Understanding the Potential of Server-Driven Edge Video Analytics

Qizheng Zhang
qizhengz@uchicago.edu
University of Chicago

Kuntai Du
kuntai@uchicago.edu
University of Chicago

Neil Agarwal
neilagarwal@cs.princeton.edu
Princeton University

Ravi Netravali
rnetravali@cs.princeton.edu
Princeton University

Junchen Jiang
junchenj@uchicago.edu
University of Chicago

ABSTRACT

The proliferation of edge video analytics applications has given rise to a new breed of streaming protocols which stream aggressively compressed videos to remote servers for compute-intensive DNN inference. One popular design paradigm of such protocols is to leverage the server-side DNN to extract useful feedback (*e.g.,* based on a low-quality-encoded stream sent to the server) and use the feedback to inform how the camera should encode and stream the video in the future. In this *server-driven approach*, an ideal feedback should (1) be derived from minimum information from the video sensor (2) incur minimum bandwidth usage to obtain (3) indicate the optimal video streaming/encoding scheme (*e.g.,* the minimum frames/regions that require high encoding quality). However, our preliminary study shows that these idealized requirements are far from being met. Using object detection as an example use case, we demonstrate significant yet untapped room for improvement by considering a broader design space, in terms of how the feedback should be derived from the DNN, how often it should be extracted, and how to determine the encoding quality of the video on which we extract the feedback.

CCS CONCEPTS

• **Networks** → **Application layer protocols**; • **Information systems** → **Data analytics**; • **Computing methodologies** → **Computer vision problems**.

KEYWORDS

edge video analytics, deep neural networks, server-driven, saliency

ACM Reference Format:
Qizheng Zhang, Kuntai Du, Neil Agarwal, Ravi Netravali, and Junchen Jiang. 2022. Understanding the Potential of Server-Driven Edge Video Analytics. In *The 23rd International Workshop on Mobile Computing Systems and Applications (HotMobile '22), March 9–10, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3508396.3512872

1 INTRODUCTION

Video sensors are ubiquitous, in smart cities, homes, industrial settings, leading to an explosive growth of live video streams from which valuable information can be extracted by computer-vision models, in forms of Deep Neural Networks (DNNs). Unlike streaming videos for human users to watch, when video sensors send live video feeds to a server for compute-intensive DNN inference in real-time, the goal is to maximize inference accuracy under the presented camera-to-server bandwidth constraints [10, 17, 27]. This is increasingly critical, as most video sensors are wirelessly connected with limited bandwidth and/or pricey data plan.

To this end, many video analytics systems have been recently proposed [6, 7, 9, 10, 12, 13, 15–17, 19, 24, 27–29], and one of the common approaches is the *server-driven* paradigm (*e.g.,* [10, 13, 15, 17, 19, 27, 29]). At a high level, the server-driven paradigm has two phases (Figure 1): the camera first sends video frames with regions that do not contain detected objects in low quality to the server-side DNN; based on the DNN output on these low-quality frames, the server sends the client a feedback that indicates which regions or frames are important (*e.g.,* might contain new objects of interests); upon receiving this feedback, the camera will resend the current frames or send future frames with only these important regions encoded in high quality. The rationale of the server-driven paradigm is that edge cameras, which are incapable of running expensive DNN inference, cannot accurately determine which regions contribute more to the inference accuracy of the compute-intensive server-side DNN. In contrast, the server-driven paradigm uniquely allows the server-side DNN to directly determine which regions should be encoded and streamed in high quality.

Given the widespread adoption of systems in the server-driven paradigm [10, 13, 15, 17, 19, 27, 29] and the numerous tuning options in these systems (*e.g.,* quality selections, frame rates, feedback types), we first study the performance of existing server-driven video streaming systems for video analytics on a variety of videos using object detection as an example application. Overall, we find that despite the early promises of the server-driven paradigm in certain settings as displayed by prior work in this space, existing pipelines leave many opportunities for performance improvement on the table. In particular, we identify that current instantiations of the server-driven paradigm fail to extract server-side feedback that directly and accurately models which regions are more important to inference accuracy. This is mainly because these systems rely exclusively on *bounding boxes* from the outputs or *region proposals* on the frames to extract server-side feedback and determine which regions should be encoded in high quality. Our study shows that such feedback extraction fails to capture (1) that many pixels outside the bounding boxes can still influence DNN inference, and (2) that many pixels inside the bounding boxes are not as influential.

