CVF: Cross-Video Filtration on the Edge

(article starts on next page)

CVF: Cross-Video Filtration on the Edge

Ali Rahmanian
Umeå University
Umeå, Sweden
ali.rahmanian@umu.se

Siddharth Amin
Chalmers University of Technology
Gothenburg, Sweden
aminsi@student.chalmers.se

Harald Gustafsson
Ericsson Research, Ericsson AB
Lund, Sweden
harald.gustafsson@ericsson.com

Ahmed Ali-Eldin
Chalmers University of Technology
Gothenburg, Sweden
ahmed.hassan@chalmers.se

Abstract

Many edge applications rely on expensive Deep-Neural-Network (DNN) inference-based video analytics. Typically, a single instance of an inference service analyzes multiple real-time camera streams concurrently. In many cases, only a fraction of these streams contain objects-of-interest at a given time. Hence, it is a waste of computational resources to process all frames from all cameras using the DNNs. On-camera filtration of frames has been suggested as a possible solution to improve the system efficiency and reduce resource wastage. However, many cameras do not have on-camera processing or filtering capabilities. In addition, filtration can be enhanced if frames across the different feeds are selected and prioritized for processing based on the system load and the available resource capacity. This paper introduces CVF, a Cross-video Filtration framework designed around video content and resource constraints. The CVF pipeline leverages compressed-domain data from encoded video formats, lightweight binary classification models, and an efficient prioritization algorithm. This enables the effective filtering of cross-camera frames from multiple sources, processing only a fraction of frames using resource-intensive DNN models. Our experiments show that CVF is capable of reducing the overall response time of video analytics pipelines by up to 50% compared to state-of-the-art solutions while increasing the throughput by up to 120%.

Keywords: Edge, Video Analytics, Video Filtration, Codecs.

ACM Reference Format:
Ali Rahmanian, Siddharth Amin, Harald Gustafsson, and Ahmed Ali-Eldin. 2024. CVF: Cross-Video Filtration on the Edge. In *ACM Multimedia Systems Conference 2024 (MMSys '24), April 15–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3625468.3647627

MMSys '24, April 15–18, 2024, Bari, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0412-3/24/04.
https://doi.org/10.1145/3625468.3647627

1 Introduction

Video cameras are ubiquitous with applications ranging from autonomous driving, to surveillance heavily relying on video processing [1, 2]. To handle the enormous size of the generated video workloads, processing video feeds on a remote cloud is not feasible due to network latency for many applications. This has led to an increased interest in using edge clouds deployed closer to the users for the analytics instead [3]. However, one problem with using edge clouds is the fact that most modern video analytics is based on computationally heavy DNN models for tasks such as action recognition, motion tracking, and object detection [4]. Accurate DNNs are costly to use, particularly on edge clusters with fewer GPU resources compared to data centers. To reduce the load on the edge, many techniques to filter frames that do not contain "interesting" information for the application have been suggested [5–7].

Frame filtration requires the filter to be context-aware of what is relevant for an application, e.g., for traffic cameras, non-moving objects such as trees on the side of the road are not interesting. Some solutions rely on binary classifiers to decide if a frame should be filtered or not [7]. However, binary classification-based filtration is not adaptive to the resources and workload variability on the edge, e.g., due to increased traffic, on-car cameras produce dynamic loads on edge GPUs. Prioritizing frames, especially when certain streams contain safety-critical information, becomes crucial given workload variability.

This paper introduces CVF, a Cross-Video Filtration framework for filtering video streams from multiple cameras on the edge. CVF employs frame differencing, lightweight object classification, and prioritization techniques designed to effectively handle frame filtering from multiple cameras. CVF manages stream oversubscription on a processing node, forwarding only relevant or prioritized frames for further processing to downstream, heavier DNN models. Our contributions can be summarized as follows;

1. We introduce an enhanced single-stream filtration technique based on the used video codecs, implementing

our filtration using H.264 codecs, one of the most commonly used codecs for streaming cameras.

2. We design CVF, a Cross-Video Filtration framework for edge systems capable of adapting the level of filtration based on the available resources and the current load from a video streaming system.
3. We introduce a novel method to rank video streams and frames for prioritization, processing based on this ranking and resource availability, offering differentiated response times according to frame priority.

2 Background

2.1 Edge Computing

Edge computing deploys a set of small-scale resources at the network edge, closer to the applications. Applications can then offload part of their computations to these edge resources [3]. Video analytics on the edge provide better communication latency while reducing the need to transfer large amounts of data over the network to remote data centers. Hence, edge clouds have been suggested as the infrastructure to use for, e.g., AR and VR applications [8, 9], DNN serving [2, 4], and transcoding of volumetric videos [10]. On the downside, edge servers are typically small-scale with fewer resources compared to large-scale cloud environments [11]. This calls for efficient edge resource usage.

Edge resources can range from small "wimpy" nodes such as Jetson Nanos [12] and TPUs to FPGAs [13, 14], to clusters with powerful GPUs such as the A100s [15]. In this paper, we target the later infrastructures, where full-fledged GPUs are deployed at the edge.

2.2 Video Analytics and Filtration on the Edge

The problem of analyzing video feeds on the edge has recently received increased attention from the research community [16–18]. Previous works have focused on, for example, optimizing DNNs to run on smaller GPUs [12, 19]; designing offloading techniques between small devices and more powerful edge resources [20, 21]; and on scheduling different DNNs based on the workload [7]. Many of these techniques focus mostly on optimizing the machine learning pipeline or algorithms to enable the offloading to the edge. However, one aspect that remains very important, especially for edge applications with massive video data generated per second, is filtration [5–7]. Filtration aims to reduce the overall computations of the edge application by removing some of the frames from video feeds that do not contain "objects-of-interest". Filtration is crucial when computationally heavy processing tasks such as pose estimation, semantic segmentation, and action recognition are involved.

While for some cameras, some filtration work can be done on cameras [5, 6] making use of the local computational power of cameras or small attached edge accelerators, many cameras lack these capabilities. In addition, local filtration does not consider what other camera feeds contain. In addition, on-camera filtering algorithms are much harder to update in real wide-area deployments. Other state-of-the-art techniques focus on training specialized per-stream neural network-based filters [7]. In this paper, we argue that for many applications, it is important to do the filtration across all the video streams from all cameras considering the global load on the edge devices, along with the video content.

2.3 Understanding Video Frame Compression

One key observation that CVF makes use of is that not all frames are created equal in a video stream. Video streams employ video codecs for compression to optimize the size of video content for efficient transmission over networks, significantly reducing bandwidth requirements. In addition, compression enhances storage capabilities and facilitates smoother playback across various devices. In CVF, we make use of this fact, focusing our work on the H.264 codecs [22], the most popular codec by far today.

