Intelligent Data Filtering in Constrained IoT Systems

Igor Burago, Davide Callegaro, Marco Levorato, Sameer Singh
Department of Computer Science
University of California, Irvine
{iburago, dcallega, levorato, sameer}@uci.edu

***Abstract*—The expansion of complex autonomous sensing and control mechanisms in the Internet-of-Things systems clashes with constraints on computation and wireless communication resources. In this paper, we propose a framework to address this conflict for applications in which resolution using a centralized architecture with a general-purpose compression of observations is not appropriate. Three approaches for distributing observation detection workload between sensing and processing devices are considered for sensor systems within wireless islands. Each of the approaches is formulated for the shared configuration of a sensor-edge system, in which the network structure, observation monitoring problem, and machine learning-based detector implementing it are not modified. For every approach, a high-level strategy for realization of the detector for different assumptions on the relation between its complexity and the system's constraints is considered. In each case, the potential for the constraints' satisfaction is shown to exist and be exploitable via division, approximation, and delegation of the detector's workload to the sensing devices off the edge processor. We present examples of applications that benefit from the proposed approaches.**

I. Introduction

With the growing interest in the Internet of Things (IoT) vision [1], many real-world systems are pushed to gain increasing capacities for self-awareness, autonomy, and adaptation in their operation. Promising applications relying on this premise include next-generation systems for smart cities [2] and intelligent vehicular networks [3], [4]. Realization of the associated monitoring and control tasks in such systems, however, is complicated by their distributed, multicomponent nature and sophisticated information processing. As participating entities of an IoT system form a network of interconnected computing devices, the architecture of its information processing has to be designed accordingly, taking into account the fundamental structural properties specific to IoT applications:

- The capacities for data acquisition (*sensing*) and data processing (*computing*) are distributed unevenly and vary across the system, with devices of different purpose having different limitations. For instance, mobile devices are usually better suited for sensing and are less capable of processing due to constraints on power expenditure, while less power-restricted stationary devices commonly afford more processing than sensing.
- The system's data sources (*sensors*), processing units (*processors*), and centers of decision making (*controllers*) which rely on the information from acquired data for coordination, typically cannot all coexist on the same devices. For sensors have to be deployed wherever the required features of the environment are to be observed, often on the system's periphery, while processors and especially controllers have to be located higher in the connectivity hierarchy where they can have a bigger scope conjoining several information streams. This necessarily diverges the system's sensing and control functionality toward devices that are distanced from each other, perhaps with some intermediary processing units in between.
- All of the devices communicate over channels of different latency and medium. Sensors typically have a shared low-bandwidth, low-latency connection to the rest of the network that is carried by a common wireless resource and mediated by a local access point at the wireless edge. Processors and controllers are located at or beyond the edge and are connected in a mostly hierarchical fashion with high-bandwidth wireless or wireline links of significantly higher latency than the one-hop links within sensor wireless islands.

The cornerstone trait of an advanced IoT system is the ability to gain timely understanding of its environment, which strongly depends on the efficiency of data processing as it progresses from the acquisition at sensors to the final decision at the central controller. Critically, in many practically interesting IoT systems, the nature of the monitoring or control task set before a system demands a high volume of sensor observations and a high degree of sophistication in their processing. Oftentimes, it involves a machine learning algorithm solving a pattern recognition problem (most commonly, in the form of an environment state classifier or object detector). If the induced processing were to happen within a single computing node having on-the-spot access to all the sensor data, such algorithm could be straightforwardly implemented on a sufficiently powerful controller.

Yet, in practice, single-node processing is almost always ruled out by a combination of restrictions on both computation and communication that follow from the three general properties listed above. Because of those restrictions, which usually

This work was partially supported by the NSF under grants IIS-1724331 and CNS-1702950.

are actualized as time-varying and device-specific constraints on available transmission bandwidth and computational energy, the processing workload has to be reorganized so that it could be adaptively divided among the nodes in the network. In an ideal case, it would be desirable to achieve the following guiding objectives:

- Minimize the amount of traffic to avoid congesting shared wireless resources and violating requirements on processing latency.
- Minimize the induced processing load on the sensors and intermediate processing nodes to avoid depletion of their computational resources.
- Minimize the loss of information in the intermediate stages of processing aimed at traffic reduction, to avoid deterioration of the system's decision making quality.

However, it is easy to see that these objectives are in conflict with each other, as, in general, it is impossible to simultaneously keep all three of the involved qualities low. Indeed, achieving traffic reduction requires adding computational work to sensors and processors, which would not be necessary for a centralized processing leaning on a single controller. Similarly, reduction in both additional computation work and potential information loss causes an increase in the amount of necessary communication. Therefore, efficient allocation of the processing workload in a sufficiently constrained IoT system requires searching for an optimal tradeoff between these objectives.