To resolve this limitation, we need to better model the relative contribution of a pixel to the inference accuracy than existing bounding-box-based heuristics. To meet this goal, we leverage the well-studied concept of *saliency* in the computer vision community [8, 22, 23], computed as the gradient of the sum of bounding box confidence scores with regard to each pixel RGB value. In contrast to bounding boxes or region proposals, saliency, by definition, captures how much

HotMobile '22, March 9–10, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9218-1/22/03…$15.00
https://doi.org/10.1145/3508396.3512872

Figure 1: A typical server-driven video analytics workflow. LQ refers to low encoding quality and HQ refers to high encoding quality.

changing each pixel value can affect the output of the server-side DNN.

To show the potential improvement (in accuracy and bandwidth savings) of using saliency as the feedback, we first make an idealized assumption that the camera can access the saliency of the *uncompressed* frames, which means how much a slight change on each pixel RGB value can have on the DNN output (§3.3). Based on the saliency information, the camera then encodes 10% (16x16) macroblocks with highest saliency sum in high quality and the remaining regions in low quality. Compared to existing server-driven designs (DDS [10], Liu *et. al* [17], etc.), we found that this idealized design reduces bandwidth usage by 18% and confidence drop by 76% compared to DDS [10], and reduces bandwidth usage by 40% and confidence drop by 56% compared to Liu *et. al* [17] on the *DDS* videos (Figure 3).

Yet, realizing the potential improvements in practice can be hard, since it requires sending the high-quality video frames to the server in the first place. Surprisingly, we found that the potential improvements can be largely attained by selecting an appropriate encoding quality and frame rate of the low-quality frames that the server-side DNN uses to extract saliency (§4.2). In contrast, previous work in this space manually picks these values, without exploring the full space of possible values. Our empirical study, over a variety of videos, shows that a realistic server-driven design can still achieve 43% bandwidth saving and 12% less accuracy drop as compared to DDS [10], or 57% bandwidth saving without degrading DNN inference accuracy as compared to Liu *et. al* [17].

In short, by changing key design choices (the format of feedback, and the video quality and frame rate at which the feedback should be extracted) of server-driven video analytics systems, we demonstrate significant yet untapped improvement over the latest server-driven designs with a simple yet efficient alternative. In doing so and outlining unanswered questions (§5 and §6), we hope to encourage more research in the larger community towards the optimal design of server-driven video analytics systems.

2 SERVER-DRIVEN VIDEO ANALYTICS

Edge video analytics seeks to run accurate but compute-intensive DNN inference on massive video feeds from cheap cameras. To leverage the abundant server-side compute to optimize the trade-off between inference accuracy and various types of cost (like bandwidth consumption [10, 17, 29], server-side compute cost [29] and end-to-end latency [17, 29]), many proposals in this space follow the *server-driven* approach. In an ideal server-driven video analytics system, the server-side DNN extracts some *feedback* that indicates the minimum regions in each frame that must be encoded in high quality in order for the DNN to maximize accuracy, and then sends this feedback back to the camera which then encodes and streams the video frames to the server based on the feedback.

	Feedback format	Feedback frequency	Base video quality
DDS [10]	Region proposals (subset)	Every frame	Low (QP=36)
Liu *et. al* [17]	Region proposals	Every frame	Hybrid (QP=35)
Elf [29]	Region proposals	Every 7 frames	Low (Res=760x432)

Table 1: Existing designs of server-driven video analytics systems and their design choices along three dimensions. We will show significant potential improvement by exploring alternative choices along these dimensions. [2]

Performance objectives: An ideal server-driven approach should meet the following two requirements.[1] (1) *High accuracy:* We define accuracy as the average *drop of the confidence* at each object, between the inference output on the highest-quality video and that on the actually streamed video. Though it is not the true inference accuracy, we choose to use this definition because it captures any adverse impact of streaming low quality videos on the inference output: in practice, if the confidence of an object is dropped below the threshold manually picked by the DNN operator, the object will not be returned. (2) *Low bandwidth usage:* We measure the bandwidth by the video file size over its duration.

Current server-driven designs: Existing proposals [10, 13, 15, 17, 19, 27, 29] of server-driven approach make different design choices along three dimensions:

- The *format* of the server-extracted feedback sent from the server to the camera,
- The *frequency* at which the feedback is updated by the server-side DNN, and
- The *quality* of the video on which the server-side DNN extracts the feedback. This is measured either by quantization parameter (QP) in video codecs like H.264 [25] or by resolution. In our experiments, we will use QP.

