Bandwidth-Aware Data Filtering in Edge-Assisted Wireless Sensor Systems

Igor Burago, Marco Levorato
Department of Computer Science
University of California, Irvine
{iburago, levorato}@uci.edu

Aakanksha Chowdhery
Department of Electrical Engineering
Princeton University
aakanksha@princeton.edu

***Abstract*—By placing processing-capable devices at the edge of local wireless access networks, Edge Computing architectures have been recently proposed to connect mobile devices to computational power through a one-hop low-latency wireless link. In this paper, we propose a new design where edge assistance is used to control local data filtering at the mobile devices in bandwidth and energy constrained systems. We focus on real-time monitoring applications, where the video input from mobile devices is processed to centrally detect and recognize objects. The edge processor controls the activation and deactivation of local classifiers implemented by the mobile devices to remove useless portions of video frames. The objective is to adapt the video stream to time-varying bandwidth constraints, while minimizing the additional energy consumption introduced by local processing. To this end, an optimization problem is formulated for a loss function embodying the balance between the risk of violating the available bandwidth and the cost of overly-conservative data filtering. The edge assists the local decision by extracting parameters of the video, such as density of objects of interest in a frame, which influence the output of the sensor. Numerical results, obtained by performing a measurement campaign based on a real implementation, illustrate the tension between energy and bandwidth use for a Haar-feature based cascade classifier.**

I. Introduction

Cloud computing architectures have been widely explored to offload storage and computational burden from mobile devices limited in energy and processing capabilities that prevent intense local data processing. However, the delay introduced by the multi-hop structure of the Internet core limits the ability of the mobile and cloud ends to interact promptly [1].

By introducing computationally-capable devices within local wireless access networks, recently proposed Cloudlets and Edge architectures [1], [2] extend the practicality of offloading real-time processing tasks to compute-intensive applications. These architectures create wireless islands where sensing, processing, and actuation capabilities are contained within a low-latency one-hop wireless topology. Examples of increasingly relevant applications that could benefit from fog and edge computing are vehicular and transportation networks. This family of applications requires tight control loops imposing stringent delay demands, under which the traditional cloud architecture cannot serve as a viable solution.

One of the biggest challenges in realizing effective edge-assisted services is the constrained bandwidth that is intrinsic to the wireless links connecting the mobile devices to the edge processor. Additionally, the available bandwidth may present considerable variability over time due to existing services which may generate competing traffic over the same channel or network resource. The ability to promptly adapt the amount of data exchanged between the mobile sensor and the edge processor is, then, instrumental to ensure reliable and feasible service. Inspired by these advancements and challenges, in this paper we design and test an edge-assisted architecture for acquisition, transmission, and processing of video input in resource-poor and bandwidth-constrained mobile environments.

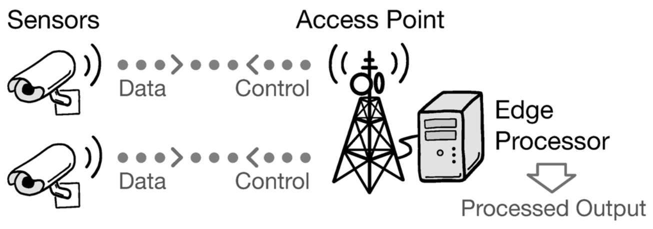

Figure 1. Organization of the edge-assisted wireless sensor streaming driven by the edge processor based on the current network and data stream conditions.

Fig. 1 depicts the components of the general scenario considered in this paper: The edge processor analyzes video input acquired by the mobile devices (which we refer to as sensors in the following) to accomplish some computational task. In the assumed scenario, the edge processor identifies pedestrians in the captured video streams to perform classification and further analysis. For instance, the edge processor could aim at the detection of imminent dangers to pedestrians, or the identification of people in a database to create security alerts. The sensors and edge processor are interconnected through bandwidth-constrained wireless links.

One of the key components of the proposed architecture is a form of local "semantic" filtering of the video input collected at the mobile cameras, where portions of the video *uninteresting* to the global detection and classification objective are removed from the data streams sent to the edge processor for further analysis. This pre-filtering reduces the bandwidth needed to transport parts of the video relevant to the task of the edge processor. Importantly, the filter can be dynamically adapted to the application's requirements. The reduction is a function of current video parameters, such as the density of objects within the picture. Our measurements show that pre-filtering can make the size of the transmitted frames by up to an order of magnitude smaller compared to video compression alone. However, such bandwidth-effective filtering technique has the drawback of a significant energy expense imposed by the additional computational load on the mobile devices. On the one hand, continuously transporting the video streams to the edge processor may exceed the available bandwidth. The consequence is an excessive delay in generating the output at the edge processor and, in those cases where the wireless resource is shared, disruption of competing services. On the other hand, permanent usage of filtering at the sensors is likely infeasible due to computational and energy restrictions.

At the center of the proposed system lies an information loop between the edge processor and a mobile sensor. The current and prognosticated parameters of the video stream and network load, extracted at the network edge, inform the local control activating filtering on the sensor. For the application considered in the following, the edge processor provides the sensor with two characteristics: the anticipated density of objects of interest in the next frame, and the amount of bandwidth that will be available for the next sensor transmission. The sensor uses this information to activate and deactivate the object-filtration procedure, based on an optimization problem which aims at the minimization of a loss function encapsulating both the risk of bandwidth-constraint violation and the cost of overly-conservative filtering activation.

The core contributions of this paper are:

- A novel view on the sensor-edge system, that leverages local data filtering of video streams transmitted to the edge to optimize the trade-off between energy and bandwidth used by the sensor.
- Derivation of optimal control to adaptively filter data at the local sensor based on available bandwidth, formulated as a loss-minimization problem.
- Evaluation on a real-world testbed showing that local adaptive control assisted by the edge achieves the desired trade-off between energy use and bandwidth use.