In this paper, we discuss three different strategies for approaching the problem of finding that balance under different assumptions about the processing to be implemented. In Section II, we first describe those approaches from the general principles and then illustrate them in Section III with three specific problem settings inspired by concrete applications.

II. Sensor-Edge Workload Management: Challenge

In the presence of constraints, the aforementioned difficulties in data processing pose a challenge even in simply organized IoT systems. For the sake of presentation, let us concentrate on one common architecture of (a part of) an IoT system, where the key roles in shaping the information flow are played by two logical entities: an array of *sensors* and an *edge processor*. In it, the sensors, connected into one or more wireless islands, are in charge of data acquisition, and the edge processor, communicating with the sensors via a local access point(s), is in charge of an integral processing of the sensor-acquired data. We assume that the sensors continuously obtain new observations at a discretized time scale with some fixed frequency and are able to transmit them to the edge within the same timeframe.

The sensors-edge system as a whole is assigned with an ongoing computational task, whose output is collected from the edge processor either as a final result or as an input for a higher-level decision making process. Often, this computation task consists in detecting occurrences of some events or conditions of interest in the sensor observations, e.g., identification of suspicious activities in a public space or dangerous states of a patient in the post-operative period. Let us consider the simplest yet significant scenario where the detection task (*detector*) is binary, that is, an observed state is either of interest to the system, and the sensors' data are, then, to be compiled and reported by the edge to a higher-level control, or not, and those observations are to be ignored.

It is this property of impermanent usefulness of sensor-acquired observational data that can be used for overcoming the IoT-related resource limitations. For the considered architecture, those can be aggregated into two critical groups:

- *Bandwidth constraints* for the sensor-edge communication, induced by the wireless channel shared between the sensors as well as other concurrent clients within the wireless island.
- *Computational constraints* for the on-sensor data processing induced by hardware limitations or power deficiency due to the mobile-oriented nature of the sensors' design. (Since our main focus here lies in the data flow originating from the sensors, the edge is assumed to be sufficiently powerful computationally to be effectively unrestricted, compared to the sensors.)

If all observations are of interest to the system and thus have to be communicated to the edge, no improvement can be made to the structure of data processing to prevent exceeding the bandwidth constraints. For this reason, in the applications with those constraints being (at least sporadically) stringent, fully centralized on-edge processing may become simply unrealizable. However, since only some of the information from the sensors' observations are ultimately of interest to the edge, bandwidth constraints can often be satisfied even in those conditions by making the sensors conscious of their input data and empowering them with an ability to transmit the captured information selectively. This, of course, comes at the price of an additional on-sensor data processing, the amount of which is inevitably capped by the computational constraints (sometimes quite severely). For this reason, full sensor self-sufficiency — the complementing extreme to the fully centralized processing — often cannot be realized, either.

For some fortuitous combinations of the detection task and bandwidth constraints, savings resulting from using a general-purpose compression scheme on the sensors' observations may be sufficient to make fully centralized processing feasible. Even then, the effectiveness of such compression, oblivious to the nature of the following processing, is rather limited, though. Bandwidth savings from lossless compression of raw observations are often insufficient, and lossy compression in many cases cannot be made to extend the gain by much either, as increasing its aggressiveness leads to information loss and feature degradation in the data fed to the on-edge detection pipeline, negatively affecting the quality of its outcome.

Thus, even in this simple architecture, we are faced with the problem of finding a nontrivial arrangement of processing between the sensors and the edge, in order to achieve a satisfactory balance between the two extreme strategies without violating the constraints. The bandwidth constraints effectively

bound the capacity of the channel available to a sensor for transmitting its observations to the edge. The computational constraints, on the other hand, bound the sensor's capacity to sift out the omissible portion of the data that is unnecessary for or is rejected by the detector at the edge processor. As a result, the absence of local processing at a sensor is fraught with data loss due to congestion and depletion of the wireless resource, while the full-fledged processing is fraught with exceedingly high (or even impossible) computational expense. This realization naturally leads us to the necessity of hybrid processing approaches that divide the detection workload into the complementing on-sensor and on-edge portions.

In this paper, we intentionally disregard a possible complication of direct peer-to-peer communications between sensors bypassing the edge. Additionally, for the purposes of exposition, herein we consider each sensor independently from others as a part of a single sensor-edge pair tasked with a binary detection problem (as presented above). In the application space, it corresponds either to the case when each sensor's observations are processed in the frame of an independent detection problem, or to the case when the detector for each sensor-edge pair is to filter out only the observations that are useful for the final joint detection problem at the edge.