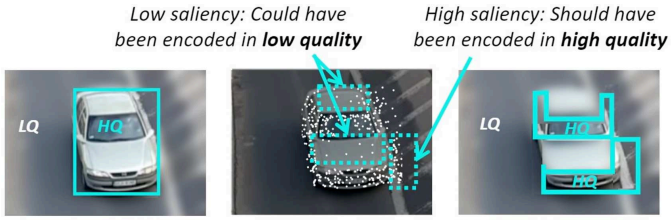
In this paper, we focus on three instantiations of server-driven approach: DDS [10], Liu *et. al* [17] and Elf [29] that have state-of-the-art bandwidth-accuracy trade-off.[3] Table 1 summarizes three representative designs from recent papers. Note that in Liu *et. al* [17] and Elf [29], the server-driven streaming of video frames is only part of the overall system, and to make our discussion more focused, the table only includes their design choices of server-driven streaming.

Performance advantages of server-driven paradigm: These instantiations show promising performance gains over camera-side frame dropping or quality downsizing, on their perspective datasets. DDS [10], for instance, maintains same or higher accuracy while reducing bandwidth usage by upto 59% or improves accuracy by upto 9% with no additional bandwidth usage; Liu *et. al* [17] improves the detection accuracy by 20.2%-34.8% on two applications with

[1] We do not explicitly evaluate server-side compute cost and end-to-end inference delay, but reducing bandwidth usage also helps reduce the end-to-end inference delay, and our optimization does not affect server-side compute cost of server-driven designs.

[2] The QP of DDS refers to the quantization parameter of H.264 video codec, while the QP of Liu *et. al* refers to the quantization parameter of JPEG image codec. We obtain the resolution of Elf by timing the resolution of Elf's input (1920x1280) by 0.4 along width and height.

[3] Compared to these three works, Pakha *et. al* [19] extracts feedback from the low-quality inference results and thus cannot identify the objects that do not appear in the low-quality inference results, resulting in low accuracy; and other approaches stream the objects and the background in the same quality and thus waste bandwidth to encode the background.



(a) Assign HQ to region proposals (Confidence drop: 0.3)

(b) Pixels of high saliency returned by server-side DNN

(c) Assign HQ to high-saliency macroblocks (Confidence drop: 0.2)

Figure 2: The comparison of regions selected to be encoded in high quality between using region-proposal results as feedback and using saliency values as feedback.

only 2.24ms latency for object tracking on the Augmented Reality device; and Elf [29] saves bandwidth by 52.6%, while with less than 1% application accuracy sacrifice.

3 BETTER SERVER-DRIVEN FEEDBACK

We begin with the question: *what should be the format of server feedback?* We first show that existing designs extract server feedback that fails to directly capture the importance of different regions (§3.1), and then propose (§3.2) and evaluate (§3.3) an alternative design based on the well-studied concept of saliency in the computer vision community.

3.1 Need for better server feedback

Previous work uses bounding boxes returned by some region proposal networks (such as RCNN in object detection DNNs [20]) and picks some or all of the bounding boxes as the feedback. Such region-proposal-based feedback, however, suffers from two issues.

First, many pixels outside the region proposals can be crucial to high inference accuracy and thus need to be encoded in high quality. For instance, in Figure 2(b), the region proposal fails to mark the pixels to the right of the car's bounding box as important to inference accuracy. The observation that some pixels outside the object bounding box are important to inference accuracy can be intuitively explained as follows. These pixels, if encoded in high quality, might provide crucial contextual information about the object and its boundary that helps the DNN to separate the object from the background with high confidence. Some work [29] also enlarges bounding boxes in all directions to encode more environmental pixels in high quality, but it will significantly increase the high-quality regions.

Second, many pixels inside the region proposal can have little impact on inference accuracy, and thus do not need to be encoded in high quality. For instance, in Figure 2(b), the region proposal feedback will fail to mark some pixels in the car bounding box as unimportant to inference accuracy. Since existing solutions encode entire bounding boxes in high quality, it will lead to higher bandwidth usage than necessary.

The fundamental issue is that the region-proposal-based feedback fails to *directly* capture the regions whose encoding quality is crucial to the inference accuracy. Essentially, region proposals (or bounding boxes that might contain objects) seek to identify the pixels whose *absence or presence* will affect the inference results, whereas what we seek to identify the pixels whose *changes of values* (due to video compression) will affect the inference results.

3.2 Saliency as a more meaningful feedback

Ideally, the server feedback should directly measure the relative contribution of each pixel to the server-side DNN inference accuracy and indicate which quality level should be used to encode each pixel. To meet this goal, we introduce the well-studied concept of *saliency* from the computer vision community [2, 4]. Mathematically, it is computed as the gradient of the sum of the bounding box confidence scores with regard to the pixel RGB values. So why does saliency, as a server feedback, better indicate the regions whose encoding qualities are more important to inference accuracy?