Figure 1 displays the various frame types in an H.264 coded video: the I-frame, often referred to as a keyframe, presents a complete, independent image unaffected by neighboring frames temporally, employing solely intra-frame coding for compression [22]. Conversely, a P-frame represents a predicted image based on preceding (historical) frames, capturing only the alterations from those frames. Meanwhile, a bidirectional predicted frame (B-frame), utilizes content from both preceding and subsequent (future) frames. I-frames lack temporal dependence and B-frames usually achieve the highest compression rates among frame types. However, certain scenarios may lead to P-frames outperforming B-frames in compression efficiency[23].

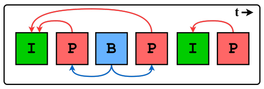


Figure 1. A sequence of frames with different types in a video encoded using H.264 format.

A sequence of video frames starts with an I-frame, followed by P- and B-frames forming a Group of Pictures (GOP). I-frames act as immediate decoder refresh points to ensure subsequent frames remain independent of preceding ones [22, 23]. A Video frame contains many pixels based on their resolution. Instead of compressing individual pixels, video codecs utilize square blocks of pixels within a frame known as macroblocks (MBs) for compression purposes. Video codecs employ macroblocks as the main unit for compression.

Within an encoded frame, motion vectors (MVs) depict the direction and distance of motion within the frame concerning preceding or subsequent frames. They signify the movement of macroblocks between frames, crucial for efficient inter-frame compression by tracking motion dynamics. These details belong to the compressed domain of a video frame. Upon decoding, the frame's bitmap demonstrates the pixel domain, representing an independent video frame.

2.4 Why CVF?

Our work is driven by two industrial real-time use-cases. Our first use-case is an industrial automation use-case where we aim to automate a large factory using hundreds of ceiling and on-robot cameras to "sense" the factory floor environment, then based on edge-analytics using DNNs, schedule the path of ground transportation robots in real-time. In this use-case, one simple, but rather costly, solution could be to assign one GPU per stream. However, by deploying CVF, we aim to reduce the total number of GPUs required to automate the factory by allowing multiple (up to 15) streams to concurrently run on an edge GPU.

Our second use-case is a 5G-enabled traffic management application where the aim is to use cameras mounted on vehicles, and from fixed road cameras to help collision-avoidance in assisted-driving vehicles. In this use-case, the cameras offload the computations to the edge, which communicates directly with the vehicles using a 5G network to send warnings based on the video analytics. Notably, in both use-cases, the number of video streams is not fixed as the number of cameras in the range of an edge can increase or decrease based on the total number of vehicles or on-robot cameras in the area. For both use-cases, the the end-to-end tail latency of the computations needs to be less than 100ms.

While for many applications the spatial density of cameras can be low, with only a few cameras at a given area, other applications such as the factory floor automation use-case have a high camera density, with ceiling cameras mounted every 5 to 10 meters, and cameras installed on tens-to-hundreds of robots with a factory typically having a few hundred robots [24]. In these applications, video feeds from all cameras need to be considered before deciding which frames need to be filtered. Many of the camera feeds will overlap, and many others will have no activity for some time (no human or robot in the scope of the camera), and can thus be filtered altogether. However, since this is a safety-critical system, and filtration can have false negatives, it is important to make sure that the system allows some frames from the filtered feeds to be processed using the more accurate and heavy DNN models. This is what we aim to achieve by designing CVF. To the best of our knowledge, none of today's state-of-the-art approaches are capable of solving the problems of varying numbers of cameras, frame-prioritization, and resource-aware filtering.

3 CVF: Cross-Video Filtration Pipeline

Figure 2 shows the CVF pipeline. This pipeline selectively filters frames in compressed and pixel domains using five primary modules: Partial Decoding, MV Filtering, Frame Ranking, Full Decoding, and Binary Classifier.

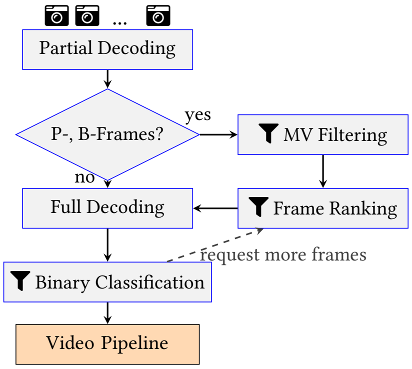


Figure 2. CVF pipeline for cross-video filtration on the edge (CVF modules are represented with the ▼ icon).

3.1 Partial decoding and MV Filtering

CVF performs partial decoding with a primary focus on extracting Motion Vectors in the initial step. This approach significantly reduces computational complexity during decoding [25], considering that numerous frames may be filtered out and do not require full decoding. The partial Decoding module receives frames from multiple video feeds and performs lightweight extraction of their compressed-domain attributes, specifically MVs.

In the next step, MV Filtering analyzes and identifies relevant MVs within the frames, assessing their activity levels to filter out frames with minimal or no activity. P- and B-frames capture temporal differences between the frame being processed and its historical or future frames. MVs represent the movement of MBs in the scene and can offer insights into the activity within the frame. CVF seeks to detect the frame activity by analyzing MVs within the frame before decoding the entire frame. This approach optimizes processing by selectively decoding frames according to their activity level, thereby enhancing the performance of the constrained edge servers and improving throughput as a result.

CVF computes the relative fraction of MVs within a frame to assess its activity level. Frames with low relative activity are filtered out do not undergo a complete decoding and are entirely skipped, optimizing system resources. Despite the numerous MVs within a frame, many might not pertain to objects of interest. Hence, it is crucial to consider only relevant MVs when determining the relative activity of frames. CVF proposes filtering out irrelevant MVs unrelated to objects of interest using two main mechanisms: 1) Repetitive MV Exclusion Filtering (RMVEF) and 2) Median Filtering (MF). Figure 3 depicts the role of MV Filtering within the complete hierarchy of the CVF filtration pipeline. Figure 4 provides an illustrative example to demonstrate how MV Filtering module performs in CVF to filter out frames and accurately calculate relative activity within a frame.

RMVEF. Certain spatial areas in video frames, often contain MVs irrelevant to the primary contents of the frames, e.g.,

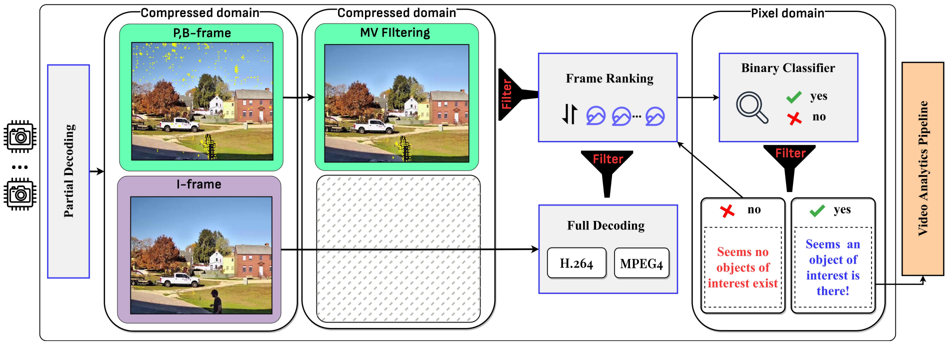


Figure 3. CVF in Action. Decoded images are not available during execution, shown here for representation of CVF performance.

areas with no objects of interest such as skies or trees in an outdoor scene. CVF employs a Binary Bitmap of Excluded MVs (BBEM) to identify and exclude these MVs. The BBEM represents areas where MVs exist but does not correspond to objects of interest, depicted as black pixels in the bitmap (See Figure 4b). CVF maintains and updates a BBEM for individual video feeds, marking areas where irrelevant MVs persist. RMVEF utilizes the BBEM to filter out non-relevant MVs, ensuring that the remaining MVs likely correspond to objects of interest (See Figure 4c-f). Figure 4d illustrates yellow arrows are MVs within the frame that each represent the past and the current location of an MB within the frame compared to the historic or future frames. The creation of MVs from the perspective of video codecs does not precisely capture the real motion of objects of interest within a frame. Also, a single MV might not denote significant activity within an object, but a collection of MVs can represent object-related actions. Sparse, dispersed MVs in a frame generally signify noise and necessitate filtering.