Let us now discuss three high-level approaches to extracting the said on-sensor processing and its supporting on-edge counterpart. All of them are unified by the common idea of resolving the conflict between the desired performance of the system's detector and the communication constraints by distributing the induced processing workload between both ends of the sensor-edge system. Conceptually, the three approaches are based on the concepts of delegation, division, and extension of the workload. The main question in relation to which the approaches can be principally differentiated is:

> *Given a pair of a wirelessly connected sensor and an edge processor, together given the task of detecting a certain kind of observations, how can the associated detection workload be managed between them, under the constraints on the communication channel capacity and the on-sensor computational expense?* (∗)

The presented approaches do not cover the space of viable strategies exhaustively, but rather represent those that we find promising for further development, listed here in the order of increasing organizational complexity. In Section III, each of them is further discussed in application to a more specific problem setting.

A. Alternating Detection Workload Offloading

The space of alternatives for satisfying the bandwidth constraints is heavily influenced by the class of the sensor's computation capacity. Clearly, the biggest savings are achieved when the workload of the detection problem is fully moved from the edge processor to the sensor. Applications in which the sensor is capable to support that, though, rarely occur in practice. However, in some cases the detector used by the edge for the sensor's data is not that computationally demanding, so the sensor is able to run it intermittently

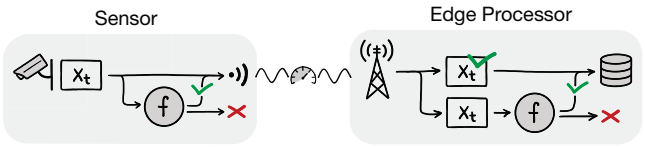

Figure 1. Workflow of the sensor-edge system implementing the *alternating detection workload offloading* strategy. Here x_t denotes a single observation at any given time slot t, and f stands for the detector algorithm shared by the sensor and the edge. Green checkmark represents arrival at the decision to accept an observation as a detection, while red cross signifies the rejection.

for some of the observations. In other words, the detection algorithm in question is computationally viable for the sensor's hardware, but for reasons of maintaining a low thermal profile or reducing power consumption, it is necessary to keep the average number of its invocations small.

This circumstance immediately suggests the following high-level strategy in response to the main question (∗).

- Lower the sensor's bandwidth consumption by selectively applying the detector to a subset of the observations locally and suppressing transmission of those that are rejected by it. Transmit the rest of them as usual.
- Keep the computational impact on the sensor contained by limiting the running average of the number of observations the detector is locally applied to.

This way, the sensor still communicates with the edge on the same granularity level of separate observations, just with some of them being omitted from the transmitted data stream. In order to avoid work duplication, the sensor accompanies each communicated observation with a flag indicating whether the detector has been run against it pre-transmission. Obviously, the decision to bypass the detector must be made locally at the sensor, but may be supported by the information supplied with the edge's assistance.

At the edge, the processing schedule remains almost the same: whenever the edge processor receives a new observation from the sensor, it either first applies the detector to it as before, if the processing has not been done already on the sensor, or immediately passes it on as a newfound detection, otherwise. Since it is the very same detector that is used by the sensor, all of the omitted observations would have been rejected by the edge anyway, had they been sent to it, and thus do not create any misses (i.e., false negatives) for the sensor-edge system overall. Obviously, it does not introduce new erroneous hits (i.e., false positives) either, as the detector is still being run at the edge for all unprocessed observations.

Figure 1 schematically shows the resulting workflow of the sensor-edge system. It effectively realizes a dynamic offloading scheme in which the burden of the detection work for each observation is dynamically shifted from the edge to the sensor and vice versa. Such flexibility adds to the system a new degree of freedom for adapting to the time-varying capacity of its wireless resource. At the same time, it creates the necessity to manage the ongoing computational impact of the on-sensor offloading of the data processing. Whether the

former factor can outweigh the latter depends on the specific amounts of bandwidth savings and computational expenses in each application. Nevertheless, for permitting configurations, it becomes possible for the system to operate under a more efficient regimes that are unattainable otherwise within the centralized processing architecture.

B. Consistent Detection Workload Division

Unfortunately, the key assumption of the approach from Section II-A makes it inapplicable to many systems. Indeed, if the computational cost of running the edge-hosted detector on the sensor exceeds the threshold of practicality even for a single observation, it becomes impossible to divide the detection workload between the sensor and the edge processor on the observation-by-observation basis anymore (without switching to a less-capable detector algorithm).

In such cases, an opportunity for splitting the work may still be available in the internal organization of the detector, though. Consider a detector that processes an observation in a pipeline-like manner and, therefore, can be broken down into two or more stages. Then, instead of offloading to the sensor the chunks of work corresponding to the full processing of *selected* observations, we can offload the chunks corresponding to different degrees of preprocessing of *every* observation. Additionally, if the amount of information at the input of subsequent stages tends to *decrease*, the structure of such detector can be exploited to address the main question (∗) of the computation-communication balance as follows.

- Cut the bandwidth used by the sensor down by preprocessing each observation with the first few stages of the detector's pipeline and transmitting the intermediary result at the output of those stages instead of the full description of the original observation.
- Control the associated computational cost at the sensor by (dynamically) limiting the number of stages included into the on-sensor portion of the detector used for preprocessing.