Saliency, by definition, measures the impact of any local changes on a pixel value (*e.g.,* due to video encoding) on the DNN output. We use a simple example to contrast saliency with region proposals. Consider a two-class classifier with a *logistic* function as the final layer. A logistic function is an S-shaped curve between 0 and 1, where the output value changes slowly when it approaches 0 or 1. If an object has a very high confidence score, *i.e.,* a larger output of the logistic function, the function value remains relatively stable with a slight change on the input pixel values, and all pixels will tend to have low saliency values, because the confidence sits on the "plateau" of the logistic function (*i.e.,* it is unlikely to leave the plateau by any local pixel changes). In contrast, all pixels related to the object will firmly belong to a region proposal.

For the advantage of saliency as a direct indicator of important regions, we use it as the server-side feedback. Figure 2(b) shows an example saliency map (with high-saliency pixels labeled in bright). Given a saliency map, we first compute for each pixel the product of the saliency and pixel value difference between high quality and low quality, then compute the sum of the product in each macroblock (16×16), and finally encode the top 10% macroblocks with the highest sum in high quality. By multiplying the saliency with the low-quality pixel distortion, we can winnow out pixels whose values change only marginally when encoded in a low quality. This is not the only way to assign quality based on saliency, but this simple encoding scheme (of which an example is shown in Figure 2(c)) already addresses the aforementioned two issues of region-proposal-based feedback (§3.1).

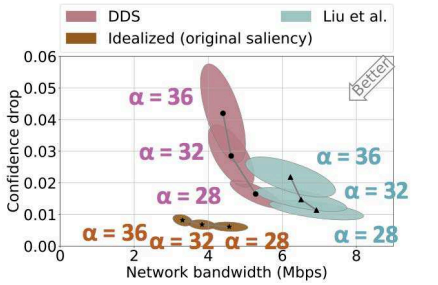
3.3 Potential gains of saliency-based feedback

Before we design a system to extract saliency as server feedback, it is natural to ask: *if* we have access to the saliency map, how much can we improve the performance, in accuracy and bandwidth usage? To show the full potential improvements that saliency as server-side feedback could bring along, we assume access to the *original saliency* derived from the uncompressed frames, *i.e.,* how much lowering the quality of each pixel from its original values will change the DNN output. We will remove this assumption in the next section.

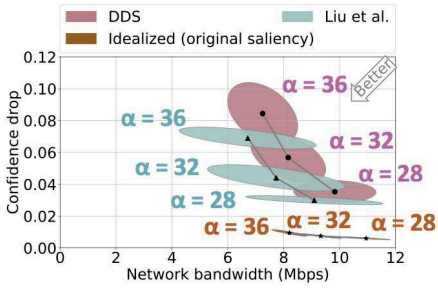
Here, we compare the performance in accuracy-bandwidth tradeoffs of this idealized design with existing server-driven designs (DDS [10], Liu *et. al* [17] and Elf [29]). We use object detection as the task, and FasterRCNN [20] from the PyTorch model zoo as the server-side DNN model. In this test, we use five traffic camera videos randomly selected from the test videos of DDS [1] and another four camera videos randomly selected from the test videos of Boggart [5] (the videos are downloaded from YouTube by keywords like "cityscape" and "street view"), and from each video, we select the first 300 frames each having at least one object. We label the two sets of test videos as *DDS* and *Boggart*. Although these videos are relatively short, their content varies from small objects to large objects and from fast-moving objects to relatively static objects, thus allowing us to test performance over a variety of content: the *DDS* videos include vehicles on highways and in intersections, whereas

the *Boggart* videos include people walking in college campuses and town squares. To compute the original saliency values, we run the backward propagation operation from the DNN output (sum of bounding box confidence scores) to the input uncompressed frame, which produces the per-pixel saliency (gradients with regard to the pixel RGB values). Given the original saliency per pixel, we select the 10% (16x16) macroblocks in the manner described in §3.2. We use QP of 2 (a high quality) to encode these macroblocks and a varying low quality for the remaining macroblocks. The encoding is done with H.264 [25]. Finally, we send the encoded video frame to the server-side DNN for inference.