MF. CVF creates a binary bitmap from frame MVs, wherein each pixel denotes an MB within the frame. Relative pixels in the MV map to MV starting and destination points are marked as 1 (black areas), and the rest of cells in the MV map without MVs as 0 (white areas) in Figure 4e and Figure 4h. To filter out noise and sparse MVs, CVF applies MF using a 3×3 sliding window over the MV map. Median values within the window replace pixel values in the MV map. MF filters an MV if both its relative starting and destination pixels in the MV map are smoothed to zero (white). Figure 4i illustrates the final relevant MVs remaining after applying the MV Filtering module to the raw bitmap of the frame. It demonstrates that RMVEF excludes MVs independent of objects of interest, while MF filters out sparse MV noises (highlighted MBs by red circles in Figure 4f are filtered out in Figure 4h when performing MF mechanism).

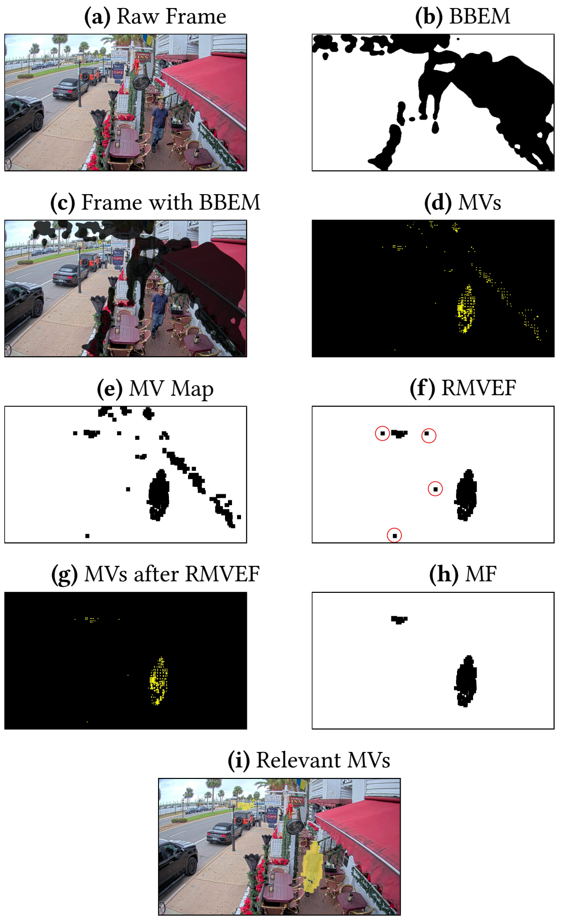
Frame Filtering. CVF filters frames based on the non-zero pixel percentage in the MV map after filtering. The threshold adapts for each camera to ensure relevance across scenes.

The threshold-based approach aligns with frame filtering and background subtraction techniques in related literature, aiding in excluding frames lacking significant activity [5, 26]. Figure 5a illustrates the MV Filtering module eliminating all MVs despite the presence of numerous initial MVs within the frame. As a result, no MVs remain in the frame after MV Filtering, leading to the exclusion of the frame from the execution of subsequent modules in the CVF pipeline and the video pipeline after partial decoding. In contrast, Figure 5b portrays a distinct frame sourced from the same video feed. In this case, relevant MVs related to an object of interest (e.g., a person) remain after MV Filtering. Consequently, the frame proceeds through the next corresponding module of the filtration pipeline under the conditions.

3.2 Frame Ranking and full decoding

After MV Filtering, the Frame Ranking module aims to prioritize which frames and video feeds get processed on the edge GPU. A frame with a higher priority has a higher ranking for processing. A batch of received frames from multiple stream video feeds compete based on their ranking for GPU resources. The number of compute instances, denoted by N, equals the maximum concurrency level for the video pipeline on a given GPU, i.e., how many frames can be processed by the GPU with heavier models on average without violating latency constraints. This value is learned during operation.

The video pipeline waits to receive a batch of N frames for batch processing with concurrent compute instances of the video analytics pipeline on the GPU. We demonstrate how a proper concurrency level can be determined for a video pipeline on a GPU in the experimental section. If the load is much larger than N, then a large number of frames must be filtered or delayed. The top N frames in the ranking list are prioritized on the GPU. CVF prioritizes frames from multiplex stream video feeds based on activities within the frames. Frame Ranking enables the processing of relevant frames, i.e., they have a high level of activity and contain relevant objects. The Frame Ranking module bases priorities on two factors: the relative activity within a frame (movement) and *the time elapsed since the video pipeline processed a frame from the stream video feed by a compute instance.* The first factor prioritizes video feeds with more activity for



(a) Raw Frame **(b)** BBEM **(c)** Frame with BBEM **(d)** MVs **(e)** MV Map **(f)** RMVEF **(g)** MVs after RMVEF **(h)** MF **(i)** Relevant MVs

Figure 4. An illustrative example of MV Filtering in CVF. Decoded images are not available during CVF execution.

frame processing, while the second ensures the allocation of compute resources to frames from less active feeds. The priority, R_s, for each frame is calculated using Equation 1:

$$R_s^t = \alpha \times A_s^t + \beta \times \lambda_s^t \tag{1}$$

where,

$$\alpha + \beta = 1 \mid 0 \leq \alpha \leq 1 \& 0 \leq \beta \leq 1 \tag{2}$$

$$0 \leq A_s^t \leq 1, \tag{3}$$

and A_s^t is the relative activity of the current receiving frame from video feed s at time t; λ_s^t is the time elapsed between the last time a frame was processed and the time the rank is calculated, i.e. t; *alpha* and β respectively denote a coefficient to give weight to relative activity within a frame and total elapsed time that a frame from the same stream video feed has not been processed.

The Frame Ranking module calculates the priority of individual frames using Equation 1 and arranges them accordingly to filter out excessive frames with lower rankings. Once the frames are ranked, the Full Decoding module constructs a frame bitmap using the video codecs, utilizing all frame elements, including MVs and residual maps. Within the filtration pipeline, CVF decodes all I-frames and any P-/B-frames not filtered by preceding modules. The resulting bitmap from full frame decoding is used for pixel-domain assessments by the Binary Classifier and potentially by the video analytics pipeline. Figure 3 illustrates the role of the Frame Ranking module within the hierarchy of the CVF filtration pipeline.