The rest of the paper is organized as follows. Section II presents the proposed architecture and its operational structure. In Section III, we formulate the optimization problem for the edge-assisted sensor control. Section IV describes the specific implementation of the system and the application used to obtain the evaluation results given in Section V. Section VI discusses related work, and Section VII concludes the paper.

II. System Description and Architecture

The concept of edge-assisted local energy and bandwidth control in resource-restricted environments proposed herein finds applications in a wide range of computationally intensive scenarios. We make our contribution concrete by realizing the proposed architecture for a specific application.

A. *System Architecture*

We focus on the class of remote video processing applications in distributed sensor systems, where individual sensors locally acquire video to be further processed for the purposes of a system-wide objective. In the most general scenario, the video streams are to be processed centrally to perform some classification or monitoring task, such as situational awareness and identification of dangers in an urban environment.

The edge processor functions as an intermediary between the two extremes of the architecture: (i) resource-poor sensors, which individually do not possess enough energy, computational or information resources to accomplish the system-wide objective; (ii) urban data centers, which are connected to the local sensors through wireless and wireline links of a constrained capacity with some communication delay. Thus, on the one hand, data cannot be processed only locally by the sensors themselves. On the other hand, delivering the continuous stream of data to the data center may not be feasible due to bandwidth limitations and excessive delay.

Computational resources at the edge of the wireless network can resolve the above-illustrated impasse. Transportation of the data streams only through the initial wireless one-hop link can result in a considerable reduction in the delay between data acquisition and the final outcome of the detection process. We observe that the edge processor may either substitute the functions of the data center, or operate as a data filter, reducing core network load. However, the issue of the time-varying and insufficient capacity of the wireless links, which may be shared with other services, remains.

The main subject of this paper is an architecture that uses the proximity of the edge processor to the local system — the sources of the video streams and the local wireless network — to optimize resource usage within the wireless island. The key components of the architecture are a semantic data-filtering technique and an edge-assisted control loop for the on-the-fly adaptation of data filtering at the sensor. The architecture gives a structural way to balance the energy and bandwidth consumption between the sensor devices and the edge processor, with all other characteristics of the observed video streams being fixed. In other words, it is the selection of the content to be transmitted that is the subject of the adaptation, and not the parameters of its encoding or representation.

Semantic Data Filtering. General-purpose video compression techniques are not well suited for object detection or tracking tasks, as those techniques are agnostic to the content within the video. In particular, modern video encoders take advantage of spatial and temporal correlations in the video, but they do not identify which regions of a frame contain the objects of interest. The lossy compression schemes can degrade video quality to meet a required bandwidth target, but are not suited to object detection or tracking. Further, conventional video compression cannot adapt the video bitrate when the available bandwidth fluctuates rapidly on the wireless channel. Finally, traditional techniques may not compress the video efficiently when its background varies intensely.

Herein, we propose to use a binary classifier at the local sensor nodes to filter out the portions of the individual frames that are unnecessary to the final computational objective. Note that such filter can be adjusted by the edge to match time-varying or context-specific computational objectives. As we will

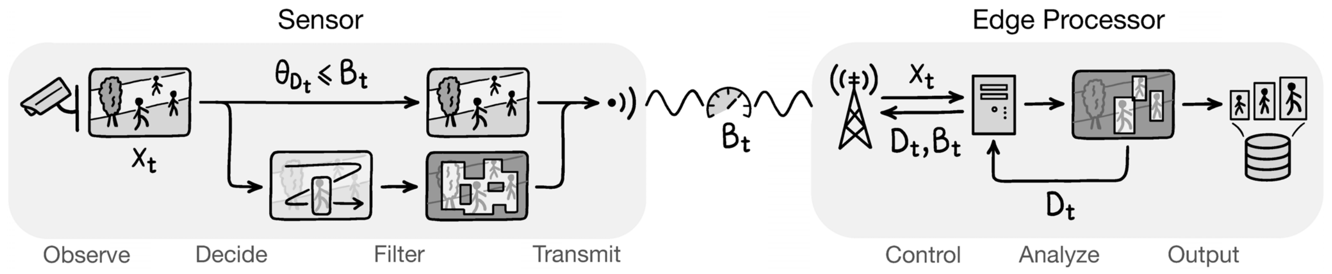

Figure 2. Operational diagram of the edge-assisted data filtering control.

demonstrate using real-world measurements in Section V, such technique provides significant compression gain compared to standard compression. The specifics of the video filtering used in the considered application are detailed in Section IV.

Edge-Assisted Adaptation. Filtration of the data stream with the semantic classifier reduces the absolute bandwidth used to transport the video, thus minimizing the probability of exceeding the available bandwidth. However, continuous operation of the classifier imposes a considerable computation and energy expense burden to the resource-limited mobile node. Ideally, the sensor should activate the classifier only when the bandwidth is insufficient to transport the unfiltered (but compressed) video stream. However, there are two major challenges to realize such objective: (i) the sensor may be unaware of the overall bandwidth perceived at the edge processor; (ii) the output size of the filtered frames is a function of the video's characteristics, such as object density within the frame and other parameters unrelated to the computation objective, which may influence the compression gain.

To overcome these issues the proposed architecture not only makes use of the edge as a low-latency substitute for the data center residing within the network core, but establishes an information flow downstream from the edge processor to the sensor to enable local control. Note that to extract such information, the edge processor is assumed to use the same filtering procedure on the frames received unfiltered, as the one used at the sensor node. We assume that the edge processor has enough energy and computational power to perform such operations, which are indeed a necessary first step to accomplish the global identification task.

