exhibit regular, constricted motion patterns. We benefit from this form of motion to learn mobility patterns in the scene. We quickly move the camera to different views needed to support different vision applications. Experiments with a live camera feeds have shown that in a typical traffic intersection setting our system captures up to 80% more events of interest in a wide scene, compared to a system which allows applications to control the cameras directly [6].

4.4 Application-level Optimizations

To further reduce resource demand and tolerate high latencies of executing expensive DNN operations in the cloud, we develop additional application-level techniques.

4.4.1 Intelligent Frame Selection

As mentioned in Section 4.1, the resource demands of certain vision algorithms can make running them on every frame to be prohibitively expensive. Fortunately, most video streams have temporal redundancy among frames making it possible to sub-sample them without losing accuracy. We develop a suite of *content-aware* and *application-aware* techniques to sample only a few key frames for processing without compromising on the accuracy of the application [7].

The core nugget is to use easily-computable "shallow-vision" features to decide whether a frame is worthy of "deep-vision" analysis. We use frame-differencing, which is extremely cheap, to discard frames that haven't changed significantly compared to the previously processed frame. We also compute cheap quality metrics such as blurriness and lighting to decide if the frame is of sufficient enough quality to process. Application-level requirements can help us discard more frames. For instance, the output of an object classifier DNN might remain the same even when the visual content changes significantly. Application-specific sampling both enhances as well as complements application-agnostic sampling.

4.4.2 Intelligent Feed Selection

When a single cameras' coverage is limited because of its position, it is reasonable to deploy multiple cameras, located at different positions, to cover the same physical space. The challenge is, for a given region, more cameras generate more video streams, which require more network bandwidth and computing resources in the cloud. Smart cameras, i.e. cameras with computational abilities or cameras that have access to edge nodes can reduce the network and cloud computing demands while improving the accuracy of the video surveillance system.

The system we have developed processes the incoming video stream at each of the intelligent cameras and/or edge nodes to detect objects of interest [8]. It then uploads only the most important frames from the different video streams to the cloud for further processing. To quantify the frames' importance, we define a new metric called Objects Per Second (OPS). Each frame in each video stream has some number of objects. The goal for our system is to use the minimum amount of bandwidth while maximizing the number of query-specified objects (of interest in the scene) delivered to the cloud. A smart traffic scheduling algorithm uploads frames from only those cameras whose video frames have the most objects that are relevant to the user's query. Our *content-aware uploading strategy* suppresses a large fraction of unrelated video data and in the process and in doing so reduces network utilization while simultaneously increasing the utility of the surveillance system because more cameras can now be supported. With this strategy in place, depending on the amount of activity in the region, we have demonstrated

that for a fixed bandwidth, our video surveillance system can support a geographical area of coverage anywhere between 5 and 200 times greater than an approach that simply streams video from the camera to the cloud. Looking at it another way, for a fixed area of coverage and bandwidth, our system outperforms the default equal throughput (per camera) allocation strategy by delivering up to 25% more objects relevant to a user's query [8].

4.4.3 Delay-tolerance

Real-time vision applications such as augmented reality can suffer from high frame processing delays. DNNs can take multiple frame-times [3] to produce a result; if processing is done over the network, the latencies could be even higher. By the time the application receives the results, they might already be stale – detected objects could have moved to a different location or out of the frame.

We develop a new technique hide such delays to provide a good user experience [7]. Our approach uses local object tracking to project the stale results (object locations) obtained from a *processed frame* to the *current frame*. To make tracking effective since objects can move significantly, we maintain an *active cache* that stores all the subsequent frames from the frame that gets selected for deep-vision processing. When the result of a processed frame comes back, we run tracking from the processed frame, through the cached frames, to catch up with the current frame. When the processed frame contains *newer* objects that may have shown up, we project start tracking them thereon.

5 Deployment

We are actively rolling out traffic analytics solutions based on the software stack in Figure 2. A multi-modal object counter is running 24x7 since December 2016 off live traffic cameras in the cities of Bellevue, WA and Cambridge, UK to enable the cities to understand volumes of cars, pedestrians and bikes. The counters implement the interfaces specified above, runs on a small distributed cluster of machines, and provides directional counts at the intersections being monitored. Output from these counters will feed actuation systems that control traffic light durations. We use wide-angle cameras to cover the whole intersection with a single camera. Each server is equipped with the NVIDIA GTX 1080 GPU to run our custom DNN to recognize cars, pedestrians, and bicycles.

Figure 4 shows the counter's video analytics pipeline as well as a sample of the output of directional counts over time. Using crowdsourcing based ground truths, we see that our counts are over 95% accurate. Using the techniques in Section 4 for picking resource-efficient configurations, each counter pipeline consumes less than 15% of a quad-core server. We are in the process of engaging with many other cities for a larger rollout of our traffic analytics.