3.3 The Binary classifier

Upon receiving an I-frame from a video feed or if a P- or B-frame passes the MV filtering stage, the frame undergoes analysis using a binary classifier after complete decoding. This step aims to discern the presence of specific objects of interest within the frame. For example, in traffic analytics, identifying cars constitutes a key object of interest. The binary classifier, a lightweight model, specializes in recognizing the existence of predefined object classes within frames.

The knowledge about the objects of interest is typically provided to the binary classifier during an offline training phase. The binary classifier is trained using a large labeled dataset containing annotated examples of frames with and without the objects of interest (i.e., "yes" when an object of interest is present or "no" when there are no objects of interest in the frame). The binary classifier learns to distinguish between frames containing the objects of interest and those that do not by analyzing various features present in the frame. These features may include shapes, textures, or other visual characteristics indicating object presence.

The output of the binary classifier signifies the existence of an object of interest with a 'yes' or 'no' response. However, it does not provide granular details about detected objects such as type or location; its sole purpose is to confirm object presence within the frame. Frames yielding a 'no' response from the binary classifier are filtered out by CVF.

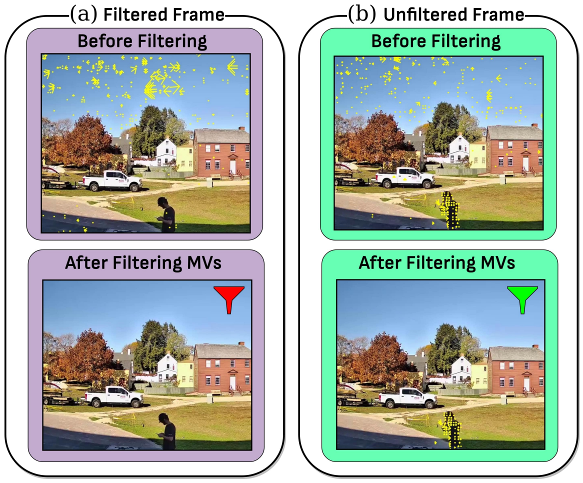


Figure 5. Example of MV filtering in CVF. Decoded images are not available during CVF execution.

Application developers configure the object classes within the binary classifier based on the requirements of the video analytics application. For instance, classes might include distinctions like *Humans* or *No Humans*, or *Robot* or *No Robot* categories. This binary classification step is crucial in the video filtering pipeline, identifying frames with objects of interest for subsequent in-depth analysis based on the specific objectives of the video analytics application.

4 Evaluation

We evaluate CVF under various workloads and scenarios, measuring its overhead costs, accuracy, and performance while comparing it with two state-of-the-art solutions.

4.1 Experimental Setup.

Hardware Setup. All experiments were conducted in an emulated edge environment with 4 servers, including 3 servers with CPUs and one server equipped with a compute accelerator. Kubernetes V1.22.13 with VM compute instances from Google Cloud [27] were used to emulate the edge cluster. We used three E2 instances with 2 vCPUs and 7.5 GB of memory for edge servers without compute accelerators. We also used an N1 compute instance from Google Compute Cloud with 2 vCPUs, 7.5 GB of memory, and one Tesla V100 GPU for the accelerator-equipped edge server.

Configuration. On the CPU servers, we simulate streaming video feeds using GStreamer [28] and the RTP protocol over UDP [29]. Video feeds are emulated on a server with a CPU. We use mv-extractor [30] to perform partial video decoding and extracting MVs from encoded frames using H.264 and MPEG-4 formats [31]. We deploy the filtration and the video pipeline on GPU servers.

Video Pipelines. We evaluate the performance of CVF using DEKR pose estimation DNN model [32] in the first two experiments and using the Amber Alert [33] video analytics pipeline in the third experiment.

Dataset. We evaluate the performance of the proposed filtration method on six captured videos from fixed-angle cameras in different locations. Table. 1 provides details of the dataset with different temporal activities. According to the table, captured videos in the dataset have variable frame rates with activities. For example, in `Targhee`, only 15% of frames with activities require processing due to high temporal changes. In contrast, `Factory` captures 82.5% of frames with activities that must be processed from an indoor ceiling camera in a factory setting.

Evaluation Criteria. We measure throughput according to the end-to-end processing latency. The total response time of a single frame includes the latency of decoding, queuing, filtration overhead, and pipeline execution. Throughput demonstrates the number of frames per second that can be completed within a second. We also measure the ratio of delayed frames by counting the number of frames that have a response time longer than the latency constrained, i.e. 100 *ms*. Response time and GPU utilization are two other evaluating metrics, that explain how fast and how heavy is CVF compared to other baseline filtration methods. Precision explains what ratio of frames with activities are correctly processed through the pipeline compared to the total number of processed frames. Recall refers to the ratio of frames processed with activities over the total received frames with activities. F1-Score represents the harmonic mean of precision and recall in order to evaluate true positive, false positive, and false negative altogether on accuracy.

Baselines. We evaluate CVF with two baseline methods: (1) FilterForward [6] and (2) NoScope [7]. FilterForward is a system designed for real-time and cloud-based video analytics, efficiently identifying and offloading only the most relevant video sequences for further analysis. It employs a localized binary classifier with a lightweight architecture for detecting objects within the original video frames. NoScope utilizes highly efficient difference detectors to highlight temporal changes across frames in video streams. NoScope enables fast detection of frames with objects and activities against static frames devoid of objects.

Video	Resolution	Activity in Frames (%)	Frames	Camera
Factory	720×480	82.5%	1920	Indoor
Targhee [34]	720×480	15%	3600	Outdoor
Village [35]	720×480	46.6%	3600	Outdoor
Museum [36]	720×480	37.5%	3600	Outdoor
Bar [37]	720×480	47.5%	3600	Outdoor
Resort [38]	720×480	18.8%	3600	Outdoor

Table 1. Details of the dataset used for stream video feeds.

4.2 Single-Stream Filtration

In this experiment, we aim to measure the accuracy and performance cost of CVF compared to the two baseline methods when processing frames from a single stream video feed.

Figure 6 shows the distribution of GPU time spent processing different parts of the filtration pipeline as well as the video pipeline. The percentages denote the average GPU time spent on each part divided by the total GPU activity time within a time window. Because there is only one stream that requires GPU processing, the GPU does not get fully saturated in all scenarios. The figure shows that CVF by far imposes lower overhead compared to baseline methods where 98% of the GPU time goes for processing video pipeline compared to 94% of GPU time in baseline methods when a single feed generates the workload. This is because baseline filtration methods fully perform in the pixel domain while CVF performs most of its evaluation in the compressed domain. CVF exhibits three-fold and two-fold higher accuracy in filtering out inactive frames compared to FilterForward and NoScope, respectively. Further investigation will be conducted in subsequent experiments.

