CoVA: Exploiting Compressed-Domain Analysis to Accelerate Video Analytics

Jinwoo Hwang, Minsu Kim, Daeun Kim, Seungho Nam, Yoonsung Kim, and Dohee Kim, *KAIST;* Hardik Sharma, *Google;* Jongse Park, *KAIST*

https://www.usenix.org/conference/atc22/presentation/hwang

This paper is included in the Proceedings of the 2022 USENIX Annual Technical Conference.

July 11–13, 2022 • Carlsbad, CA, USA

978-1-939133-29-8

Open access to the Proceedings of the 2022 USENIX Annual Technical Conference is sponsored by

CoVA: Exploiting Compressed-Domain Analysis to Accelerate Video Analytics

Jinwoo Hwang *KAIST*
Minsu Kim *KAIST*
Daeun Kim *KAIST*
Seungho Nam *KAIST*
Yoonsung Kim *KAIST*
Dohee Kim *KAIST*
Hardik Sharma *Google*
Jongse Park *KAIST*

Abstract

Modern retrospective analytics systems leverage cascade architecture to mitigate bottleneck for computing deep neural networks (DNNs). However, the existing cascades suffer from two limitations: (1) decoding bottleneck is either neglected or circumvented, paying significant compute and storage cost for pre-processing; and (2) the systems are specialized for temporal queries and lack spatial query support. This paper presents CoVA, a novel cascade architecture that splits the cascade computation between compressed domain and pixel domain to address the decoding bottleneck, supporting both temporal and spatial queries. CoVA cascades analysis into three major stages where the first two stages are performed in compressed domain, while the last one in pixel domain. First, CoVA detects occurrences of moving objects (called *blobs*) over a set of compressed frames (called *tracks*). Then, using the track results, CoVA prudently selects a minimal set of frames to obtain the label information and only decode them to compute the full DNNs, alleviating the decoding bottleneck. Lastly, CoVA associates tracks with labels to produce the final analysis results on which users can process both temporal and spatial queries. Our experiments demonstrate that CoVA offers 4.8× throughput improvement over modern cascade systems, while imposing modest accuracy loss.

1 Introduction

Every day, a massive corpus of video data is produced, which is only growing (9.4 exabytes per day, as of 2021 [1]). Extracting insights and actionable semantics from the captured video can enable a variety of applications in healthcare, smart cities, security, customer behavior analysis, etc. Prior works [2–7] have built *retrospective analytics systems* that allow analysts to interactively query over a large corpus of accumulated video data stored in disk.

Modern retrospective analytics heavily rely on deep neural networks (DNNs). Although DNNs are effective, they come at the cost of significant compute complexity, even for an image. Evidently, passing all the frames of a video through DNN inferencing is computationally prohibitive. To address this challenge, recent works [2–4, 6–13] have focused on *cascade* architectures. They stage processing as (relatively) inexpensive predicates to filter the incoming frames of video by trading analysis accuracy for higher throughput. As such, only a handful of frames arrive at the last stage that performs the full DNN inferencing.

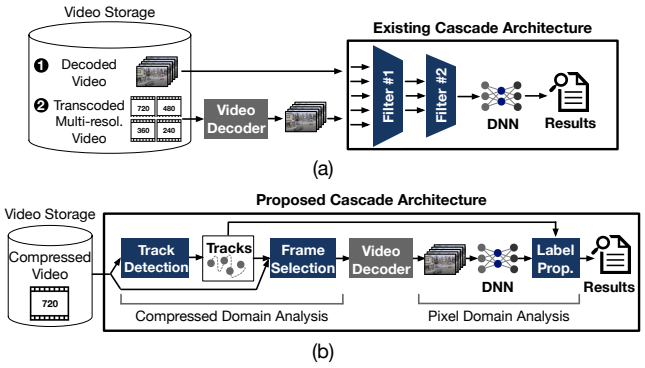

Figure 1: (a) Existing state-of-the-art cascade systems [2, 3], excluding video decoding from the end-to-end setting with two costly assumptions; (b) the proposed cascade architecture that addresses the decoding bottleneck and supports spatial queries, exploiting the compressed domain analysis.

While effectively resolving the DNN throughput bottleneck, the existing cascade systems have two limitations. First, as shown in Figure 1(a), these systems either ignore or sidestep a new bottleneck stage, *video decoding*, by making one of the two costly assumptions: (1) input video is decoded a priori and the raw frames are stored in storage [2, 3, 5, 7], or (2) input video is pre-transcoded and stored in multiple lower resolutions at ingest time to facilitate the query time decoding [4, 6]. However, in practice, decoding (or transcoding) the entire video corpus and storing the uncompressed (or

duplicate) data in disk is often infeasible due to the significant compute and storage cost.

Second, to achieve otherwise-unachievable throughput, the existing cascade systems often exclusively support *temporal* queries. More specifically, many cascade systems [2, 3, 5, 11] only support binary predicate query, which is to get timestamps of frames that contain the queried object. However, recent studies in video analytics [7, 15] propose *spatial* queries (e.g., car in upper right region) and demonstrate their usefulness, which cannot be supported by the current cascades.