B. Information Flow and System Operations

Let us now discuss the operations of the system, from the capture of an individual video frame at the sensor to the fully processed output at the edge, going through the components of local processing, information exchange, and control loop. The sensor-edge system processes captured video in the following four stages, schematically depicted in Fig. 2.

Sensor: Observe and Process. When a new frame is captured by the camera, it is handed over to the sensor's processing and control unit. The sensor, then, has two choices: (i) Stop the processing of the input frame, and transmit it unaltered to the edge, leaving the object detection work to be done at the edge. (ii) Continue processing the frame by running object detection,

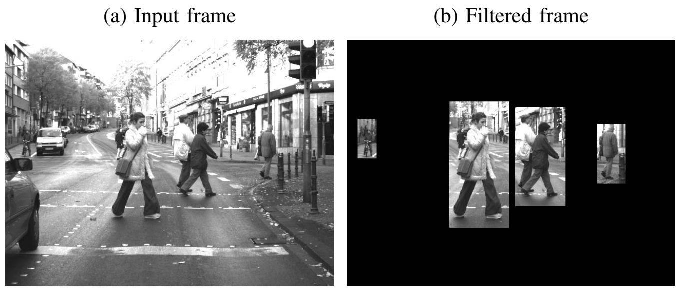

Figure 3. Video frame filtered with a Haar feature-based pedestrian classifier.

and reduce the full frame to the portions where objects are detected. Choice (ii) minimizes the required bandwidth, while choice (i) minimizes the energy consumption, which is fully justified when object detection and transmission of the resulting reduced frame requires more energy than transmission of the original frame (in our experiments, the former used, on average, 2.14 times more energy and was 2.12 times longer).

Sensor: Transmit. The streaming encoder compresses the unaltered or reduced frames and transmits them to the edge with an indicator of whether the sensor applied filtering locally.

Edge: Complete Processing. The edge receives the frame and, if the object detection has not been done locally at the sensor, applies the detection procedure to this incoming frame, reducing it to the regions that contain target objects. Regardless of whether the frame is filtered on the sensor or edge side, the final object detections on the edge remain the same, as the detection procedures used by both sides are identical; it is only the place of execution that is changing. With the frame fully processed, the edge estimates the *density* of objects in the frame as the ratio of the number of pixels that contain objects to the total area of the frame.

Edge: Analysis and Feedback. Based on the results of the processing of the current frame, as well as some history of past frames and network conditions, the edge sends the feedback to the sensor. The feedback consists of two components: (i) the object density that the sensor should expect in the following frame, and (ii) the estimated amount of bandwidth that the sensor should not exceed in the next transmission.

The object-detection procedure uses a binary *classifier* algorithm to identify whether a frame contains target objects or not. The classifier scans the current frame with sliding windows of varying sizes to detect objects in each window.

If the window contains a target object, the frame keeps the corresponding region, otherwise it filters that region out. We assume that the sensor receives all parameters necessary to run the classifier beforehand. The resulting collection of object-containing regions detected at the sensor is transmitted to the edge. To efficiently transmit the images in the regions of detection, we use the following simple technique. Each frame is compressed with the streaming video encoder and transmitted as usual, but with the pixels that do not belong to any detected object set to the black color (see Fig. 3). More sophisticated compression schemes taking advantage of filtered-out areas are subject to future work.

This straightforward compression technique provides considerable gain in the average output frame size, since the blacked-out parts of the frame are encoded with great efficiency. Further, applying a particular video compression scheme can lead to even higher gains in compressed frame sizes. The compression gains are high in application scenarios when the background is dynamic, and video compression alone does not obtain sufficient bandwidth savings. Such application scenarios include video captured by mobile agents or intelligent vehicles, where only a specific class of moving objects is targeted (e.g., pedestrians on a busy street).

III. Optimization Problem

By design of the system, it is the interplay between the two key aspects that define its operational optimality: the power consumption by frame filtration and the amount of channel bandwidth utilized for transmission. Indeed, an increase in the number of filtered frames reduces the network load but raises the energy use, and vice versa. Therefore, construction of the mechanism controlling the activation of local video filtering has to be governed by the trade-off between these two factors.

In order to quantitatively evaluate the efficiency of the system, we will use the loss function $\ell(u)$ describing this trade-off in more precise terms:

$$\ell(u) = u \cdot \big(\alpha - \mathbb{1}[u < 0]\big), \tag{1}$$

where $\alpha \in (0, 1)$ is a weighting parameter, and the variable u has the meaning of the difference between the true size b_t of some tth video frame and an estimation φ_t, so that for each frame $u_t = b_t - \varphi_t$.

The utility of the loss $\ell(u_t)$ is twofold. If $u_t > 0$, it penalizes for the risk of loosing a frame by underestimating its size and, therefore, not activating the filtering when it could have saved the frame transmission from failing. Conversely, if $u_t < 0$, $\ell(u_t)$ discourages channel under-use that causes excessive energy consumption by unnecessary filtration. Thus, the loss function factors in the incentives for both energy conservation and thrifty channel use, in proportion to weights $1 - \alpha$ and α.

Here we adopt the assumption of a deterministic transmission success policy. Namely, if the size of a transmitted frame is greater than the available channel bandwidth at the moment of the transmission, its success is not guaranteed, and the frame is considered to be lost. Otherwise, the frame is assumed to successfully reach the edge processor.