Figure 7a-7c demonstrates the accuracy of CVF compared to the baseline methods in filtering out frames without any interesting activities. The error bars in the figures represent the standard error of the mean for accuracy. We annotated frames demonstrating the movement of objects of interest as relevant frames, while the remainder were deemed unattractive frames. Frames exhibiting minimal or no movement in the objects of interest, or frames devoid of such objects, were classified as unattractive frames. The accuracy of frame filtration is calculated based on the fraction of unattractive frames and frames with interesting activities.

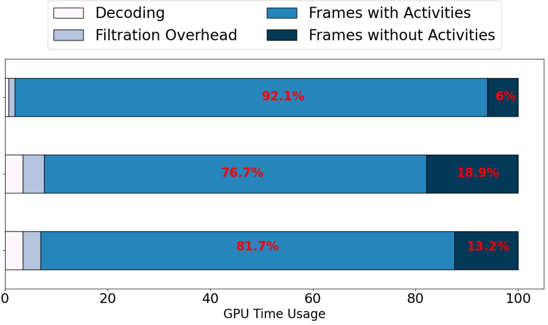
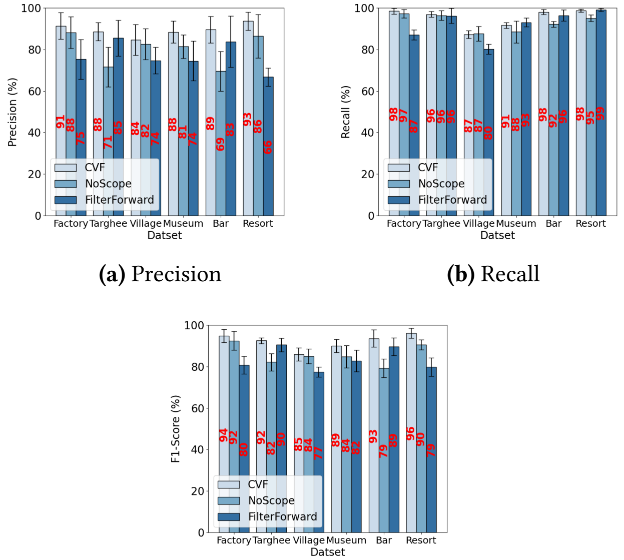


Figure 6. Average fraction of GPU usage for different parts of filtration and video analytics pipeline in different video filtration methods where there is a single stream video feed.

Figure 7c shows that CVF provides better precision compared to both baseline methods in all videos in the dataset. NoScope performed well when temporal differences mostly caused a change in the activities within the frame. Otherwise, it performs poorly if temporal differences within frames are caused by changes in other parts of the scene that do not contain any objects of interest (See an example in Figure 8). Environmental conditions, time, and location state all contribute to this phenomenon. Factors like wind or lighting changes can lead to unattractive temporal alterations in frames. This indicates that solely relying on frame differences to filter out unattractive frames is not feasible.

Figure 8a shows an example scene from the `Bar` video, where a surveillance camera is placed in a corner of a street in front of a bar. There is a sidewalk along a street as well as several palm trees, two fans, and many more objects. This video was recorded on a stormy day that caused the shaking of palm trees and other light objects. Also, cars temporarily and often enter the camera’s view when crossing the street. The fans were turned on and spinning when the video was shot. All these lead to frequent and varying temporal changes in the frames that are not because of a change in an object of interest. Therefore, NoScope had the worst precision for `Targhee` or `Bar` videos compared to other methods.

Figure 8a shows another example scene from the `Resort` video in the dataset where there is a pool and people are relaxing around it. As a result, people do not move in most of the video frames. This becomes a bottleneck if filtration counts completely on binary object classifiers in such situations where objects of interest, i.e. person, are present within the frame but do not have any activities. This bottleneck leads FilterForward to have the worst precision for `Village` and `Resort` videos compared to other methods.

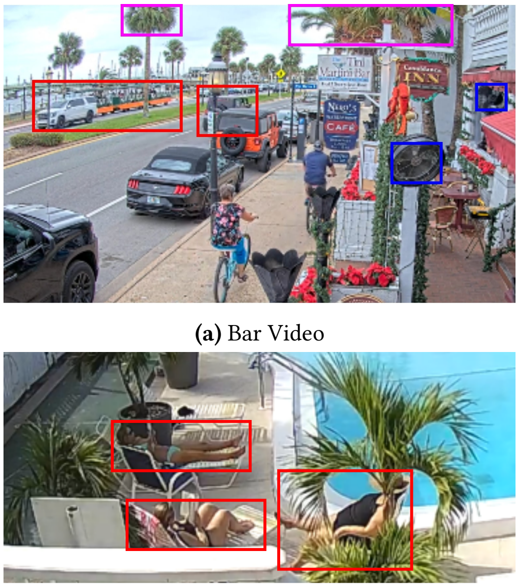


(a) Precision

(b) Recall

(c) F1-Score

Figure 7. Performance of different video filtration methods in accuracy on different videos in the dataset.



(a) Bar Video

(b) Resort Video

Figure 8. (a) Failure of methods relying on temporal changes with background subtraction; (b) Failure of methods relying on binary classifiers.

Figure 7b depicts that NoScope does not have good performance in recall metric because it missed a larger fraction of frames with activities for processing by the pipeline, i.e. false negatives. In more dynamic videos, NoScope exhibits poorer performance as it struggles with the challenge of using a well-tuned threshold to filter out frames lacking activities (as demonstrated in Figure 8a). FilterForward struggles to effectively filter out frames containing static objects of interest, as illustrated in Figure 8b. CVF provides the lowest false negatives and the best values for recall since it uses both temporal difference and a binary object classifier for filtration. Also, Figure 7c demonstrates that CVF's F1-score outperforms baseline filtration methods in accuracy. In contrast to baseline filtration methods, CVF manages to filter out frames that are not involved in objects of interest accurately by performing a light-weight compress-domain mechanism without the need for full frame decoding.

4.3 Cross-Stream Filtration

In this section, we perform a series of experiments where a GPU-enabled edge server receives frames from multiple streaming video feeds. We test with a varying number of streams, for runs with more than six streams, we reuse some of the videos, time-shifted. These experiments aim to study the performance of CVF for filtering out unattractive frames where there are incoming frames from cross-videos and the compute capacity on the edge server is constrained. Thus, it is important not only to filter the video based on the content but also based on the limited computing capacity of the edge.

Figure 9a-9d illustrates different performance criteria with multiple streams of videos processed on the same edge server and no frame filtration. Figure 9a shows that throughput increases with the number of streams of video crossed on the same edge server. If the compute capacity exceeds available resources, throughput drops significantly (i.e., with 16 or more stream feeds). Additionally, Figure 9b shows that processing an uncontrolled number of frames on an edge server can result in longer response times and delayed frames. Figure 9c illustrates that longer response times cause latency violations when there are eight or more videos on an edge server with a V100 GPU and no frame filtration is performed. For instance, Figure 9b shows that over 80% of frames are delayed when the edge servers receive frames from 16-stream video feeds with no filtration.