To tackle the two limitations, this paper sets out to devise CoVA[1], an alternate cascade architecture. As illustrated in Figure 1(b), the key contribution of CoVA is to split cascade computation between compressed domain and uncompressed pixel domain, which collaboratively alleviate the decoding bottleneck at query time without requiring any pre-processing and support both temporal and spatial queries. To design this cascade architecture, we leverage the following two insights:

(1) A small set of encoding metadata, commonly used by modern video codecs, provides noisy, yet rich, information to accurately locate potential objects and track them across frames in compressed video, while decoded pixel data is only necessary to classify objects.

(2) Video analytics queries can be fulfilled by answering the following three questions: (1) where and when are interesting objects present in the video (i.e., spatiotemporal information); (2) what are the object classes (i.e., label information); and (3) what specific information do queries ask about these objects?

With these insights, CoVA divides video analytics over compressed footage into three major stages. The first stage (**Track Detection**) detects occurrences of moving objects (called *blobs*) over a collection of consecutive compressed frames (called *tracks*). To realize this objective, we devise a novel compressed-domain blob tracking technique, refitting a neural network based segmentation algorithm and a multiple object tracking algorithm, both of which are originally designed for pixel domain. Our second stage (**Frame Selection**) avoids decoding the whole track and selects a minimal set of frames that are representative and yet minimize the decoding load. CoVA passes only this subset through the full DNN object detection. The third stage (**Label Propagation**) takes the labels and the coordinates of the detected objects in the subset and uses spatiotemporal information from the first stage to propagate labels across all the frames of the track. Altogether, these approaches offer a novel cascade architecture that performs its first and second stages in the compressed domain, while the third stage is in the pixel domain.

Finally, the three stages produce a collection of analysis results for each frame, which include a list of present objects, their pixel coordinates, their labels (e.g., car), and all other properties associated with the objects (e.g., color). Note that the results are query-agnostic and not specific to a certain query. Therefore, CoVA runs the three stages only for the initial query and stores the analysis results along with the video in database. When other queries are requested over the same video in a future, CoVA simply retrieves the results and process the queries without reprocessing the video.

We prototype a CoVA system[2] on NVIDIA's streaming analytics framework, DeepStream [16]. We evaluate the effectiveness of CoVA using five video streams and four queries. Compared to existing cascade systems for query time retrospective analytics, CoVA offers 4.8× throughput improvement, while compromising only modest accuracy loss. We also show that CoVA is capable of serving spatial queries without having significant accuracy loss, compared to the full DNN analytics baseline.

Contributions. Our key contributions are as follows:

- We show that encoding metadata is sufficiently rich to identify objects of interest along with their spatiotemporal information for retrospective video analytics.
- To extract the spatiotemporal information, we devise a novel compressed-domain blob tracking technique, refitting the pixel-domain video segmentation and object tracking algorithms.
- We present the design of CoVA, a mixed-domain retrospective analytics system that leverages the track information to alleviate the decoding bottleneck, and support both temporal and spatial queries.
- Our experiment shows that CoVA offers significant throughput improvement over conventional cascade systems, while compromising modest accuracy loss.

2 Background and Motivation

CoVA aims to tackle limitations of existing retrospective analytics systems. Below, we first provide background on state-of-the-art retrospective analytics and discuss their limitations. We also discuss common compression mechanisms of modern video codecs, which drive the design of proposed techniques.

2.1 Retrospective Analytics

Modern retrospective analytics systems [2–8, 10–14] share two common properties: (1) heavy reliance on DNNs, and (2) cascade architecture to resolve the DNN compute bottleneck.

[1] **CoVA**: **Co**mpressed **V**ideo **A**nalytics.

[2] Our prototype is available at https://github.com/casys-kaist/CoVA.

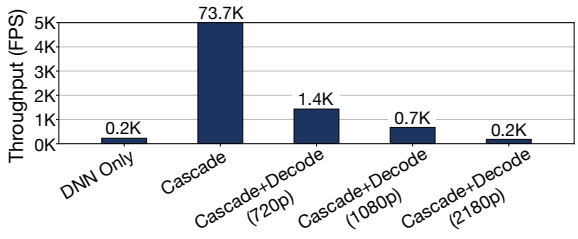

Figure 2: Throughput comparison among various system environments of cascade video analytics.

While they have these common properties, there are two different dimensions that categorize the instances of retrospective analytics systems.

Time of analysis – query time vs. ingest time. Retrospective analytics systems are categorized into two groups, depending on whether the analysis occurs at *query time* [2, 3, 11] or *ingest time* [4, 6, 8, 13]. While ingest time analysis leverages offline pre-processing to facilitate and expedite the query time analysis, it requires to scan the entire video data corpus and consume compute resources on it, even though a significant portion of the data is not queried. This approach is not only cost-inefficient but also environmentally suboptimal since it would consume a massive amount of energy for mostly unnecessary computations. In contrast, query time analysis performs the full analysis at query time without having any pre-processing. Therefore, it does not touch raw video data unless it is queried, which allows analysts to prevent the waste of resources. To this end, this work focuses on the query time analysis and aims to address its limitations.