The goal of the sensor control, then, can be defined to maintain the estimation φ_t minimizing the expected loss over the lifetime of the sensor:

$$\mathbb{E}\big[\ell(b_t - \varphi_t(\delta_t, \theta)) \,\big|\, \delta_t \in D_t\big] \longrightarrow \min_{\theta}, \tag{2}$$

where the expectation is taken over the video frames x_t with corresponding true sizes $b_t(x_t)$ and features $\delta_t(x_t)$ that can be used in the model of φ_t, which also becomes a function of a parameter vector θ. The form of the estimation $\varphi_t(\delta_t, \theta)$ has to be chosen beforehand and is assumed to be free up to θ. This choice should be guided by the a priori knowledge of the distribution of frames in the observed video stream and the specifics of the video compression used to encode it. In addition, the sensor has access to short-term contextual information about the anticipated features δ_t of the newly observed frame x_t, obtained through the assistance of the edge processor providing updates to the feature region D_t over time.

Switching to the empirical version of the problem (2) and taking into account the form of the loss function (1), we obtain the following final problem:

$$\sum_{\delta_t \in D_t} \ell(u_t) = \alpha \sum_{\substack{\delta_t \in D_t \\ u_t > 0}} |u_t| + (1 - \alpha) \sum_{\substack{\delta_t \in D_t \\ u_t < 0}} |u_t| \longrightarrow \min_{\theta}, \tag{3}$$

where the loss function parameter α and its counterpart $1 - \alpha$ are easily interpretable as weights defining the relative significance of over- and underestimation, respectively. The meaning of α can be further clarified, as it is well known [3] that the solution to problem (3) is the αth quantile function of the distribution of frame sizes b_t, conditioned on $\delta_t \in D_t$. Hence, the parameter α has the statistical sense of the average proportion of frames whose delivery success the system is not willing to risk due to possible channel overuse.

In general, although the function $\varphi_t(\delta_t, \theta)$ can be arbitrarily complex, for computational purposes we are interested in simple models. Luckily, when the feature regions D_t can be made small, such that the values of φ_t do not exhibit great variation inside each region, φ_t can be reduced to the conditional constant function

$$\varphi_t(\delta_t, \theta) \triangleq \theta_{D_t}, \tag{4}$$

where θ_{D_t} is nothing but the αth conditional quantile calculated over a subsample of frames for which $\delta_t \in D_t$.

So far, δ_t denoted an abstract vector of frame features. As we can see now, it is in our best interest to find one or more features that (i) form regions of constant-like φ_t, and (ii) deliver sufficiently informative subsamples to obtain desired estimations. One easily computable feature is the density of objects in the frame, defined as the ratio of the area occupied by objects, as decided by the filtering procedure, to the total area of the frame.

When the quantile θ_{D_t} as a function of the frame object density δ_t does not change significantly, it allows us to use this feature to subdivide video frames x_t into density intervals D_t with separate sample quantiles in each. As we will see in Section V, this property holds true for the object density at least in the evaluation video dataset we used.

Optimal Sensor Control Procedure. Putting it all together, we construct the following control algorithm for the edge-assisted on-sensor data filtering (see Fig. 2). As each successive frame x_t of the video stream is observed, the sensor:

1) Receives a feature region D_t and a bandwidth constraint B_t from the edge processor.
2) Queries the sample α-quantile θ_{D_t} for the region D_t.
3) Compares θ_{D_t} with B_t, and activates the filtering for the current frame x_t if $\theta_{D_t} > B_t$, or leaves it unaltered otherwise.
4) If the frame x_t was filtered, computes the resulting density $\delta_t(x_t)$ and updates the sample quantile θ_D for the region $D \ni \delta_t$.
5) Encodes and transmits the frame x_t (filtered or not) to the edge as the next frame in the video stream.

The above algorithm is agnostic to the nature of the object filtration procedure, and, as it follows from the properties of problem (3), is optimal.

To improve the timeliness in quantile updates when on-sensor filtering is infrequent, it can be additionally assumed that the edge processor also maintains the sample quantiles for the feature regions and periodically updates the sensor with them.

IV. Implementation and Experimental Setup

A. Experimental Setup

For the purposes of experimental evaluation, the role of the end-device hosting the sensor was played by a single-board computer Raspberry Pi 3 Model B with a 1.2 GHz quad-core ARM Cortex-A53 CPU and 1 GB of LPDDR2 RAM.

Throughout experiments, neither the frame processing nor video compression and transmission relied on the limited GPU acceleration capabilities of Raspberry Pi in order to avoid obscuring general trends in energy measurements with gains specific to the particular video application chosen for demonstration.

A sequence of frames from a continuous video recording was preloaded on the end-device to emulate video capturing on the sensor. Frame arrival rate was set to the frequency based on the timestamps available in the dataset. Frames were then sequentially processed by our custom software implementing the sensor's logic, including communications with the edge processor, frame filtering, quantile updates, and submission of frames to the stream encoder.

To encode the outgoing video, we used the MPEG-4 AVC (H.264) video compression algorithm as implemented in the VideoLAN x264 open-source library. The encoder was used in the variable bitrate mode with the quality of output video set to the same constant level maintained over all transmissions. The sizes of the compressed frames on their way out to the network stack were monitored and logged.

For the sensor-edge communication, the Raspberry Pi's built-in 2.4 GHz Wi-Fi 802.11n chip was used to connect over UDP to the edge processor hosted on a stationary computer connected to the same Wi-Fi network as the end-device.

The power consumption of the end-device was measured using the ODROID Smart Power device providing measurements accurate within 1 mW every 100 ms. The resulting power traces, time-aligned with the activity of the sensor, were integrated over time to obtain the energy consumption of the device within the 1 mJ resolution.