Figure 3 compares the performance trade-offs between existing server-driven video streaming systems and our idealized design based on original saliency. Note that we assume that Elf [29] and Liu *et. al* [17] know the region proposals on all frames based on high quality video (this makes their performance strictly better). Thus, they lie on the same bandwidth-accuracy trade-off since they leverage the same technique (region-of-interest encoding) to encode the video based on the same feedback (the region proposals). We can see that under the assumption that we have accurate saliency values that models the real contribution of each pixel to the inference accuracy, on the *DDS* videos, using saliency as the server-side feedback reduces bandwidth usage by 18% and confidence drop by 76% compared to DDS [10], and reduces bandwidth by 40% and confidence drop by 56% compared to Liu *et. al* [17]; on the *Boggart* videos, it reduces accuracy drop by 87% compared to DDS [10] and by 84% compared to Liu *et. al* [17] without incurring substantially more bandwidth. We conclude that the potential of performance improvement of using saliency as the server-side feedback is significantly large. Note that this improvement is achieved by specifying encoding quality at a finer granularity (macroblocks) than prior work (region proposals).



(a) *DDS* videos



(b) *Boggart* videos

Figure 3: Performance trade-off comparison between existing server-driven video streaming systems and our idealized design (based on original saliency) on two video datasets featuring different contents. Each ellipse describes the 1-σ distribution of the performance across our test videos. α next to each ellipse represents the QP value used to encode the low-quality regions.

4 PRACTICAL DESIGN FOR SALIENCY-BASED FEEDBACK

While the potential improvements shown in Figure 3 are impressive, fully realizing them in practice is hard, since obtaining the original saliency requires sending the uncompressed video frames to the server DNN in the first place. To make it practical, we use an empirical study to test whether the server-side DNN can still extract near-original saliency values on a relatively low video quality and frame rate (sent in Phase 1 shown in Figure 1). Our preliminary results on a variety of videos have been encouraging—lowering the video quality and frame rate in phase 1 can still unleash most of the potential improvements of saliency-based feedback.

4.1 Key parameters of feedback extraction

Any implementation of a feedback extraction process has to set two parameters: (1) the video quality α (2) the frame rate k at which the client sends (in the first phase) to the server-side DNN to extract feedback. Existing designs of server-driven video analytics systems use certain static values for these parameters (see Table 1 for some examples), but as we will see shortly, these values often are not optimal.

Impact on performance: These parameters affect the trade-offs between the fidelity of the saliency information and the bandwidth usage to get the saliency. Sending uncompressed or high-quality frames to the server will use too much bandwidth, but if the saliency is derived from very low-quality frames, the low-quality pixel values can be so distorted from the original value that the inference output's gradients on these distorted pixel values will no longer indicate the original saliency of the pixels. A similar trade-off can be made by choosing the frequency (the frame rate of the video) at which DNN extracts saliency. A natural question, therefore, is whether it is possible to choose "sweet spots" for these parameters, *i.e.,* significantly lower the quality and frame rate of the video frames (in phase 1 of Figure 1) while still allowing the server-side DNN to extract near-original saliency to save bandwidth and improve accuracy.

Rationale of "sweet spots": We believe these sweet spots exist (which corroborates with the empirical results shown shortly). Intuitively, lowering the video quality α does change the pixel-level saliency, but these changes will not significantly alter the relative ordering between higher-saliency pixels and lower-saliency ones. This is because the saliency values empirically follow log-linear distribution (largely hold in our test videos), which means high and low saliency values have substantial gaps. Similarly, nearby frames have different yet similar pixel values, so if we lower the frequency of saliency feedback extraction k (*i.e.,* reusing the saliency extracted from recent frames), the relative order of high-saliency pixels and low-saliency pixels will be largely preserved.

4.2 Empirical impact of first-phase frame quality and frame rate on performance

To confirm the intuitions, we empirically compare the accuracy-bandwidth trade-offs of using the original saliency with the performance under different frame qualities (measured in QP $\alpha \in \{28, 32, 36, 40, 44\}$) and frame rates (calculating saliency every $k \in \{2, 3, 5, 10\}$ frames) in the first phase of Figure 1. We follow the same

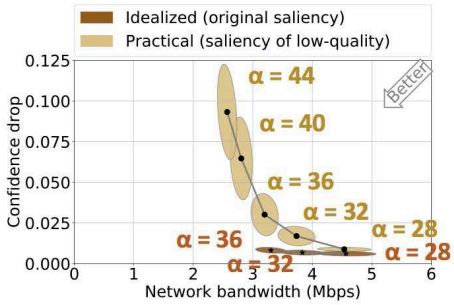


Figure 4: Accuracy-bandwidth trade-off comparison of our approaches with different first-iteration frame quality on the *DDS* videos.