Supported query – temporal vs. both temporal and spatial. Most, if not all, of query time cascade systems [2, 3, 11] limit the types of supported queries to be only the *temporal* ones and specialize the cascade stages for a specific temporal query to achieve high throughput. However, recent work [7] points out that *spatial* information can enable richer capabilities for video analytics. CoVA is a novel cascade architecture that leverages compressed-domain analysis to address both spatial and temporal queries.

2.2 Video Decoding: the New Bottleneck

Decoding for end-to-end cascade. With the volume of video data growing at an explosive rate, the use of compression is imperative to keep storage costs in check. Video codecs such as H.264 strike a balance between quality and storage size, being used as the de facto way of storing large corpus of video data. As such, the first step in an end-to-end system for processing video queries is to decode the video data before further processing. However, even with hardware-acceleration for standard codecs baked-in to modern CPUs and GPUs, video decoding can be up to orders-of-magnitude slower than the capabilities of cascade systems to process raw video frames.

Bottleneck analysis. To quantify this bottleneck, we examine the performance impact of video decoding for an existing state-of-the-art cascade system, Tahoma [3], using NVIDIA RTX 3090 GPU, and present the results in Figure 2. The detailed methodology is provided in Section 8.1. The cascade system is effective in addressing the DNN-execution bottleneck and offers up to 327× improvement in performance compared to a native DNN-only solution. However, even with decoding accelerator hardware NVDEC [17], the decoding throughput is significantly lower than the throughput of cascade system, which curtails most performance gains.

Further, as video resolution increases, the decoding throughput almost linearly decreases, exacerbating the decoding bottleneck. Considering the trend that even IoT devices such as surveillance cameras produce HD (1080p) or higher resolution video, we believe that this decoding bottleneck will become increasingly severe and significantly hinder the usefulness of video analytics in interactive applications. Motivated by these insights, the objective of CoVA cascade is to address the decoding bottleneck in query time retrospective analytics.

2.3 Block-based Video Coding

To alleviate the decoding bottleneck, CoVA leverages the unique characteristics of *block-based* compression, used in many modern video codecs. Below, we provide background on block-based compression and discuss opportunities that it opens for compressed-domain analysis.

Video codecs. Many video codecs, such as H.264, HEVC, VP8, VP9, and AV1, use block-based compression algorithm. In this paper, we primarily focus on the H.264 format since it is one of the most widely used codecs in various applications as of publication date [18]. However, CoVA is compatible with other block-based codecs since all of them compress video, generating the same set of metadata we use for compressed-domain analysis in CoVA.

Block-based compression. Block-based codecs compress (or encode) video frames by splitting each frame into a two-dimensional array of fixed sized blocks, called *macroblocks* (e.g., 16x16 pixels). There are three *macroblock types* – I, P, and B – depending on the way how the macroblocks are compressed. An I-macroblock is independently compressed, while P- and B-macroblocks are compressed referring to one and two other macroblocks, respectively. To maximize compression ratio, the codecs select dependent macroblocks for P and B-macroblocks with the highest similarity and store the spatial offsets as metadata called *motion vectors*. Depending

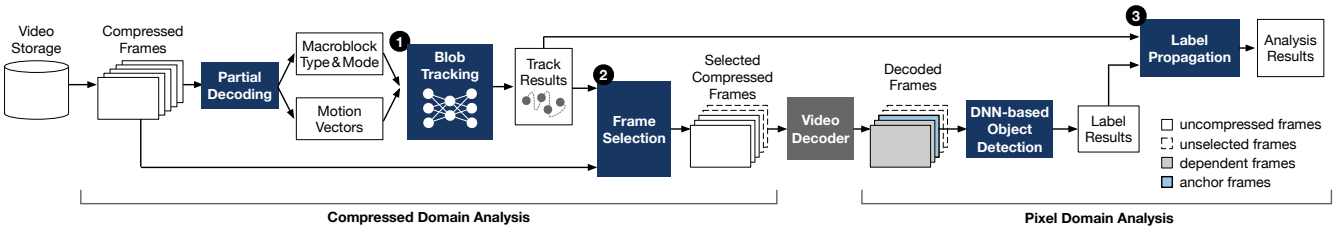

Figure 3: Overview of CoVA.

on the composition of macroblocks, frames are again categorized into three types, I, P, and B. An I-frame, also known as a keyframe, is only composed of I-macroblocks, while a P-frame contains I/P-macroblocks and a B-frame has all of the I/P/B-macroblocks.

To maximize the compression rate, codecs can *partition* macroblocks (e.g., 16x16) into smaller sub-macroblocks (e.g., 4x4). This optimization allows codecs to achieve a higher compression rate but at the expense of storing larger metadata. Modern codecs employ multiple *macroblock partitioning modes*. For instance, H.264 uses six modes from no partitioning (i.e., 1 macroblock of size (16×16)) to 16-way partitioning (i.e., 16 sub-macroblocks of size (4×4)).