As it can be clearly seen on the schematic diagram of this strategy in Figure 2, the sensor and the edge become very similar in terms processing of their respective inputs, as both of them implement structurally similar pipelines, which differ only in the interpretation of their output. As before, the quality of the processing remains unchanged; it is only the distribution of processing workload that is affected.

Under this strategy, unlike the one from Section II-A, the sensor always transmits some information about every observation, so the necessary reduction in bandwidth usage is achieved by taking advantage of the detector's structure instead. Even when the processing of the detection pipeline is only generally reductive, if there is at least one *breaking point* in it such that the output of all the stages up to that point can be encoded more efficiently than the corresponding input observation, this first part of the detector effectively realizes a *semantic compressor* specific to the nature of the particular detection logic.

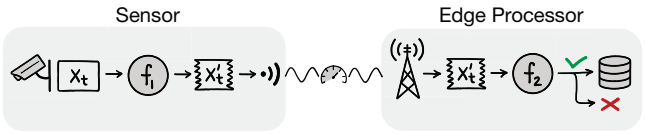

Figure 2. Workflow of the sensor-edge system implementing the *consistent detection workload division* strategy. Here x_t and $x'_t = f_1(x_t)$ denote an input observation and its partially processed reduced version, respectively; f_1 and f_2 stand for the on-sensor and on-edge portions of the detector algorithm $f(x_t) = f_2(f_1(x_t))$ implemented by the system. Green checkmark represents arrival at the decision to accept an observation as a detection, while red cross signifies the rejection.

Although in theory the presence of a breaking point in the detector alone guarantees the potential for bandwidth gains, its position in the processing pipeline is also quite important in practice. If all breaking points are condensed into the very last stages of the pipeline, it may be infeasible to reap the fruits of those gains due to the computational constraints, as the aggregate workload of the stages preceding even the earliest of those breaking points (which is to be offloaded to the sensor) might not be sufficiently less costly than the complete workload of the unbroken detector.

Thus, for the combinations of detectors and constraints that permit extraction of such compressing sub-detector, the sensor-edge system gains another tool for managing the computation-communication balance. Moreover, in the cases where the detector allows for multiple practical breaking points, this new degree of freedom is non-binary, so that the amount of offloaded on-sensor workload may be controlled by the edge with higher granularity than it is possible in the approach from Section II-A.

C. Approximate Detection Workload Extension

There are detection problems in which the prior two approaches are impracticable still. That is, the detector algorithm that has to be used by the system is too heavy for the sensor and, at the same time, is structured in a way that either prevents seamless splitting of the workload (voiding the option to offload a part of it) or fails to grant any gains in the sensor's channel use (taking away the whole point of such offloading), or both. Hence, the detector, as is, no longer can provide a way to balance the computation-communication tradeoff. Of course, it can be replaced with a less capable detector suitable to the above approaches, but at the undesirable price of an inevitable downgrade in the system's performance.

However, in such instance, an additional resource for bandwidth savings may be found in the distribution of the sensor's inputs. Many applications involve observations that carry significant temporal correlations, so that successive readings have the tendency to be in each other's vicinity in the observation space. Furthermore, it is not uncommon for the locality encompassing a chain of subsequent non-stationary observations to be small enough, so that the separating surface of the detector is significantly less complex in it. When it is so, in that locality, the detector can be temporarily approximated with a simpler version of itself that may be viable for on-sensor execution. If

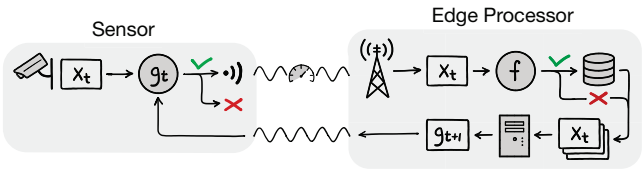

Figure 3. Workflow of the sensor-edge system implementing the *approximate detection workload extension* strategy. Here x_t denotes a single observation, and f and g_t stand for the full-fledged on-edge detector and its complementing simplified on-sensor detector, respectively. Green checkmark represents arrival at the decision to accept an observation as a detection (tentatively, when using g_t), while red cross signifies the rejection.

this condition of *locality preservation* does indeed hold more often than not along the observed trajectory in the space of the sensor's readings, the main question $(*)$ may be brought to resolution as follows.

- Reduce the sensor's bandwidth consumption by applying a simplified detector to each incoming observation at the sensor, and leaving out of transmission those that are rejected by this complementing on-sensor detector.
- Bound the computational load of the on-sensor detector by restricting its choice to a simpler family of classifiers that satisfy the device's constraints.