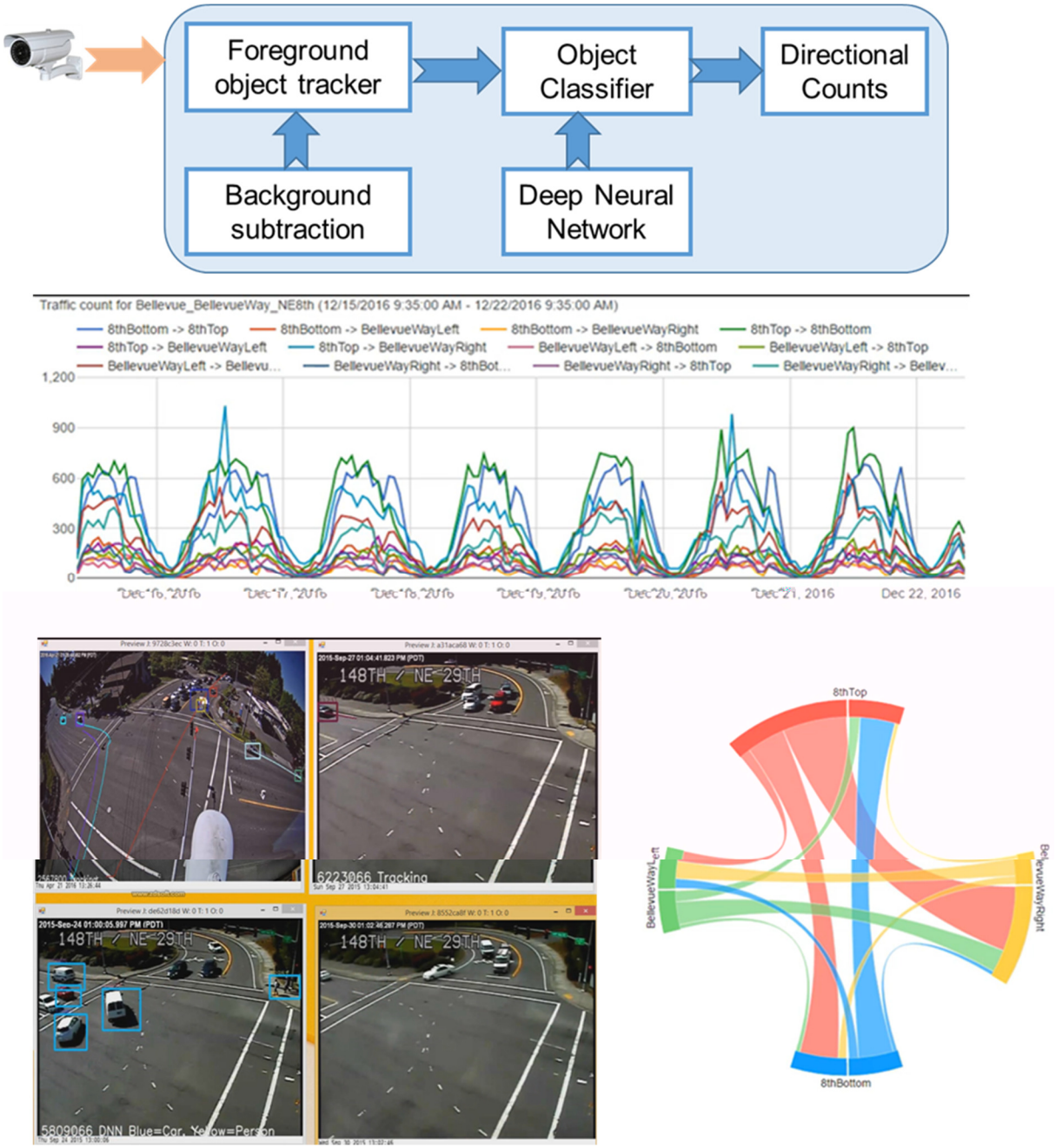

Figure 4: Production deployment details of traffic analytics at Bellevue, WA based on live traffic intersection camera streams. The figure shows a dashboard of multiple camera feeds being analyzed simultaneously, the video analytics pipeline, and temporal graphs of directional volumes.

References

[1] The Vision Zero Initiative. http://www.visionzeroinitiative.com/

[2] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, I. Stoica, "*Low Latency Geo-distributed Data Analytics*", ACM SIGCOMM 2015.

[3] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, A. Krishnamurthy, "*MCDNN: An Approximation-Based Execution Framework for Deep Stream Processing Under Resource Constraints*", ACM MobiSys 2016.

[4] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, M. J. Freedman, "*Live Video Analytics at Scale with Approximation and Delay-Tolerance*", USENIX NSDI 2017.

[5] N. Jouppi, "*Google supercharges machine learning tasks with TPU custom chip*", May 2016.

[6] S. Jain, V. Nguyen, M. Gruteser, P. Bahl, "*Panoptes: Servicing Multiple Applications Simultaneously using Steerable Cameras*", ACM/IEEE IPSN 2017.

[7] T. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "G*limpse: Continuous, real-time object recognition on mobile devices*", ACM SenSys 2015.

[8] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee. "*The design and implementation of a wireless video surveillance system*", ACM MobiCom 2015.

[9] Y. Kim, E. Park, S. Yoo, T. Choi, L. Yang, D. Shin, *"Compression of deep convolutional neural networks for fast and low power mobile applications",* ICLR, 2016.

[10] Y. Guo, A. Yao, and Y. Chen, *"Dynamic network surgery for efficient DNNs",* NIPS, 2016.

[11] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, *"Quantized neural networks: Training neural networks with low precision weights and activations"*, arXiv preprint arXiv:1609.07061v1, 2016.

[12] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, M. Satyanarayanan, "*Scalable Crowd-Sourcing of Video from Mobile Devices*", ACM MobiSys 2013.

[13] J. Wang, B. Amos, A. Das, P. Pillai, N. Sadeh, M. Satyanarayanan, "*A Scalable and Privacy-Aware IoT Service for Live Video Analytics*", MMSys 2017.

Bio

Ganesh Ananthanarayanan (ga@microsoft.com), Paramvir Bahl (bahl@microsoft.com), Peter Bodik (peterb@microsoft.com), Krishna Chintalapudi (krchinta@microsoft.com), Matthai Philipose (matthaip@microsoft.com), Lenin Ravindranath (lenin@microsoft.com) and Sudipta Sinha (sudipsin@microsoft.com) are members of the Mobility and Networking Group at Microsoft Research. The group conducts research in all aspects of systems & networking and regularly publishes in the leading conferences while also deeply influencing Microsoft's products.