CoVA leverages the insight that the encoding metadata – (1) macroblock types, (2) motion vectors, and (3) macroblock partitioning modes – in the compressed video is sufficiently rich to detect potential objects and track them across frames.

Compression rate optimization. Due to the higher compressibility, codecs tend to prefer P/B-macroblocks over I-macroblock. However, the preference for P/B macroblocks ends up creating long dependency chains among the macroblocks, which cause compression errors to propagate across the chains and hinder random access to frames in the video. To resolve the problems, the codecs insert I-frames at regular intervals, typically every 250 frames, to create independent sets of consecutive video frames, called Groups of Pictures (GoP). Within a GoP, the number of dependent frames that need to be decoded grows linearly, with zero for the first I-frame and maximum for the last frame.

CoVA exploits the inter-frame dependencies and object track information extracted from compressed-domain analysis to prudently select the frames with the least number of dependencies in each GoP that enable to identify all the objects present and minimize decoding effort.

3 Overview of CoVA

CoVA divides video analytics over compressed footage into three major stages, as illustrated in Figure 3.

❶ **First Stage: Track Detection.** First, CoVA detects occurrences of moving objects over a collection of consecutive compressed frames, which we call tracks. The track detection stage further breaks down into two steps: (1) *blob detection*: CoVA *spatially* detects whether and where moving objects (called blobs) are present in each compressed frame; and (2) *blob tracking*: CoVA *temporally* associates the blobs across frames to identify unique blob tracks. For the blob detection, we devise a novel compressed-domain blob detection model, refitting a neural network architecture originally designed for pixel-domain video segmentation. The neural network only takes as input three encoding metadata commonly used by modern codecs, recognizes movements as masks, and spatially associates the masks clustered in a region as blobs. While the neural network architecture is fixed, CoVA trains the model individually for each video to learn the data-specific patterns of blobs and specialize for the target video. Finally, the found blobs are fed into the blob tracking step that employs an object tracking algorithm, SORT [19], which was also originally developed for pixel domain. Note that the blob track results still lack the object class labels.

❷ **Second Stage: Frame Selection.** To attain the object labels for the detected blobs, CoVA needs to perform DNN-based object detection for the frames where tracks appear, which ordinarily require decoding all the frames. However, as frames on a track most likely contain the same object, it is enough to perform the object detection on a subset of the frames in the track, which we call *anchor frames*. Thus, CoVA only decodes frames required to decode the anchor frames, which improves the effective decoding throughput. The challenge is how to prudently select the anchor frames so as to minimize the decoding cost and at the same time acquire the accurate label information. We develop a frame selection algorithm that leverages a common property of video codecs where compressed frames are encoded in dependency chains. Thus, anchor frames are the ones that are located on the max-

imal number of tracks and at the same time have the short dependency chain with respect to the decoding algorithm. Note that while the anchor frames are the only ones that are inferred upon for object detection, all the frames in the track need to be labeled to handle various video analytics queries.

❸ **Third Stage: Label Propagation.** In the third stage, CoVA takes the approximate positions of potential objects (or blobs) from the first stage and labels for the anchor frames from the second stage to temporally propagate the labels across all the frames of the tracks. To merge the spatial and temporal results, CoVA first spatially correlates blobs with objects on anchor frames using the intersection ratios of their bounding boxes. Then, CoVA uses the tracking information to identify the same objects across the frames and propagates the labels, while populating bounding boxes around the corresponding blobs in the temporally consecutive frames.

Finally, when a video passes through the three stages, CoVA produces a collection of analysis results for each frame, the examples of which are a list of present objects, their pixel coordinates, their labels (e.g., car), and all other properties associated with the objects (e.g., color). Note that the results are created only once when CoVA receives the initial query over a video and they are permanently associated with the video in the database. After then, analysts can use the same results to process various future queries without reprocessing the video.

4 Compressed Domain Blob Tracking

In this section, we describe the track detection mechanism that is the first stage of CoVA's cascade architecture. Figure 4 depicts the overall workflow.

4.1 Learning to Detect Blobs

Limitations of existing compressed domain video processing techniques. Detecting objects or blobs from compressed video is a traditional research problem in the computer vision community [20–25]. However, the following two limitations prevent the simple adoption of these techniques. First, the techniques often require human-crafted parameters that need to be tuned for each input video, which makes automated analytics impossible. Secondly, the techniques are not sufficiently robust to be applied to arbitrary video data, producing inadequately accurate tracking results for video analytics. To overcome such limitations, recent works [26, 27] explored to use neural networks for vision tasks over compressed video. Unfortunately, we could not employ the neural networks for CoVA since they not only still require pixel-domain data for a subset of frames, but also offer insufficient throughput that is significantly lower than the decoder.

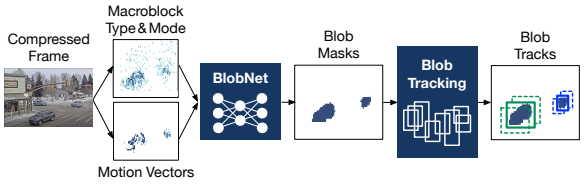

Figure 4: Track detection.