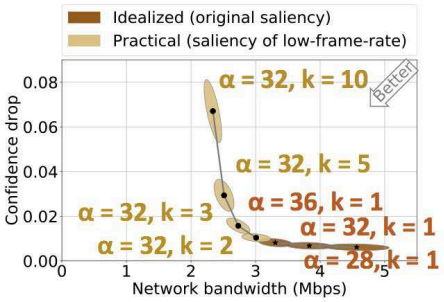


Figure 5: Accuracy-bandwidth trade-off comparison of our approaches with different saliency extraction frequency on the *DDS* videos. k next to each ellipse represents the frame rate of the first phase, *i.e.,* extract saliency once every k frames.

experiment setting (*e.g.,* DNN models, video encoders) as described in §3.3 and use the five *DDS* videos.

Frame quality for saliency extraction: Figure 4 shows that confidence drop increases approximately exponentially as the quality of the frames in the first phase degrades (*i.e.,* increasing α). More importantly, the performance of $\alpha = 28$ and $\alpha = 32$ show that saliency values derived from mildly compressed images yield a sufficiently accurate estimate on which regions in a frame should be encoded with high quality. The resulting performance is close to the idealized design.

Frame rate for saliency extraction: Next, we test if one can further reduce bandwidth usage without hurting accuracy by extracting server feedback less frequently. We use $\alpha = 32$ as the QP value of the frame for feedback extraction as part of the findings in the previous experiment. Figure 5 compares the accuracy-bandwidth trade-offs under different frequencies of server-side feedback extraction with the trade-offs and with the potential gains when using the original saliency values derived from the uncompressed frames. We can see that extracting saliency values once every 3 frames can provide sufficiently low accuracy drop and 29% reduction in network bandwidth as compared to extracting saliency values for every frame on the *DDS* videos.

Improvement of the realistic design: Finally, we show that the potential gain displayed in §3.3 can be largely attained in practice. We use the values identified above ($\alpha = 32, k = 3$) to set the frame rate and encoding quality based on which saliency feedback is extracted. Figure 6 compares the resulting performance with that of several alternative server-driven design as shown in Table 1. We can see that even when we relax the assumption that the client has access to saliency values derived from the uncompressed video frames, on the *DDS* videos, we still achieve 43% more bandwidth saving and 12% less accuracy drop compared to DDS [10], or 57% more bandwidth saving compared to Liu *et. al* [17] without compromising inference accuracy greatly; on the *Boggart* videos, we achieve 21% more bandwidth saving and 68% less accuracy drop compared to DDS [10], or 15% more bandwidth saving and 60% less accuracy drop compared to Liu *et. al* [17]. In short, our system, despite featuring simple and preliminary design choices, already shows significant performance improvement as compared to latest systems that take the server-driven approach, which demonstrates the untapped yet large potential of server-driven designs.

We acknowledge that the best parameters for our test videos do not necessarily apply to all videos. Our goal here is not to design the optimal parameters settings; instead, we want to illustrate that choosing appropriate values for these key parameters can lead to favorable performance tradeoffs.

5 LIMITATIONS AND DISCUSSION

More vision tasks: In this short paper, we use object detection as an example application to demonstrate the potential performance gain brought by the idea of saliency. The concept of saliency is compatible with all vision DNNs and thus our approach could be applied to other computer vision tasks. However, there is no guarantee that our approach would yield substantial performance gain on these tasks as compared to existing baselines.

Other camera/network settings: In this paper, we make the assumption that the cameras have very limited local compute resources and are only able to encode and stream video frames. If the camera could support more expensive operations such as running DNN inference, we could support more designs, *e.g.,* running a cheap client-side model to identify pixels important to inference accuracy. Also, our current design can support parameter tuning to adapt to dynamic network bandwidth. For example, under bandwidth-constrained conditions, we could decrease the frequency at which saliency is extracted in order to reduce bandwidth usage. The logic of tuning these parameters for different network settings is part of our future work.

System usage and overhead: Saliency computation involves back-propagation operations and is expected to incur substantial server-side GPU memory usage. Our measurement with FasterRCNN [20] shows that as compared to forward inference, saliency computation incurs 82% more GPU memory usage. However, our measurement shows that it would not involve substantial overhead in terms of CPU usage and main memory: as compared to forward inference, saliency computation incurs approximately the same CPU usage and main memory usage.

Limitations of current datasets: We collect all videos from the highway traffic camera dataset used by the DDS paper [10] and the videos used by the Boggart paper [5]. It is possible that individual videos contain biases due to camera positions and road conditions. For example, if the camera is positioned such that a large proportion of the video frames contain no object or very few vehicles on the road, then the room for improvement would be relatively small as there are few objects of interests for which we can improve inference accuracy. However, the datasets we selected contain videos featuring a variety of contents, which help reduce the biases of individual videos.