In order to prevent any noticeable loss in the system's detection performance, the simpler on-sensor detector should be trained to minimize its false negative rate (with an optional constraint on its false positive rate). This way, the sensor is effectively supplied by the edge with a detector of some (hopefully, as many as possible) uninteresting observations that could be identified with a less complex algorithm fitting the sensor's processing constraints.

From the standpoint of the sensor, there is little difference in the processing protocol of this approach versus the first one we have described in Section II-A. Since it is only the edge processor that is capable of running the full-fledged detector, the accompanying detector for the on-sensor use has to be maintained by the edge based on the observation statistics arriving from the sensor and periodically communicated back to it (see the workflow diagram in Figure 3). The sensor, then, may be even unaware that the detector supplied by the edge is not original, as it does not affect its operation in any way.

As in Section II-A, the processing of each observation passing the on-sensor filter and being communicated to the edge, is completed there by applying the full-fledged detector to make the final distinction between a false positive of the simplified on-sensor detector and a true detection to be reported as such to the outside. In a similar fashion, if the computational price of the on-sensor detector is too high to be paid for every observation, it may be used only for a portion of them, without requiring a change in the on-edge processing.

This approach, certainly, adds a significant complication by requiring timely estimation of the current locality encompassing the sensor's observations, without having all of them available at edge (as it would violate the communication constraints). Generally, two strategies can be employed here depending on the application. One strategy is to charge the sensor with the task of supplying the edge with a compact description of the probability distribution fitted to its recent observations. Having this distribution, the edge is able to solve the stochastic optimization problem for the parameters of the sought on-sensor detector by minimizing a dissimilarity measure of its output with the output of the master on-edge detector over a sample distributed according the reported distribution.

The other strategy is to keep the sensor's logic unchanged, so that the only information reported from it remains to be the raw observations that pass its current local detector. The same optimization problem is then continuously solved by the edge, but on the sample of precisely those received observations. This policy is usually preferable to the previous one as it allows to relieve the sensor of any additional work beyond the pre-transmission filtering. However, it requires more intelligence from the edge in updating the on-sensor detector to compensate for the fact that the sample of observations available to the edge may be skewed if the on-sensor detector has not caught up with a recent locality change. In some cases, this might, in turn, prevent it from catching up even further, as the necessary observations may keep being rejected by the ever more obsolete on-sensor detector in the name of sparing the bandwidth usage. For this reason, the edge has to err on the side of making the on-sensor detector (periodically) more permissive to allow more false positives, in order to reduce the asymmetry of the reported sample and to maintain a high confidence in its relevance.

III. Sensor-Edge Workload Management: Problems

A. Edge-Assisted Bandwidth-Aware Observation Filtering Problem

As an illustration for the approach introduced in Section II-A, let us consider an instance of a system using an on-sensor edge-assisted filtering of object detections as a way to cope with the limited wireless bandwidth in a video monitoring application. The sensor-edge system in question is designed to observe a real-time video stream capturing a view of a city street for detecting and extracting the images of all pedestrians that happen to be caught on camera. This is an example of a typical safety-related monitoring task that may arise in a smart city system. To collect the pedestrian images, the edge processor is equipped with a pre-trained high-speed Haar feature-based cascade classifier.

The role of the sensor is played by a smart camera, which, at a fixed frequency, captures the frames of the input video stream. Due to the low computational profile of the detector, it is viable for the camera to run this detector locally on a separate video frame. However, in order to minimize the energy consumption, it is desirable to use it there only as often as it is necessary to satisfy the bandwidth constraints of the shared wireless channel. It is easy to see that the circumstances of this application are well-suited for the requirements of the full observation-based on-sensor detector offloading. Adhering

to the strategy from Section II-A, we can construct the following frame-processing system.

First, the smart camera sensor captures a video frame, and then makes a decision on whether to scan it with the Haar feature-based pedestrian detector, depending on the bandwidth pressure put on the sensor by the edge in the current time period t. If the detector is applied to the frame, the pixels that are outside of the detected pedestrian regions can be safely dropped from transmission to spare bandwidth usage, as only those regions are to be collected for further processing at the edge processor. Otherwise, the whole frame image is sent as is. In either case, the usual image compression is applied.

At the edge, then, the arriving frames that have not been filtered with the detector at the sensor are scanned with it. The pedestrian images found in the frame, regardless of whether they were extracted at the sensor or at the edge, are reported at the system's output. As a feedback, the edge reports back to the sensor two characteristics to assist the sensor's decision on running the detector: (a) the amount of bandwidth b_{t+1} guaranteed to be available for the sensor in the next time slot $t+1$, and (b) an estimated interval D_{t+1} of the *object density* in the next frame, based on the few recent ones. For the purposes of this system, the object density of a frame is defined as the portion of the frame area that is taken by the pedestrian detection regions. In general, one or more other application-specific features may be used in addition to or instead of the object density, making D_{t+1} a multidimensional feature region.