Figure 10a-10d evaluates the use of concurrent container instances of the video analytics pipeline executing in parallel utilizing the Nvidia MPS feature which enables GPU sharing among different applications. The containers are referred to as compute instances. This figure represents the average performance of different compute instances on the edge server over a different number of streams of video. The first observation is that the average response time increases with the increase in the number of compute instances from 6 to 9. When there are 7 compute instances, the best response time and throughput are achieved. Similar to Figure 9a, throughput increases when we increase the number of compute instances from 6 to 7. However, throughput drops significantly when there are eight or, even worse, nine compute instances on the edge server. This is because the edge server is overutilized. In general, 7 compute instances ensure the highest throughput, and CVF selects it for edge servers with V100 GPUs, while this number of instances also largely guarantees a response time within the latency constraint.

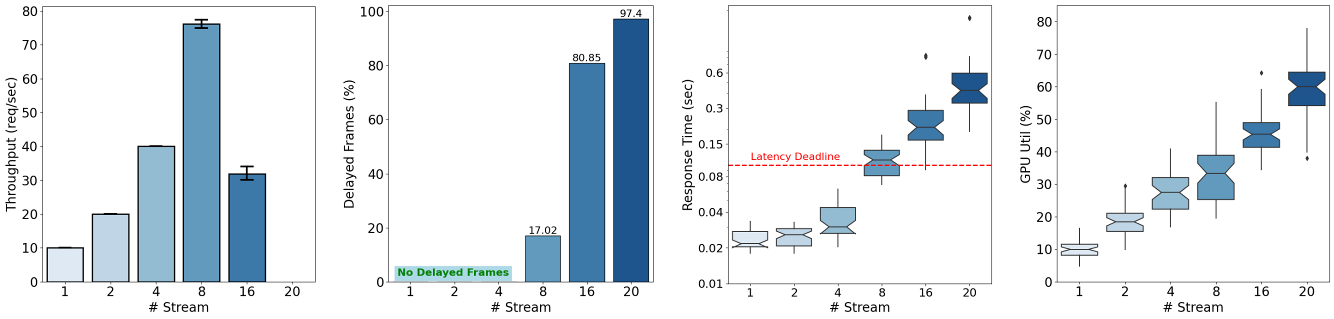
Figures 11a-11d depict the execution time of different parts of the pipeline for CVF compared to the baseline methods over different numbers of streams of video crossed on the same edge server. These figures reveal that the processing overhead of the baseline filtration pipelines is higher in contrast to CVF when multiple video streams traverse an edge server. This disparity arises because baseline filtration methods cannot filter frames based on available compute resources. Consequently, it leads to unexpectedly prolonged response times due to over-utilization. Furthermore, all modules in baseline filtration pipelines operate in the pixel domain, requiring complete decoding of incoming frames and processing using heavier pixel-domain components. CVF outperforms baseline methods in response time for processing multiple video streams at the edge.

Figure 12a shows the throughput across different numbers of video streams as the input workload for an edge server. The diagram demonstrates that CVF achieves the highest throughput compared to baseline methods as the number of streaming video feeds increases. However, in baseline filtration pipelines, throughput significantly decreases with more than twelve video feeds multiplexed as input workloads on an edge server. Similarly to the observations in Figure 7c, Figure 12b demonstrates the superior accuracy of CVF compared to the other two baseline filtration methods.

Figure 12c illustrates the fraction of filtered-out frames and the fraction of delayed frames over different numbers of streams on an edge server. The first finding is that the baseline methods filter out frames regardless of the number of receiving frames (i.e., the number of stream video feeds) in the edge server, and as a result, NoScope and FilterForward methods filter out almost the same percentage of frames in different experiments on average. On the other hand, CVF method filters out a larger fraction of incoming frames when it receives more frames, depending on resource availability. Therefore, the percentage of delayed frames dramatically increased with the increase in the number of stream video feeds, while the percentage of delayed frames increased smoothly when the edge server received frames from a larger number of stream video feeds.

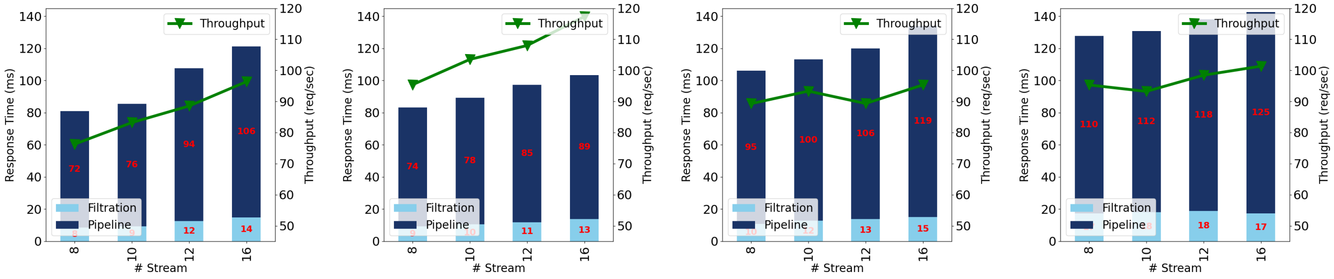
4.4 Filtration in Different Pipelines

This section aims to evaluate the effectiveness of the CVF pipeline in filtering frames lacking significant activity and eliminating excessive frames for more complex video pipelines with multiple components. We utilize the Amber Alert video pipeline, consisting of decoding, preprocessing, DNN object detection, face detection, and car detection components.



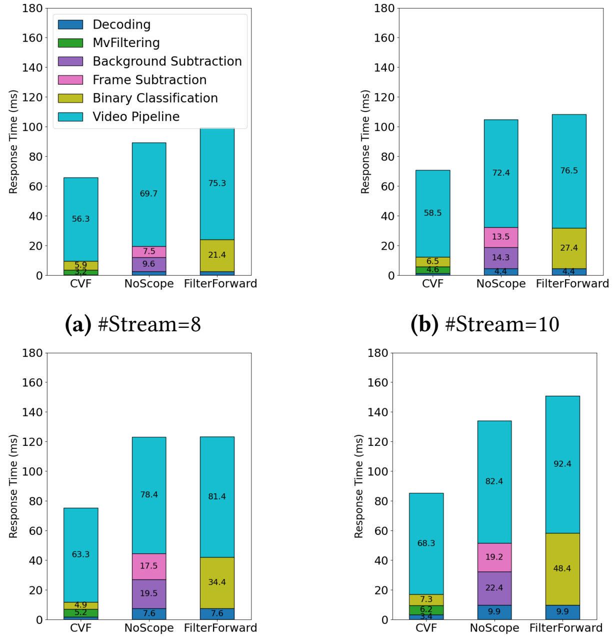
(a) Throughput **(b)** Frame Loss **(c)** Response Time **(d)** GPU Util

Figure 9. Evaluation of the processing different numbers of video feeds on a V100 GPU without frame filtration.