B. Application and Video Processing

As a realistic example of an application that can benefit from the proposed on-sensor adaptive filtering, we choose a simple pedestrian monitoring task whose objective is to detect and extract images of all pedestrians occurring in a given video stream. To this end, the system has at its disposal a pre-trained pedestrian classifier capable of deciding whether a given image depicts a pedestrian [4]. Having this building block, the sensor-edge pair running the monitoring task filters the video stream frame by frame, every time scanning each with a rectangular search window and repeatedly applying the classifier to the portion within the window at its current location. The windows identified as pedestrians, then, make up the output of the system.

For our experiments, we focus on the family of linear Haar-feature based binary classifiers [5] which are known to be a practical choice for CPU-bounded devices due to their low computational requirements. These classifiers are constructed as an ensemble of weak classifiers $h_k(z)$ implementing thresholding on a single feature $f_k(z)$ from the set of features induced by a limited basis of Haar patterns for all possible rectangles within a window z to be decided. Every rectangle generates one feature per basis pattern, with its value $f_k(z)$ being a weighted sum of pixel intensities in the rectangle. The binary decision rule of each weak classifier has the form

$$h_k(z) = \mathbb{1}[\iota_k f_k(z) < \iota_k \tau_k], \tag{5}$$

where τ_k is the threshold and $\iota_k = \pm 1$ is the polarity of the comparison.

The sought ensembled classifier $h(z)$ is amalgamated from these weak classifiers $h_k(z)$ via the AdaBoost procedure, making the final decision rule a linear combination of a number of weak classifiers' decisions:

$$h(z) = \mathbb{1}\left[\sum_{t=1}^{T} \gamma^{(t)} h_{m(t)}(x) \geq \frac{1}{2}\sum_{t=1}^{T} \gamma^{(t)}\right], \tag{6}$$

where the coefficients $\gamma^{(t)}$ are obtained through the following process of supervised learning with boosting.

Initially, the classifier is empty, and all available training windows z_l, labeled with their true classes $y_l \in \{0, 1\}$, are considered to have uniform weights $w_l^{(1)}$ within each of the two classes. On each iteration $t = 1, 2, \ldots$, the classifier is expanded to include one more feature into the ensemble. For this purpose, all weak classifiers $h_k(z)$ are retrained on the given samples (z_l, y_l) in accordance with their current weights $w_l^{(t)}$, so that each minimizes the error rate

$$e_k^{(t)} = \sum_l w_l^{(t)} |h_k(z_l) - y_l|. \tag{7}$$

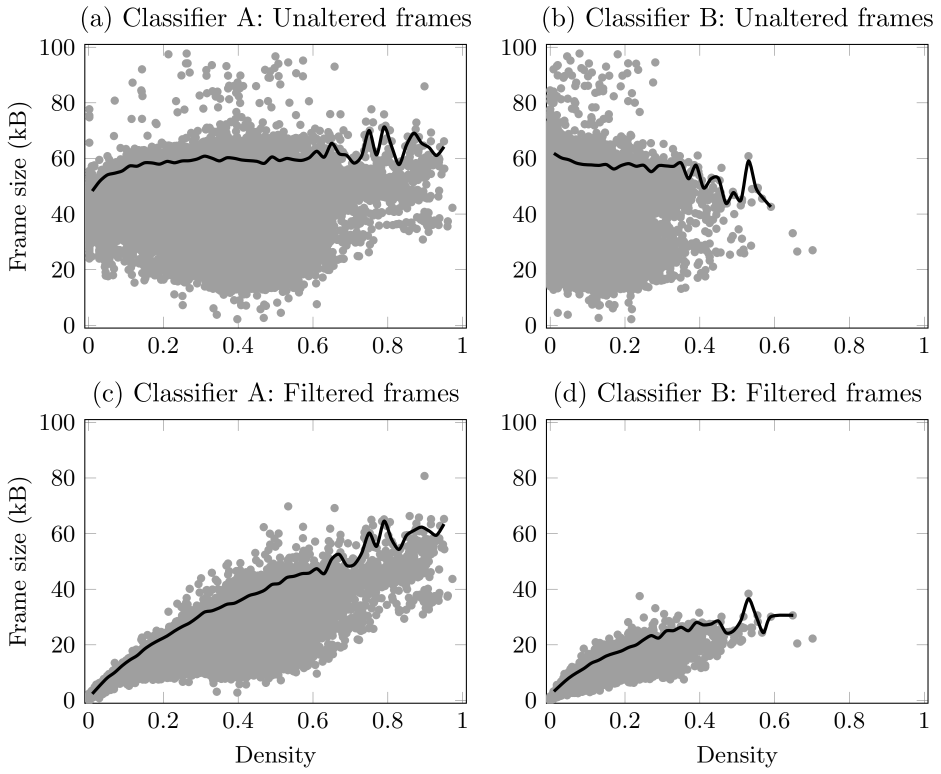

Figure 4. Sizes of unaltered (a, b) and filtered (c, d) frames in compressed video stream as a function of the density of objects detected in them by two different classifiers. Black lines depict the corresponding 95%-quantiles.

The classifier $h_{m(t)}(z)$ with the lowest error rate $e^{(t)}_{m(t)}$ is chosen for addition with the coefficient

$$\gamma^{(t)} = -\log \varepsilon^{(t)}, \tag{8}$$

$$\varepsilon^{(t)} = e^{(t)}_{m(t)} \Big/ \big(1 - e^{(t)}_{m(t)}\big), \tag{9}$$

and the sample weights are updated and renormalized:

$$w_l^{(t+1)} \propto \begin{cases} w_l^{(t)} \varepsilon^{(t)}, & \text{if } h_{m(t)}(z_l) = y_l; \\ w_l^{(t)}, & \text{otherwise.} \end{cases} \tag{10}$$

The training is stopped when either a desired ensemble size or accuracy is reached.