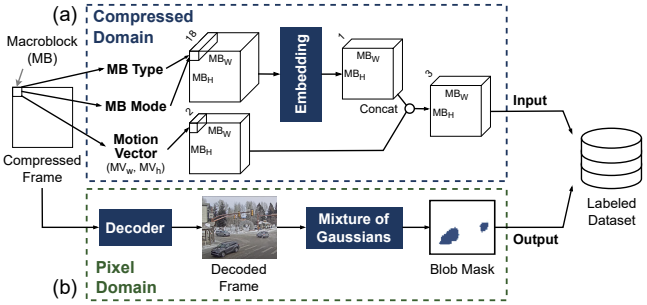

Figure 5: (a) Feature engineering that transforms three compression metadata into a tensor of input features; (b) labeled data collection using the Mixture of Gaussians (MoG) model.

Leveraging the similarity between video segmentation and blob detection. To address these limitations, we exploit an observation that blob detection using compression metadata is akin to the problem of the semantic image segmentation using pixel data. *Blob detection* task aims to find potential objects and their approximate position within video frames. *Image (or video) segmentation* task, on the other hand, aims to semantically split an image (or frames of a video) and classify each segment into one of the predetermined labels. When there are only two classes – blob and non-blob – the image segmentation task can be reduced to the approximate blob detection task. This observation allows us to tap into the vast range of techniques, including Deep Neural Network (DNN) based image and video segmentation, that can be geared towards compressed domain blob detection.

4.2 BlobNet

To this end, we devise a lightweight DNN-based blob detection model, called BlobNet, building upon the state-of-the-art Temp-UNet [28] model for video segmentation. Unlike the Temp-UNet model, which operates on pixel frames, BlobNet operates on compression metadata.

Feature engineering. Figure 5(a) depicts the feature engineering, which converts the three metadata into a tensor of input features. BlobNet takes the three types of metadata as input – macroblock types, macroblock partitioning modes, and motion vectors. To obtain the metadata, CoVA performs

only a few early stages of the decoding process required to extract metadata, called *partial decoding*. CoVA encodes the first two metadata, macroblock types and partitioning modes, by mapping each of their combinations into an one-hot vector (e.g., total 12 combinations for H.264). These one-hot vectors are fed into an embedding layer, which converts each one-hot vector into a scalar weight value. This weight value is concatenated to the motion vector (MV_w, MV_h) for each macroblock, which finally results in a 3D tensor ($MB_W \times MB_H \times 3$). CoVA temporally stacks these tensors from consecutive frames and constructs a 4D tensor, which is the input for BlobNet.

BlobNet architecture. Similar to the architecture of Temp-UNet[3], BlobNet has three major components: (1) **encoder** that extracts the presence and approximate location of blobs from noisy metadata; (2) **decoder** that reconstructs the shapes of blobs from the blob presences; (3) **skip connections** that offer spatial information to the decoder for assisting the shape reconstruction process. While this overall composition is the same as that of Temp-UNet architecture, we maximally reduce the depth of encoder and decoder modules such that the resulting model still offers high accuracy while maximizing the inference throughput.

Video-specialized model training. Pixel video segmentation models typically train once during a training phase, followed by inference on unseen video data. However, CoVA trains BlobNet at query time for every video data to specialize the model for the specific data. This design choice is derived from our empirical observation that without such model specialization, the model cannot capture the variations of data-specific encoding parameters and fails to reach sufficient accuracy. Note that once training is completed for a video data, no further training is required for additional video if the video is recorded from the same angle of view with the trained one. We empirically observe that $\approx 3\%$ of the video is sufficient to train the model for the evaluated video (see Table 2). The training process, including data collection and training, takes only a few minutes, which can be amortized for multiple queries on the same video data. Such training cost amortization is inspired by existing query-time cascade systems [2, 3, 6, 8] that train specialized neural networks for each video.

Labeled data collection for supervised learning. As CoVA aims for large-scale video analytics, manually labeling the video data is infeasible. As such, CoVA needs a method to automatically label the video data. Similar to prior works [2, 3, 8, 29], using pixel domain object detection is a possible option. However, object detection models are not only computationally expensive but also produces labels for non-moving objects, which should not be used to train BlobNet, designed to detect only moving objects. Instead, we exploit the conventional Mixture of Gaussians (MoG) based background subtraction technique since it is lightweight and only looks for the moving objects.

[3]We omit the detailed architecture and refer to the paper [28].

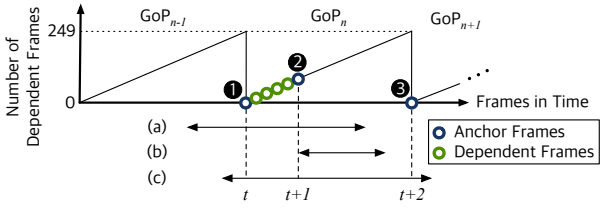

Figure 6: Example scenario of track-aware frame selection.