In this setting, we set the on-sensor adaptation problem in the following form: Find an optimal decision policy minimizing the number of frames which are to be processed locally, while keeping the probability of frame-by-frame bandwidth violations below some threshold $1-\alpha$. More precisely, the objective of the on-sensor detector activation control is to find optimal parameters

$$\theta^* = \arg\min_{\theta} \mathbb{E}[\ell(b_t - \varphi_t(\delta_t, \theta)) \mid \delta_t \in D_t], \tag{1}$$

for an estimation $\varphi_t(\delta_t, \theta)$ of the transmission size of an unprocessed, unfiltered video frame x_t whose object density is $\delta_t(x_t)$, where $b_t(x_t)$ is the true transmission size of x_t, and $\ell(u)$ is the loss function determining the cost of overestimating (for $u < 0$) or underestimating (for $u > 0$) the transmission size of a frame. The expectation is taken over the video frames x_t whose object densities belong to the edge suggested density interval D_t.

Although different optimal policies can be found depending on the preferred definition of the loss function, for one particular choice of it, the optimization problem (1) can be solved analytically. Namely, for the loss function of the form

$$\ell(u) = u \cdot (\alpha - \mathbb{1}[u < 0]), \tag{2}$$

the problem (1) can be rewritten in its empirical version as:

$$\theta^* = \arg\min_{\theta} \Big[\alpha \sum_{\substack{\delta_t \in D_t \\ b_t > \varphi_t}} |b_t - \varphi_t| + (1-\alpha) \sum_{\substack{\delta_t \in D_t \\ b_t < \varphi_t}} |b_t - \varphi_t|\Big], \tag{3}$$

where the factors α and $1-\alpha$ weigh the costs of overestimating the frame size (i.e., wasting the energy on processing when it could have been spared) and underestimating it (i.e., risking a frame loss due to attempting to transmit unprocessed frame that is too big). In other words, α takes the meaning of the expected number of frames whose transmission should not fail due to the sensor sending them unprocessed when the available bandwidth is insufficient for that.

It can be easily seen that the optimal estimator for the loss function (2) is

$$\varphi_t(\delta_t, \theta^*) = \theta^*_{D_t}, \tag{4}$$

for $\theta^*_{D_t}$ being the αth conditional quantile for the frames x_t whose object densities $\delta_t(x_t)$ belong to D_t. This quality of the resulting policy is quite desirable for a real-time application, as it allows to avoid any optimizing work during the system's operation.

In more detail, this problem has been described in our paper in the Proceedings of the 2017 IEEE Conference on Sensing, Communication, and Networking (SECON) [5].

B. Deep Neural Network Splitting and Observation Compression Problem

A natural fit for the workload division approach proposed in Section II-B may be found in applications where the detector is implemented with a Deep Neural Network (DNN) classifier. By construction, a multilayer neural network can be seen as a pipeline of sorts, different stages of which correspond to portions of the processing done by separate layers of the network, with clear information flow between them in the form of input and output links connecting neighboring layers.

For the approach to be useful, the pipeline to be split into the on-sensor and on-edge sub-pipelines has to have the key property: the tendency for the reduction of the amount of information passed from each subsequent stage to the next. In practice, this presents a challenge for implementation of this strategy within many DNN architectures. On the one hand, such network built for a binary classification problem must exhibit a semantically reducing data flow, so that, at least in one of the last few layers of the network, the size of the input does not exceed the size of the whole observation itself. On the other hand, depending on the specific composition of layer types, the amount of data passed between successive layers may not change monotonically, leaving us with a limited set of breaking points for splitting the detector's workload.

As we mentioned in Section II-B, we are most interested in breaking points that realize a good tradeoff between the computational cost of the layers preceding it in the network and the size of the output of their last layer, which is passed to the the first layer beyond that point. For the networks that have well-balanced breaking points, the approach can be applied directly to choose the earliest splitting place satisfying the bandwidth constraints. Although not every DNN detector is guaranteed to have that property, there exist neural network classifiers having intermediate layers early enough in their processing pipeline whose output is smaller than the network's

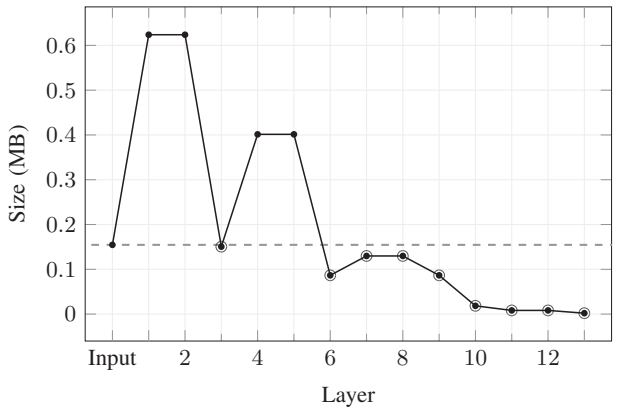

Figure 4. Sizes of the uncompressed output of the layers in the AlexNet convolutional neural network trained for the ImageNet visual recognition dataset. The dashed line is set at the size of the network's input. Potential breaking points for the layers whose output takes less space than that are circled.