We implemented two versions of this Haar-feature based classifier, herein referred to as Classifiers A and B. *Classifier A* processes the windows sequentially in a single thread applying the full ensemble of hundreds of weak classifiers, as shown in (6). *Classifier B* has approximately five times more features, but is multi-threaded and uses the cascade heuristic [5]. This heuristic breaks the ensemble into a sequence of stages, each consisting of just a portion of terms in the sums in (6), that are evaluated cumulatively and in order, until the window in question is either rejected or the cascade is exhausted to confirm a detection. The technique helps to avoid paying the full computational price of a big ensemble for every classifier use, as a window without a pedestrian is often rejected after going through the first few stages. Both classifiers were trained on the Daimler Pedestrian Detection Benchmark Dataset [6].

C. Network Model and Bandwidth Constraints

As we saw in Section II, the nature of the adaptive on-sensor control requires an input of the per-frame bandwidth constraints B_t communicated to the sensor by the edge. For performance testing of the control, we used a simple model of local wireless network in order to generate realistic bandwidth traces simulating the traffic activity of an environment of coexisting devices.

The model rests on standard assumptions about the character of traffic needs that are typically made in generic settings. Namely, we model a number of concurrent wireless nodes whose traffic arrival processes are stationary and governed by Poisson distributions. Nodes are assumed to have finite lifespans of activity that follow binomial distributions. Finally, with small probability, the set of active nodes grows to accept a newly connected node with randomly chosen parameters of its traffic generator.

Using this model, we generate the time series of bandwidth requirements for each coexisting node and then subtract them all from the channel's capacity and align with the schedule of frame arrivals on the sensor to finally obtain constraints B_t that can be used to evaluate the sensor. To test the control under different loads, we differentiate resulting bandwidth traces by the *filtering stress* they induce on the sensor, i.e., the fraction of frames that had to be filtered before sending. In the following, we show evaluation results obtained for simulated network loads of different levels of filtering stress.

Table I
AVERAGE ENERGY CONSUMPTION, BANDWIDTH VIOLATIONS AND BANDWIDTH SLACK PER FRAME FOR THE ADAPTIVE FILTERING STRATEGY USING TWO CLASSIFIERS FOR DIFFERENT NETWORK LOADS VERSUS THE TRIVIAL STRATEGIES OF NO FILTERING AND FULL-TIME FILTERING

Filtering	Processing energy (mJ)		Streaming energy (mJ)		Bandwidth violations (%)		Bandwidth slack (%)	
Net. load	Classifier A	Classifier B	Classifier A	Classifier B	Classifier A	Classifier B	Classifier A	Classifier B
None								
Low	36	41	94	97	14	14	19	19
Medium	"	"	"	"	30	30	4	4
High	"	"	"	"	54	54	−7	−7
Adaptive								
Low	358	99	81	76	2	2	37	42
Medium	629	158	69	60	3	2	37	53
High	922	210	57	46	3	2	47	69
Full-time								
Low	1227	268	45	28	0	0	69	90
Medium	"	"	"	"	0	0	63	89
High	"	"	"	"	1	0	59	88

V. EVALUATION

The proposed system architecture was evaluated on the hardware described in Section IV-A, for the sensor control procedure introduced in Section III with the weighting parameter α set to 0.95, pedestrian filtering at up to 15 frames per second using Classifiers A and B implemented as discussed in Section IV-B, and three classes of simulated network load at the low, medium, and high filtering stress levels of 25%, 50%, and 75%, generated as explained in Section IV-C. The sequence of 21790 video frames comprising the test portion of the Daimler Pedestrian Detection Benchmark Dataset [6] played the role of the evaluation video stream observed and transmitted by the sensor.

Fig. 4 shows how the sizes of all compressed frames in the evaluation video are distributed as functions of their object density. The 95%-quantiles for different density values are shown with black lines. Fig. 4(a) and 4(b) display the distributions for the full input frames as they appear in the original video. For this reason, the y-coordinates of the points in these two plots are identical. The x-coordinates, though, differ as they correspond to the frames' object density estimated from the output of filtering with Classifiers A and B, respectively.

As we can see from the quantile line, the sizes of unfiltered frames in both cases do not exhibit a significant dependence on the object density. In other words, the visual complexity of the objects (pedestrians) in this video is close to that of the background (trees, cars, city streets). The flatness of this quantile line means that the number of density regions D used in the control procedure from Section III can be small, which makes the subsamples for each D more representative and the corresponding estimations θ_D more statistically significant.

Fig. 4(c) and 4(d) show the distributions for the same frames but filtered with both Classifier A and B. Comparing the pairs of plots 4(a), 4(c) and as 4(b), 4(d), we see the confirmation of our expectations: The fewer pedestrians there are in a frame and, therefore, the bigger is the blacked out part of it, the larger are the savings in the compressed frame after filtration. The difference in scale between 4(c) and 4(d) is explained by the higher efficacy of Classifier B as compared to Classifier A due to a greater number of features used in the former, making the latter more conservative. Fig. 3 shows an example of a frame from the evaluation video filtered with Classifier B.

Table I summarizes the numerical results obtained throughout all of our experiments. In addition to the evaluation results for the proposed adaptive strategy, results for the two baseline strategies are also presented: when the filtering is always disabled (*None*) and, conversely, when the filtering is always enabled (*Full-time*). These baselines represent the two extremes of the spectrum of possible sensor behaviors. In the former strategy, frame processing consumes almost no energy due to the pedestrian filtering being inactive, but makes the sensor suffer from high frame loss due to frequent violations of the bandwidth constraint (in more than half of the cases under high network loads). In the latter strategy, the processing energy for each frame raises by an order of magnitude (or more for less efficient single-threaded Classifier A), and the energy required for compression and transmission of the stream drops by more than a half, while the frequency of bandwidth violations plummets towards zero.