4.3 Tracking Blobs

Blob detection results. The output of BlobNet is merely a collection of 1's on the resulting bitmap, which lacks the notion of objects. CoVA uses connected-component labeling algorithm to uniquely identify the interesting regions in compressed frames as potential objects, called *blobs*. Once the blob identification process is completed, CoVA obtains the spatial information of blobs on each frame. However, the blobs existing across consecutive frames are not yet temporally associated with each other, which necessitates the next stage of CoVA, blob tracking.

SORT-based blob tracking. The end objective of blob tracking in compressed domain is to minimize the number of frames to be decoded to mitigate the decoding bottleneck. Hence, the tracking algorithm must (1) offer high throughput that significantly outperforms the decoder throughput, (2) while accurately tracking the inter-frame blobs to minimize the accuracy loss at the label propagation stage. We extensively explore existing object tracking techniques in pixel domain [19, 30–36], and choose the SORT object tracking algorithm [19], which satisfies the above two requirements. SORT offers the near-best tracking accuracy among the state-of-the-art tracking techniques and keeps the computation lightweight by exploiting conventional optimization algorithms, Kalman filter and Hungarian assignment.

5 Track-aware Frame Selection

Leveraging the track information, CoVA prudently select a small subset of frames to decode, called *anchor frames*, so as to maximize the decoding throughput. The key idea behind the anchor frame selection algorithm is to pick the ones that require to decode the least number of frames and thus maximize the *effective* decoding throughput.

Dependency between compressed frames. As described in Section 2.3, block-based compression uses a combination of (1) independent frames that are self-contained (i.e., I-frame), and (2) dependent frames (i.e., P/B-frames) that depend on

```
Input  : efs: compressed frames in a GoP_t
         tracks: blob tracks that maintain across GoPs
Output : dfs: compressed frames chosen to be decoded
         afs: anchor frames
1  cur_tracks = tracks that terminate in GoP_t
2                    with no anchor frames assigned
3  dfs = afs = ∅
4  if cur_tracks ≠ ∅ then
5      start_timestamps = sorted(cur_tracks.starts())
6      end_timestamps = sorted(cur_tracks.ends())
7      sidx = eidx = 0
8      for ef in efs do
9          while start_timestamps[sidx] == ef.timestamp do
10             candidate_af = ef
11             sidx = sidx + 1
12         end
13         while end_timestamps[eidx] == ef.timestamp do
14             afs.add(candidate_af)
15             dfs.add_dependants(candidate_af, efs)
16             eidx = eidx + 1
17         end
18     end
19 end
20 dfs.output()
21 afs.output()
```

Algorithm 1: Track-aware frame selection algorithm.

either preceding frames, subsequent frames, or both. Due to the presence of P-frame and B-frame within a GoP, the number of dependent frames that need to be decoded to fully decode a frame follows a saw-tooth structure, as depicted in Figure 6. The number of dependent frames is zero for I-frame at a GoP boundary and grows linearly until it resets to zero at the end of GoP[4].

Selecting anchor frames for decoding. To minimize the decoding load, we leverage two insights: (1) CoVA can find the consecutive frames where an object keeps appearing in the video, and (2) the computations load to decode a frame is proportional to its number of dependent frames. Within each GoP, CoVA identifies a set of anchor frames, which can identify all objects present in the GoP and perform the least computation for decoding, by minimizing the number of dependent frames. The selected anchor frames are the only ones that are passed to the DNN object detector to produce the label information.

Example. Figure 6 presents an example where CoVA identifies three unique objects, (a), (b), and (c), as well as the range of frames where each object stays in the video. In this example, the best choice of anchor frame would be Frame ❷ since (1) all the objects are present in Frame ❷, and (2) Frame ❷ has the least number of dependent frames among frames where all the objects are present.

[4]For brevity, we simplify Figure 6 by only visualizing dependency chains for P-frames since the number of dependent frames for B-frames is similar to that of the nearby P-frames.

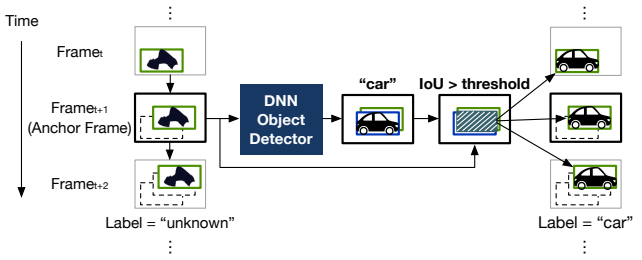

Figure 7: Label propagation.

Algorithm. Algorithm 1 describes the frame selection algorithm in detail. *Line 1*: When a GoP arrives at the frame filtering, to select the anchor frames, CoVA only considers tracks that (1) terminate in that particular GoP and (2) do not have any anchor frames yet (e.g., object (a)/(b) at time t). *Line 9*: Then, as CoVA visits frames in order, it first checks if a track starts appearing in the visiting frame. *Line 10*: If it does, the visiting frame is marked as "candidate" anchor frame (e.g., Frame ❶ at t). Later on, if a new track starts appearing in a successive frame, the frame becomes the new candidate (e.g., Frame ❷ at $t+1$). *Line 14–15*: When a track ends, CoVA adds the current candidate frame into the anchor frame list (e.g., Frame ❷) and inserts all the dependent frames into the dependent frame list (e.g., all frames between Frame ❶ and Frame ❷). The intuition behind this algorithm is that, if a track started but did not terminate, any frame in between can be an anchor frame. However, when a track ends, an anchor frame for the track must be selected, because otherwise, we may not have any anchor frame for the terminating track.