6 POSSIBLE EXTENSIONS

So far, we have only scratched the surface of server-driven video analytics systems. Though our initial design (described in §4) already shows impressive improvement, it is by no means optimal; instead, we hope that it would inspire more research to explore the

broader design space of server-driven video analytics systems. In particular, our design can be extended along at least five dimensions, which gives rise to new challenges for the mobile and networking communities.

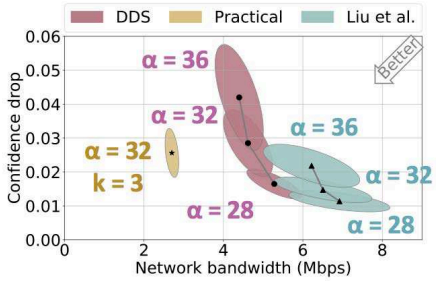
Fine quality granularity: Our initial design in §4 (and other server-driven solutions) employs only two quality levels (a high quality for high-saliency macroblocks and a low quality for remaining pixels). Our analysis, however, shows a log-linear distribution of the saliency values across pixels (*i.e.,* the saliency values are significantly different even among the top 10% macroblocks that are assigned with high quality), which naturally suggests the use of multiple quality levels to differentiate the impact of macroblocks.

Saliency-aware temporal encoding: To focus on the spatial quality assignments driven by server-side feedback, we disabled temporal encoding (*e.g.,* motion vectors) in this preliminary work. Temporal encoding, such as motion vector estimation, is complementary to spatial encoding, and we will study it as part of the future direction. While adding the temporal encoding (as supported in existing video codecs) is logically straightforward, the fact that the saliency-based feedback varies across frames means that care should be taken to stabilize the saliency-based quality assignment across frames in order to facilitate efficient use of motion vectors.

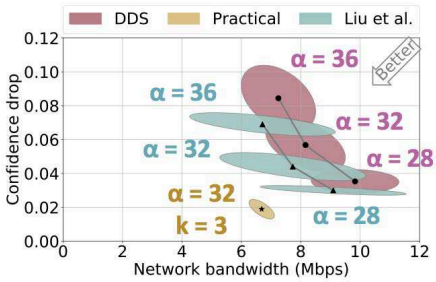
Efficient feedback extraction: A high-level message from this short paper is to rely on saliency from the server-side DNN, but extracting its values (and other fine-grained information) from the server-side DNN involves additional compute overheads (*e.g.,* backward propagation), which might be crucial under server-side resource constraints. While we empirically found that lowering the frequency of saliency computation helps, we also notice that there are several optimizations proposed in other contexts to speed up backward propagation and saliency inference, by exploiting sparsity of gradients [21] or approximating saliency with a cheaper model [8].

Server-driven input: Besides the particular design choices mentioned in section §4.2, there are a variety of designs that make clever use of saliency in server-driven video analytics systems. In particular, we could adapt the input to the DNN based on server-side extracted saliency values. For example, the server could ask the client (video source) to lower the quality of certain regions with lower saliency values, thus allowing these regions to be encoded with lower quality in future streaming through a closed feedback loop. Consider the case where the location and angle of the camera is fixed. In this scenario, this approach would enable the system to quickly identify regions in video frames that consistently do not contain objects (*e.g.,* static and fixed background) and label them as areas that should be consistently encoded in low quality in the future.

Cheap client-side saliency extraction: The idea of using saliency for improving video analytics can be extended beyond server-driven video analytics systems. One way of using saliency in a different scenario is to train a cheap quality selector on the edge clients that quickly and cheaply estimates the saliency values for all macroblocks in a frame and determines near-optimal quality levels for them. This design might reduce end-to-end delay and server-side computation load significantly as compared to systems in the server-driven paradigm. However, there exists a trade-off between the compute cost of the model and the accuracy of extracted saliency, and how to find a design that optimizes this trade-off is part of our future work.



(a) *DDS* videos



(b) *Boggart* videos

Figure 6: Accuracy-bandwidth trade-off comparison between several server-driven video streaming systems and our practical design on two video datasets featuring different contents. α and k are the QP value and the frame rate (in the first phase) at which the saliency feedback is extracted.

7 RELATED WORK

Besides those based on the server-driven approach (§2), here we briefly discuss alternative designs that optimize the trade-offs between bandwidth usage and inference accuracy.