The proposed adaptive strategy, resulting from the control procedure introduced in Section III, manages to balance the extremes of the baselines in all three measures: As expected, the energy for both processing and streaming of the frame in the adaptive case is smaller than the corresponding maximums achieved in the cases of no filtering and full-time filtering. Further, the processing energy increases together with the network load as it imposes higher filtering pressure levels on the sensor. The streaming energy, to the contrary, decreases as the filtering pressure strengthens due to reducing computational cost of compressing frames consisting of more and more blacked out regions. In both cases, the relation between the energy and filtering pressure is close to linear. Most importantly, regardless of the network load, the sensor is able to achieve guaranteed frame delivery for the vast majority of the frames, violating the bandwidth constraints in less than 3% of the stream duration (for the tolerance of $1 - \alpha = 5\%$).

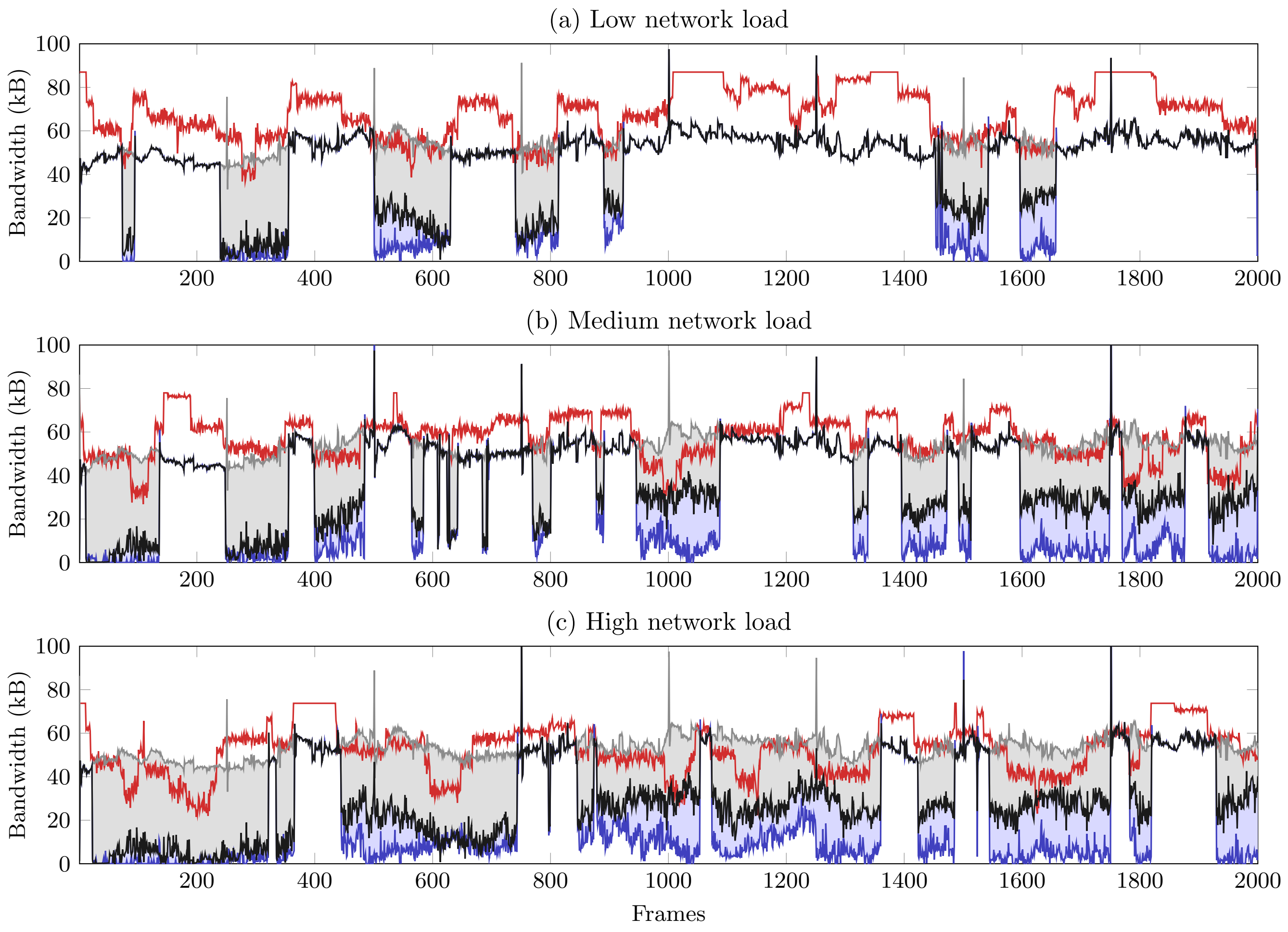

Figure 5. Exemplars of bandwidth traces for streaming video adaptively filtered with two classifiers in the scenarios of low (a), medium (b), and high (c) network load causing roughly 25%, 50%, and 75% of frames being filtered, correspondingly. Red lines (—) show the maximum bandwidth per frame available to the sensor, grey lines (—) show the bandwidth necessary to transmit unfiltered stream, black (—) and blue (—) lines show the actual bandwidth used by the sensor while filtering with Classifiers A and B, respectively. The strides where the sensor decided to activate filtering are highlighted in light gray (●). The difference in bandwidth slack between the two classifiers is highlighted in light blue (●).