6 Label Propagation

In the last stage, CoVA takes the blob tracks and labels for the anchor frames to temporally propagate the labels across all the frames on the tracks. Figure 7 illustrates the example workflow of label propagation. When the selected anchor frames and their dependent frames are decoded, CoVA takes only anchor frames to perform the DNN object detection and obtain the labels (e.g., "car") as well as their spatial information. To associate the labels with blobs, CoVA first spatially correlates blobs with the detected objects using the intersection over union (IoU) between their bounding boxes (e.g., bounding boxes of blobs and detected objects are denoted using green and blue boxes, respectively). When the IoU is larger than a threshold, CoVA associates the detected objects with blobs and propagates the labels to all frames in the tracks.

Multiple-objects overlapping problem. One challenge with the label propagation mechanism is that when BlobNet fails to separately identify multiple objects clustered together and creates a large single blob, CoVA cannot correctly propagate the multiple labels. To overcome the challenge, we prepend

an additional step to the label propagation. When a multitude of detected objects are spatially overlapped with a single blob, CoVA splits the blob into multiple blobs, proportionally projecting the locations of objects in the anchor frame to the blob. The proportional projection is also applied to other frames in the same track, populating multiple tracks from a single track. This way, CoVA is able to propagate the multiple labels to the separated tracks, instead of giving a single erroneous label to the clustered objects.

Static object handling mechanism. As CoVA relies on the compressed domain analysis to detect blobs, it is impossible to detect static objects from the compression metadata. Therefore, our BlobNet focuses on detecting moving objects, intentionally excluding the static object information from the training data through the use of MoG. However, CoVA still performs full-fledged object detections on anchor frames. Therefore, the static objects can be detected at least on the anchor frames. As the static objects stay still at the same location across subsequent anchor frames, CoVA is able to associate them as the same object and produce the corresponding track.

7 Implementation

System architecture and constituent software modules. We prototype a CoVA system using DeepStream, which is built upon GStreamer, for constructing the skeleton pipeline of video analytics. As described in Section 4, the initial stage of CoVA is the partial decoding, which extracts the metadata. Hardware-accelerated decoder (e.g., NVDEC) does not support partial decoding and only generates the fully decoded frames. Thus, we modify an open-source video codec, libavcodec, such that it only produces the three types of metadata. In addition, CoVA performs two neural network inferences, one for the blob detection and the other for the full DNN inference (YOLOv4). We use on a TensorRT-based DNN inference plugin on DeepStream, nvinfer [37].

Parallelization in CoVA. Our prototype system distributes the computations of pipeline stages over CPU and GPU, while exploiting their parallelism. Initially, CoVA scans the entire video and splits it into chunks at the I-frame boundaries to parallelize the computation on CPU threads. This scanning takes just a few seconds even for hours of video data, which imposes negligible overhead. Such parallelization results in cutting tracks at the chunk boundaries, but its impact on accuracy is negligible since there are only a few dozens of chunks. For a chunk, the first two stages, track detection and frame selection, should be pipelined in the same thread since these algorithms rely on the temporal dependencies of frames. For object detection, anchor frames are independently computed, which can be fully parallelized. Therefore, CoVA maintains only a single thread for object detection and anchor frames from different chunks are batched together for inference.

Table 1: Descriptions of example video analytics queries.

Query	Abbr.	Description	Metric
Binary Predicate	BP	Return frames where querying object appears	Accuracy
Count	CNT	Return the average count of querying object in frames	Absolute Error
Local Binary Predicate	LBP	Return frames where querying object appears in a certain region of frames	Accuracy
Local Count	LCNT	Return the average count of querying object in a certain region of frames	Absolute Error

8 Evaluation

8.1 Methodology

Queries. To demonstrate the effectiveness of CoVA, we evaluate four example queries, two queries widely used in prior work [2, 3, 8], and their spatial variants supported by CoVA. Table 1 reports the list of evaluated queries with their descriptions and accuracy metrics:

(1) Binary Predicate. Binary predicate (BP) query finds frames where queried objects appear. Collecting frames with queried objects is an initial step of advanced analysis, which makes BP an important query for evaluation despite the simplicity. Many retrospective analytics systems evaluate their solutions only using the BP query [2, 3].

(2) Count. The count (CNT) query is introduced by a prior work, BlazeIt [8], which is an aggregate query that counts the number of queried objects appearing in the whole video. As the aggregated count is largely dependent on the length of each dataset, the number is normalized by dividing it by the number of frame counts.

(3) Local Binary Predicate and Local Count. The local binary predicate (LBP) and local count (LCNT) queries are spatial variants of BP and CNT queries, respectively; however, the only difference is that they exclusively look for objects located in a certain region of interest. For instance, users can query northbound traffic in highway monitoring video by annotating the corresponding region of video as "northbound". Serving these queries not only requires the temporal query results, but also needs spatial information to determine the object locations.