A common approach runs certain simple logic (using the camera's limited local compute resource) to identify which frames are of little contribution to the inference accuracy and thus can be discarded [6, 7, 15, 27]. However, this approach encodes the remaining frames with uniform quality, and thus wastes bandwidth to encode the background. Another approach deploys cheap camera-side vision models to distinguish objects and background, and then uses low quality to encode the background to save bandwidth [9, 28]. Due to the limited compute resource, however, the camera-side model is not accurate enough to detect all potential objects (especially small or partially occluded objects), causing the server-side DNNs to miss these objects, as observed in [10].

Another line of research extracts and compresses intermediate features of the video by camera-side DNN and streaming the features to the server for analytics [3, 11, 14, 18, 26]. Though it is promising for video/image classification, it cannot efficiently compress the video for object detection since object detection requires knowing much more information, like the location and the size of multiple objects, to deliver accurate object detection results while classification only needs to generate a correct class label.

8 ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd, Chunyi Peng, for their insightful feedback. This work is supported by NSF grants CNS-1901466, CNS-2131826, CNS-2153449, CNS-2152313, and CNS-2140552.

REFERENCES

[1] Dds codebase. https://github.com/KuntaiDu/dds.

[2] Saliency user study. https://arxiv.org/pdf/2002.00772.pdf. (Accessed on 10/01/2021).

[3] Video coding for machines (the moving picture experts group). https://mpeg.chiariglione.org/standards/exploration/video-coding-machines.

[4] Visualizing cnn using saliency maps. https://towardsdatascience.com/practical-guide-for-visualizing-cnns-using-saliency-maps-4d1c2e13aeca. (Accessed on 10/01/2021).

[5] N. Agarwal and R. Netravali. Boggart: Accelerating retrospective video analytics via model-agnostic ingest processing, 2021.

[6] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. R. Dulloor. Scaling video analytics on constrained edge nodes. *arXiv preprint arXiv:1905.13536*, 2019.

[7] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *SIGMOBILE 2016*.

[8] P. Dabkowski and Y. Gal. Real time image saliency for black box classifiers. *arXiv preprint arXiv:1705.07857*, 2017.

[9] X. Dai, X. Kong, T. Guo, and Y. Huang. Cinet: Redesigning deep neural networks for efficient mobile-cloud collaborative inference. In *SIAM 2021*.

[10] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang. Server-driven video streaming for deep learning inference. In *SIGCOMM 2020*.

[11] J. Emmons, S. Fouladi, G. Ananthanarayanan, S. Venkataraman, S. Savarese, and K. Winstein. Cracking open the dnn black-box: Video analytics with dnns across the camera-cloud boundary. In *HotEdgeVideo 2019*.

[12] A. Habibian, T. v. Rozendaal, J. M. Tomczak, and T. S. Cohen. Video compression with rate-distortion autoencoders. In *ICCV 2019*.

[13] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: optimizing neural network queries over video at scale. *In VLDB 2017*.

[14] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *Acm Sigplan Notices*.

[15] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *SIGCOMM 2020*.

[16] H. Liu, H. Shen, L. Huang, M. Lu, T. Chen, and Z. Ma. Learned video compression via joint spatial-temporal correlation exploration. In *AAAI 2020*.

[17] L. Liu, H. Li, and M. Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *MobiCom 2019*.

[18] Y. Matsubara, S. Baidya, D. Callegaro, M. Levorato, and S. Singh. Distilled split deep neural networks for edge-assisted real-time systems. In *HotEdgeVideo 2019*.

[19] C. Pakha, A. Chowdhery, and J. Jiang. Reinventing video streaming for distributed vision analytics. In *HotCloud 18*.

[20] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS 2015*.

[21] A. Sarma, S. Singh, H. Jiang, A. Pattnaik, A. K. Mishra, V. Narayanan, M. T. Kandemir, and C. R. Das. Exploiting activation based gradient output sparsity to accelerate backpropagation in cnns. *arXiv, 2021*.

[22] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV 2017*.

[23] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[24] Y. Wang, W. Wang, J. Zhang, J. Jiang, and K. Chen. Bridging the edge-cloud barrier for real-time advanced vision analytics. In *HotCloud 19*.

[25] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 2003.

[26] S. Xia, K. Liang, W. Yang, L.-Y. Duan, and J. Liu. An emerging coding paradigm vcm: A scalable coding approach beyond feature and signal. In *ICME 2020*.

[27] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee. Awstream: Adaptive wide-area streaming analytics. In *SIGCOMM 2018*.

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

[29] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang. Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading. In *MobiCom 2021*.