(a) Compute Resources=6 **(b)** Compute Resources=7 **(c)** Compute Resources=8 **(d)** Compute Resources=9

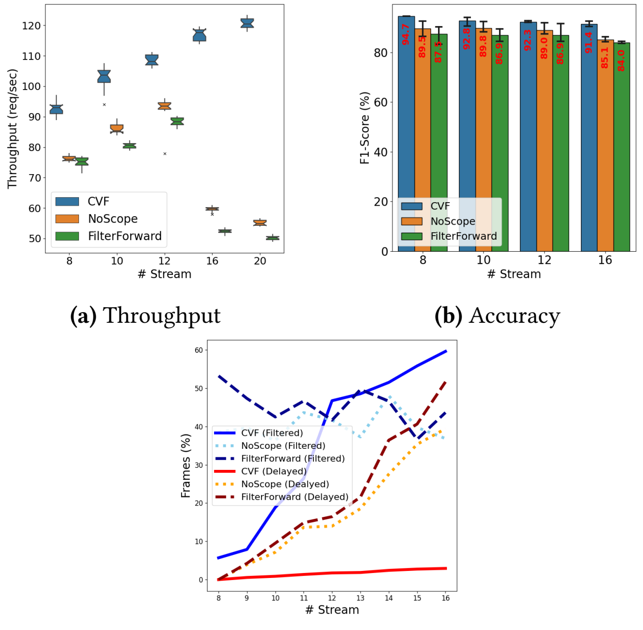
Figure 10. Performance evaluation with varying compute instances on response time and throughput.



(a) #Stream=8 **(b)** #Stream=10

(c) #Stream=12 **(d)** #Stream=16

Figure 11. Stacked plot comparing response time and latency of different components of the proposed and baseline filtration methods over different numbers of streams.



(a) Throughput **(b)** Accuracy

(c) Delayed and Filtered

Figure 12. Evaluation of proposed and baseline filtration methods on Tesla V100 GPU with varying stream numbers.

Figure 13a illustrates the overall throughput on an edge server equipped with a V100 GPU for the proposed and baseline filtration methods. The graph demonstrates a consistent increase in throughput when employing CVF as the filtration pipeline, particularly under heavier workloads from concurrent video feeds. Conversely, NoScope and FilterForward experience a notable decline in overall throughput with 20

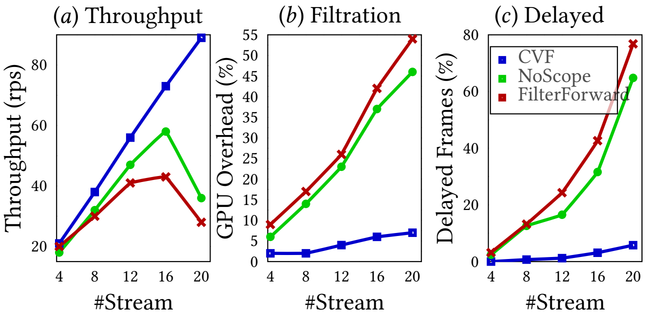


Figure 13. Evaluating the Amber Alert pipeline across varying numbers of video feeds.

stream video feeds, showing only marginal improvement when handling 16 stream video feeds.

Figure 13b evaluates the computational load on the GPU imposed by different filtration pipelines when handling varying numbers of stream video feeds. It demonstrates that the baseline methods, NoScope and FilterForward, impose significantly higher GPU loads compared to CVF. The load increases linearly with more concurrent streams, revealing scalability limitations in the baseline methods.

The inefficiency of the baseline approaches stems from their need to decode all incoming frames and employ compute-intensive methods, such as binary classification or heavy object detection, across every frame in the pixel domain. Conversely, CVF utilizes a lightweight mechanism in the compression domain, bypassing the necessity to decode all frames and conducting pixel-domain evaluation solely on frames relevant to the GPU compute capacity. As a result, the computation imposed by the CVF filtration pipeline is minimally affected by the incoming frame rate, focusing on additional computation in the compression domain and evaluating motion vectors of frames.

The increased GPU overhead for frame filtration in the baselines renders them impractical for scalability, resulting in lower throughput and more delayed frames (See Figure 13c).

5 Related work

To reduce the computational load on GPUs, many approaches have been proposed in the literature. One such method is to use compressed DNN models for video analytics and image processing with different depths and input sizes compared to the original model. The compact versions require less computation at the expense of accuracy [39, 40].

In addition, there are a few proposed techniques for video filtration. Hu et al. presented a filtering system for video analysis applications based on user-defined queries [41]. For instance, the user defines "a person approaches a car" as a filtering query. The main applications of user-defined query filtering methods are video data storage and querying. Li et al. [5] presented a real-time on-camera frame filtering method for video analytics called Reducto. Reducto enables adaptive filtering decisions according to video content and query accuracy. It queries by utilizing previously unused resources to perform on-camera frame filtering. However, most cameras today cannot run these filtering algorithms limiting the usability of these approaches. In addition, it requires that each camera is configured with any new changes to the context of the application. Finally, on-camera filtering does not keep the filtered frames for offline analysis.

Another technique used for filtering is the usage of binary classifiers. Canel et al. presented FilterForward [6] as an edge-based filtration method that extracts features and finds relevant frames for processing using a special type of binary classifier. Tchaye-Kondi et al. presented, SmartFilter, an edge-to-cloud filtering method for video analysis [26]. SmartFilter uses a fast and light-weight binary classifier to filter out frames without enough activities. Kang et al. introduced NoScope [7], an inference-optimized querying system to reduce the computation cost of video analysis, Given an input video, a target object, and a reference network, Noscope searches for a cascade of specialized models for object detection and difference detectors for filtering out irrelevant frames. NoScope achieves better speed-ups compared to binary classifications for frame filtering.

6 Conclusion

In this paper, we introduce CVF, a cross-video filtration pipeline for saftey critical systems such as factory automation. CVF filters frames across video feeds from hundreds of cameras efficiently, reducing the overall computational load of edge video analytics with heavy DNNs. CVF facilitates lightweight filtration solutions by analyzing frames within the compressed domain through partial frame decoding. CVF prioritizes frames and feeds based on activity in the frames detected via MVs from the compressed domain, based on the presence of objects of interest, and based on the time since the video stream has been last considered. Our results show that compared to state-of-the-art filtration techniques, CVF can reduce the overall response time of the stream processing by almost 50% and increase the throughput by up to 120% compared to the state-of-the-art while maintaining the accuracy and precision of the analytics framework. To the best of our knowledge, there are no attempts to filter out irrelevant frames based on resource availability when the video analysis pipeline receives frames from several cameras.

Acknowledgments. Ali Rahmanian is partially supported by Ericsson Research and the Industrial Doctoral School at Umeå University. Ahmed Ali-Eldin is supported by an SSF Future Research Leader grant (FFL8).