input. For instance, Figure 4 shows that the amount of data required to transmit the uncompressed output of layers 3 and 6–13 in the AlexNet convolutional neural network is smaller than the amount of data necessary to transmit an input observation. However, for a network that does not have a practical set of breaking points, it is not necessarily impossible to break its workload into suitable subnetworks. In those cases, one of the two following complications may be employable.

Due to a high degree of cross-layer connectivity, the network may not have a break point satisfying the bandwidth constraints in terms of the layer's output size early enough in the processing pipeline (i.e., in the first few layers) for the computational constraints to hold. Yet, it may be possible that, although the number of bits required to communicate the output of the candidate layers for splitting is not significantly lower than the size of the raw input observation, the entropy of those output vectors for is lower than the entropy of those observations themselves.

Normally, in the straightforward approach, an observation is passed through the first few offloaded layers at the sensor, and then the resulting output is transmitted to the edge, where it is supplied as an input to the first of the rest of the layers giving the final classification decision. In this case, the network can be broken down into two — the on-sensor and on-edge one — right after one of those layers having a lower output entropy, with an additional *compressor* and *decompressor* added at the place of the split. Therefore, in addition to the usual operation, the sensor is to compress the output of its part of the neural network before transmission, and the edge is to decompress the received data before passing it to its part of the neural network for completion of the observation processing.

In situations where there are no breaking points available in a DNN classifier that do not violate either the bandwidth or computational constraints, we may consider modifying the given network in order to introduce one or more breaking points in it. One way to achieve that is to use the same examples that have been used for supervision during the training of the original DNN detector in order to train a new DNN of the same structure and having the same parameters, but with an additional regularization term added to the objective of the gradient descent that would penalize nonzero weights at the output of a certain fixed layer. If such retraining is possible (establishing it constitutes an important research question for each particular application) so that the accuracy of the new network is not dramatically lower than that of the reference one, the new network can be used instead, assuming that enough weights have been zeroed out to guarantee a bandwidth-saving breaking point.

Analogous modifications to existing DNN architectures that enable some mechanisms for adaptive workload truncation at the end-devices have been a subject of recent developments in the literature. The proposed techniques explore different ways of restructuring DNN-based observation processing in a cascade-like [6], [7] or other modular [8] or hierarchical [9] manner allowing for early exit with a partial outcome, under the pressure of computational constraints.

Other related techniques approximate the outcome of a DNN with a dynamically selected model out of a catalog of less accurate but less computationally expensive ones [10], or replace a DNN with its sparser approximation fitting reduced computational requirements [11]–[13].

C. Dynamic Locality-Aware Observation Filtering Problem

Following the high-level approach introduced in Section II-C, let us now describe the architecture of the sensor-edge system implementing it in more precise terms.

Let $f\colon X \to \mathbb{R}$ be the master detector, partitioning the space of possible observations X with the decision rule $f(x) \gtrless 0$. This binary classifier f is the full-fledged detector implementing the system's desired recognition behavior, that is assumed to be fully trained and available at the edge processor. For each time period t, the edge has to find a localized on-sensor detector $g_t\colon X \to \mathbb{R}$ from some fixed family of classifiers $\mathcal{G}$ that adapts the on-edge classifier f to the current condition of the system and is determined by the following three aspects.

- *Observation locality* $\mathcal{D}_t$. The sought detector g_t is to achieve desired levels of false-positive and false-negative rates for the inputs distributed similarly to the recent ones observed by the sensor up to the period $t-1$, as described by some distribution $\mathcal{D}_t$.
- *Bandwidth constraint* b_t. The sensor's wireless communication channel constraints are given in the form of the expected channel capacity b_t that is available to the sensor in the period t. It is assumed to be available at the edge as a parameter of the system or be acquirable by it from a wireless channel access model.
- *Energy constraint* e_t. The sensor's computational constraints are expressed in terms of the maximal energy expense e_t that the sensor can use for local data processing in the period t. It is assumed to be immediately accessible at the sensor and be reported to the edge.

(The computational resources of the edge processor are assumed to be unconstrained.)