The last two columns in Table I display an additional measure of bandwidth slack for each strategy and network load, i.e., the ratio of the average bandwidth used and the amount of bandwidth available for each frame. Although this measure is not directly involved in the on-sensor control, it may be useful as a secondary factor, e.g., for choosing classifier parameters in specific systems following this architecture.

Fig. 5 illustrates the dynamics of the channel use by the sensor for a fragment of the stream under three different network loads. In each case, the resulting frame sizes are shown for the two versions of the pedestrian filter using Classifiers A and B. Notice that, since the moments of activation and deactivation of filtering are defined by the quantile estimation, the only differences between classifiers are in bandwidth slack.

To summarize, the results in Table I indicate that an efficient trade-off between meeting bandwidth constraints and energy use is possible and achievable on practice. With less than a half of processing energy required for full-time filtering, it is possible to reduce the number of bandwidth violations manyfold compared to the power-saving strategy of disabled filtering, down to the levels close to those of full-time filtering.

VI. Related Work

Edge and fog computing techniques have been considered in prior work to reduce the load on the central communication and computation infrastructure [2], [7].

Offloading Mobile Applications to Cloud. Several research papers [8], [9] have analyzed the energy consumption of mobile applications to save the phone battery. For example, [10] analyzes the performance of search on mobile phones when they cache user data in advance. Similarly, [11]–[13] analyzed the gains of offloading compute-intensive parts of a mobile application to the cloud for face recognition and graphics rendering tasks draining phone batteries.

Recent papers have also analyzed the architectural framework where routers and computers belonging to end-users are edge-devices [14] and the energy consumption of nano-datacenters [15] is managed via flow-based and time-based energy consumption models for shared and unshared network equipment. However, the proposed offloading schemes do not account for the network constraints of the devices.

Edge Computing for Vision Applications. Vision applications can be especially compute-intensive and seek to gain the most from the edge-computing paradigm. A recent system Gabriel [16] reduces the latency of uploading data to the cloud by partitioning computation tasks between mobile sensors and the cloud for personal mobile devices in augmented-reality applications. Similarly, Odessa [17] supports interactive perception applications by dynamically offloading parts of computation tasks from mobile devices to the cloud. However, the solutions explored so far focus solely on data processing, without analyzing energy-bandwidth trade-off.

Content-Aware Compression. In the Vigil system [18], the authors propose a real-time distributed wireless surveillance system that leverages edge-computing to support real-time tracking and surveillance across smart cities. One of the key features of Vigil is that it runs a content-aware compression algorithm to determine the most valuable frames from cameras within a cluster and eliminates redundant observations of multiple cameras capturing the same objects to minimize communication bandwidth without actually exchanging the redundant frames. The work shows significant gains for content-aware compression approach to video monitoring.

Further, another recent work [19] demonstrates the loss of packets belonging to a video stream due to imperfect coordination in a network scenario where a D2D link underlays to an LTE network affects video monitoring application. Interference may cause artifacts that would significantly impair the performance of detection and tracking objects. Content-aware compression schemes can tackle such interference patterns robustly. However, most of the recent work does not investigate the trade-offs incurred by energy consumption when leveraging such content-aware compression schemes.

VII. Conclusions

In this paper, we considered the problem of adaptive filtering of data streams from wireless sensors, focusing on the dynamics of the sensor-edge system. We proposed a formal view of the balance between the usage of bandwidth and energy, and formulated the on-sensor control algorithm that is able to achieve optimality with the assistance of the edge processor. An adaptive framework for controlling the activation and deactivation of the on-device data filtering was evaluated on a realistic video processing task in a coexistence scenario. Numerical results show that the proposed bandwidth-aware system can reach an efficient trade-off between transmission guarantees and energy savings.

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12, 2012, pp. 13–16.

[3] R. Koenker, *Quantile regression*. Cambridge University Press, 2005.

[4] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[5] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001.

[6] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.

[7] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the Internet of Things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, 2015.

[8] A. Redondi, L. Baroffio, M. Cesana, and M. Tagliasacchi, "Compress-then-analyze vs. analyze-then-compress: Two paradigms for image analysis in visual sensor networks," in *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*. IEEE, 2013, pp. 278–282.

[9] B. Girod, V. Chandrasekhar, D. M. Chen, N.-M. Cheung, R. Grzeszczuk, Y. Reznik, G. Takacs, S. S. Tsai, and R. Vedantham, "Mobile visual search," *IEEE signal processing magazine*, vol. 28, no. 4, pp. 61–76, 2011.

[10] E. Koukoumidis, D. Lymberopoulos, K. Strauss, J. Liu, and D. Burger, "Pocket cloudlets," in *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XVI. New York, NY, USA: ACM, 2011, pp. 171–184. [Online]. Available: http://doi.acm.org/10.1145/1950365.1950387

[11] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 49–62. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814441

[12] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 301–314. [Online]. Available: http://doi.acm.org/10.1145/1966445.1966473

[13] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision," in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '13, 2013, pp. 69–82.

[14] 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*, ser. MobiArch '14. New York, NY, USA: ACM, 2014, pp. 43–48. [Online]. Available: http://doi.acm.org/10.1145/2645892.2645901

[15] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728–1739, May 2016.

[16] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '14, 2014, pp. 68–81.

[17] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 43–56. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000000

[18] T. Zhang, A. Chowdhery, 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*. ACM, 2015, pp. 426–438.

[19] S. Baidya and M. Levorato, "Content-based cognitive interference control for city monitoring applications in the Urban IoT," *IEEE Globecom 2016, Dec. 4-8, Washington DC, USA*, 2016. [Online]. Available: http://arxiv.org/abs/1606.01965