References

[1] “Surveillance camera statistics: which cities have the most cctv cameras?” 2022. [Online]. Available: https://www.comparitech.com/vpn-privacy/the-worlds-most-surveilled-cities/

[2] A. Rahmanian, A. Ali-Eldin, S. K. Tesfatsion, B. Skubic, H. Gustafsson, P. Shenoy, and E. Elmroth, "Ravas: Interference-aware model selection and resource allocation for live edge video analytics," in *2023 IEEE/ACM Symposium on Edge Computing (SEC)*, 2023, pp. 27–39.

[3] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *computer*, vol. 50, no. 10, pp. 58–67, 2017.

[4] Z. Fang, D. Hong, and R. K. Gupta, "Serving deep neural networks at the cloud edge for vision applications on mobile platforms," in *Proceedings of the 10th ACM Multimedia Systems Conference*, 2019, pp. 36–47.

[5] 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 *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 359–376. [Online]. Available: https://doi.org/10.1145/3387514.3405874

[6] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. Dulloor, "Scaling video analytics on constrained edge nodes," in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 406–417. [Online]. Available: https://proceedings.mlsys.org/paper/2019/file/85d8ce590ad8981ca2c8286f79f59954-Paper.pdf

[7] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: Optimizing neural network queries over video at scale," *Proc. VLDB Endow.*, vol. 10, no. 11, p. 1586–1597, aug 2017. [Online]. Available: https://doi.org/10.14778/3137628.3137664

[8] W. Zhang, S. Lin, F. H. Bijarbooneh, H. F. Cheng, and P. Hui, "Cloudar: A cloud-based framework for mobile augmented reality," in *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, 2017, pp. 194–200.

[9] S. Shi, V. Gupta, M. Hwang, and R. Jana, "Mobile vr on edge cloud: a latency-driven design," in *Proceedings of the 10th ACM multimedia systems conference*, 2019, pp. 222–231.

[10] J. Jansen, S. Subramanyam, R. Bouqueau, G. Cernigliaro, M. M. Cabré, F. Pérez, and P. Cesar, "A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency dash," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 341–344.

[11] A. Ali-Eldin, B. Wang, and P. Shenoy, "The hidden cost of the edge: a performance comparison of edge and cloud latencies," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–12.

[12] Q. Liang, P. Shenoy, and D. Irwin, "Ai on the edge: Characterizing ai-based iot applications using specialized edge architectures," in *2020 IEEE International symposium on workload characterization (IISWC)*. IEEE, 2020, pp. 145–156.

[13] C. Xu, S. Jiang, G. Luo, G. Sun, N. An, G. Huang, and X. Liu, "The case for fpga-based edge computing," *IEEE Transactions on Mobile Computing*, 2020.

[14] S. Biookaghazadeh, M. Zhao, and F. Ren, "Are {FPGAs} suitable for edge computing?" in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[15] "Nvidia egx edge ai platform brings real-time ai to manufacturing, retail, telco, healthcare and other industries," https://nvidianews.nvidia.com/news/nvidia-egx-edge-ai-platform-brings-real-time-ai-to-manufacturing-retail-telco-healthcare-and-other-industries.

[16] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 119–135.

[17] H. Guo, S. Yao, Z. Yang, Q. Zhou, and K. Nahrstedt, "Crossroi: cross-camera region of interest optimization for efficient real time video analytics at scale," in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021, pp. 186–199.

[18] M. Zhang, F. Wang, Y. Zhu, J. Liu, and Z. Wang, "Towards cloud-edge collaborative online video analytics with fine-grained serverless pipelines," in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021, pp. 80–93.

[19] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "Coedge: Cooperative dnn inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 595–608, 2020.

[20] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "Ionn: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proceedings of the ACM symposium on cloud computing*, 2018, pp. 401–411.

[21] 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 SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[22] I. I.-T. IEC JTC, "Advanced video coding for generic audiovisual services," *ITU-T Recommendation H. 264*, 2003.

[23] T. Wiegand, "Draft itu-t recommendation and final draft international standard of joint video specification (itu-t rec. h. 264-iso/iec 14496-10 avc)," *JVT-G050*, 2003.

[24] C. Ramer, J. Sessner, M. Scholz, X. Zhang, and J. Franke, "Fusing low-cost sensor data for localization and mapping of automated guided vehicle fleets in indoor applications," in *2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2015, pp. 65–70.

[25] W. You, M. S. H. Sabirin, and M. Kim, "Real-time detection and tracking of multiple objects with partial decoding in H.264/AVC bitstream domain," in *Real-Time Image and Video Processing 2009*, N. Kehtarnavaz and M. F. Carlsohn, Eds., vol. 7244, International Society for Optics and Photonics. SPIE, 2009, p. 72440D. [Online]. Available: https://doi.org/10.1117/12.805596

[26] J. Tchaye-Kondi, Y. Zhai, J. Shen, D. Lu, and L. Zhu, "Smartfilter: An edge system for real-time application-guided video frames filtering," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 772–23 785, 2022.

[27] Google, "Google computing services," 2023. [Online]. Available: https://cloud.google.com/

[28] "Accelerated gstreamer user guide - nvidia developer," 2019. [Online]. Available: https://developer.download.nvidia.com/embedded/L4T/r32_Release_v1.0/Docs/Accelerated_GStreamer_User_Guide.pdf

[29] "Gstreamer udp," https://gstreamer.freedesktop.org/documentation/udp/index.html.

[30] "Mv-extractor," https://github.com/LukasBommes/mv-extractor.

[31] L. Bommes, X. Lin, and J. Zhou, "Mvmed: Fast multi-object tracking in the compressed domain," in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2020, pp. 1419–1424.

[32] "Dekr," https://github.com/HRNet/DEKR.

[33] H. Shen, L. Chen, Y. Jin, L. Zhao, B. Kong, M. Philipose, A. Krishnamurthy, and R. Sundaram, "Nexus: A gpu cluster engine for accelerating dnn-based video analysis," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, ser. SOSP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 322–337. [Online]. Available: https://doi.org/10.1145/3341301.3359658

[34] "Grand targhee resort plaza," https://www.youtube.com/watch?v=eHL_FJOoVTE.

[35] "Santa claus village," https://www.youtube.com/watch?v=Cp4RRAEgpeU.

[36] "Strawbery banke museum," https://www.youtube.com/watch?v=g5xfCRdBhaA.

[37] "Tini martini bar st. augustine fl," https://www.youtube.com/watch?v=NlmLu4C66n8.

[38] "Emerald beach resort north pool panama city beach, fl," https://www.youtube.com/watch?v=S6_NatR64uI.

[39] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "Yolov6: A single-stage object detection framework for industrial applications," 2022. [Online]. Available: https://arxiv.org/abs/2209.02976

[40] H. Law, Y. Teng, O. Russakovsky, and J. Deng, "Cornernet-lite: Efficient keypoint based object detection," 2019. [Online]. Available: https://arxiv.org/abs/1904.08900

[41] Z. Hu, N. Ye, C. Phillips, T. Capes, and I. Mohomed, "Mmfilter: Language-guided video analytics at the edge," in *Proceedings of the 21st International Middleware Conference Industrial Track*, ser. Middleware '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–7. [Online]. Available: https://doi.org/10.1145/3429357.3430518