In other words, the problem of finding an optimal on-sensor detector g_t requires finding a function in $\mathcal{G}$ that is close to f for points distributed according to $\mathcal{D}_t$, but does not violate on average the bandwidth and energy limits of b_t and e_t set by the edge. More formally:

$$g_t^* = \arg\min_{g_t \in \mathcal{G}} \mathbb{E}_{x \sim \mathcal{D}_t}[L(f(x), g_t(x))], \quad (5)$$
$$\text{s.t. } \mathbb{E}_{x \sim \mathcal{D}_t}[R_E(E_{g_t}(x), e_t)] \leq \varepsilon_t,$$
$$\mathbb{E}_{x \sim \mathcal{D}_t}[R_B(B_{g_t}(x), b_t)] \leq \beta_t.$$

Here the objective function has the meaning of the expected loss $L(f(x), g_t(x))$ between the predictions of the candidate and reference detectors g_t and f, respectively, averaged over (a sample from) the distribution $\mathcal{D}_t$. The loss function L can be set to any suitable measure of dissimilarity between the two separating surfaces, e.g., the squared difference loss, the logistic loss, the binary error loss, etc. In the latter case, to prevent missing detections, the function L may be asymmetrical, assigning different costs to the instances of false positive and false negative errors of g_t.

The intuition behind the addition of the expected value operator over $\mathcal{D}_t$ comes from the simple fact that, a less capable, more energy-efficient algorithm, g_t cannot, in general, be a uniformly good fit for f (otherwise f could be replaced with some $g \in \mathcal{G}$ globally). Hence, in each period t, it has to be optimized separately to take advantage of the temporarily reduced data variation in the corresponding fragment of the observation sequence.

Both computation and communication constraints are handled in a similar fashion. The functions $E_{g_t} \colon X \to \mathbb{R}_{\geq 0}$ and $B_{g_t} \colon X \to \mathbb{R}_{\geq 0}$ define the amount of (or an estimation of) energy to be consumed by the sensor for evaluating g_t for an observation $x \in X$ and the amount of bandwidth that is required for its transmission to the edge, given that $g_t(x) > 0$. Functions $R_E(\cdot, e_t)$ and $R_B(\cdot, b_t)$ play the roles of nonnegative penalties capturing the cost of exceeding the energy and bandwidth limits e_t and b_t. By convention, both penalties should be zero for an observation if the constraints are satisfied for it. Finally, the thresholds ε_t and β_t denote tolerances on the expected values of the said penalties over the given distribution of $\mathcal{D}_t$.

The observation locality distribution $\mathcal{D}_t$ is introduced into this problem statement explicitly, because it plays an important role in communication between the sensor and the edge. As we have mentioned in Section II-C, the information about $\mathcal{D}_t$ may be communicated in two different ways in a real system. One way implicates that, in a parameterized form, the distribution $\mathcal{D}_t$ becomes the sensor's request message for receiving an updated local detector g_t. In that case, the sensor and the edge must agree on a compactly describable distribution beforehand, so that sharing its description with the edge requires sending less information than would be necessary for the underlying sample of observations.

In the other way, $\mathcal{D}_t$ plays only a formal role in the problem (5), as the corresponding expected values are simply computed as averages over the observations reaching the edge processor. This choice necessitates a more complex loss function L that would dynamically adjust for an extra slack in g_t to add a margin of false positives to the separating surface $g_t(x) = 0$. Otherwise, if subsequent observations $x_{t+1}, x_{t+2}, \ldots$ are shifting toward the subspace $g_t(x) < 0$, the points from which are not communicated to the edge, the sensor risks being stuck with a stale detector g_t that would cause many detection misses, plummeting the system's detection performance.

For a more detailed exposition of this problem setting, refer to the conference proceedings of the 2017 IEEE Information Theory and Applications Workshop (ITA) [14].

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] P. Neirotti, A. D. Marco, A. Cagliano, G. Mangano, and F. Scorrano, "Current trends in smart city initiatives: Some stylised facts," *Cities*, vol. 38, pp. 25–36, 2014.

[3] C. T. Barba, M. A. Mateos, P. R. Soto, A. M. Mezher, and M. A. Igartua, "Smart city for vanets using warning messages, traffic statistics and intelligent traffic lights," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 902–907.

[4] F. J. Martinez, C.-K. Toh, J.-C. Cano, C. T. Calafate, and P. Manzoni, "Emergency services in future intelligent transportation systems based on vehicular communication networks," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 2, pp. 6–20, 2010.

[5] I. Burago, M. Levorato, and A. Chowdhery, "Bandwidth-aware data filtering in edge-assisted wireless sensor systems," in *Proceedings 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2017.

[6] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5325–5334.

[7] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2017, pp. 615–629.

[8] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, 2016, pp. 2464–2469.

[9] ——, "Distributed deep neural networks over the cloud, the edge and end devices," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 328–339.

[10] 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*. ACM, 2016, pp. 123–136.

[11] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[12] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[13] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *arXiv preprint arXiv:1511.06530*, 2015.

[14] I. Burago, M. Levorato, and S. Singh, "Semantic compression for edge-assisted systems," in *Information Theory and Applications Workshop (ITA)*. IEEE, 2017.