Metrics. Table 1 also reports metrics used for each query. We use the same metrics that prior works use to evaluate their

Table 2: Descriptions of video datasets, queried objects, ground truth results, and region of interest used for spatial queries. Note that we use the Yolov4 DNN model applied frame-by-frame to the original video to get ground truth.

Video Name	Num of Frames	Length	Object in Interest	Object Occupancy	Object Count	Local Occupancy	Local Count	Region of Interest
amsterdam [38]	3,580K	33H	Car	70.07%	1.40	29.05%	0.44	Lower Right
archie [8]	3,567K	33H	Bus	10.48%	0.17	6.63%	0.11	Upper Left
jackson [39]	2,921K	27H	Car	31.91%	0.56	18.28%	0.29	Lower Left
shinjuku [40]	1,782K	16H	Car	82.29%	2.19	19.91%	0.38	Lower Left
taipei [41]	3,564K	33H	Car	84.48%	5.03	22.16%	0.64	Lower Right

solutions. For BP and LBP, as in prior works [2, 3], we use *accuracy*, which is a traditional metric for binary classification that evaluates how many observations, both positive and negative, are correctly classified. Similarly, for CNT and LCNT, we use *absolute error* as used in BlazeIt [8].

Datasets. Table 2 reports the video datasets used for the evaluation. Taking a similar approach with prior works [2, 3, 6–9, 42], we collect the video datasets from YouTube live streams [38–41]. They are recorded from statically installed cameras, which is a widely used setup in various applications domains such as traffic monitoring [43–45], security [46, 47], surveillance [48], and healthcare [49]. Video contents involve various kinds of scenarios, which include traffic circle, highway, harbor, city streets, and park. As the datasets have different resolutions ranged from 720p to 2160p, we transcode them to 720p and evaluate the throughput and accuracy for ease of comparison. Note that higher resolutions (e.g., 2160p) create more severe decoding bottleneck, so using them would be favorable to CoVA, producing higher throughput gains and therefore, to be conservative, we choose to use 720p for all video datasets. The rightmost five columns report the ground truth results for the four queries and the region of interest that spatial queries focus on. Getting the ground truth results by manually labeling the hours of video data is infeasible, so we apply a full DNN model (YOLOv4) to the entirety of video in a frame-by-frame manner.

Hardware specifications. Our CoVA prototype is built on a server with two 16-core Intel Xeon Gold 6226R CPU (2.9 GHz), 192 GB of DRAM, and an NVIDIA RTX 3090 GPU (24 GB GDDR6 DRAM). We turn off hyperthreading to avoid interference among threads.

Decoder. For all experiments, we use NVIDIA's hardware accelerated decoder, NVDEC, for both baseline and CoVA systems to make a fair comparison. We choose not to use the CPU decoder, libavcodec, since it shows lower throughput than NVDEC even with 32-core parallelization.

Baseline cascade system. As the baseline, we use existing cascade systems for query time retrospective analytics. As discussed in Section 2, cascade systems such as Tahoma [3]

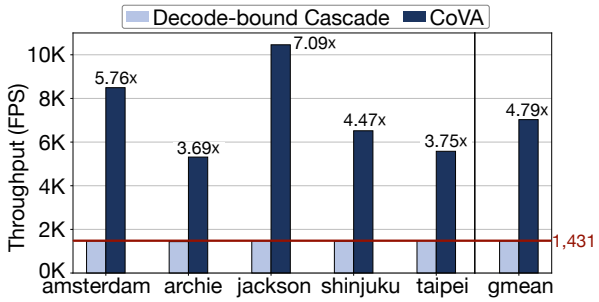

Figure 8: End-to-end system throughput of the baseline decode-bound cascade and CoVA. The throughput of decode-bound cascade is equivalent to the throughput of NVDEC (i.e., 1,431 FPS), which is marked with a red line.

are significantly bottlenecked by video decoding. Therefore, for a conservative comparison with these decode-bound cascade systems, we assume that the cascade systems are only bottlenecked by the decoder, not by any other stages. With this assumption, the throughput of cascade systems is equivalent to the decoder throughput. We refer to this baseline as *decode-bound cascade* in this paper.

8.2 Performance Implication of CoVA

Throughput improvement. Figure 8 compares the end-to-end system throughput of the baseline decode-bound cascade system and CoVA across five video datasets. CoVA achieves on average 4.8× throughput improvement, which ranges from 3.7× for archie to 7.1× for jackson. The significant speedup shows that CoVA effectively pushes a large proportion of analysis to the compressed domain, unclogging the decoding bottleneck that prevents the existing cascades to achieve beyond the constant NVDEC throughput. The results also suggest that depending on the datasets, CoVA sees different speedups. The datasets, jackson and amsterdam, see relatively larger gains, while archie and taipei datasets show lower benefits. These gaps can be attributed to the unique content properties of each evaluated video dataset that deliver varying throughput for the

Table 3: (1) Filtration rate at decoder stage (decode filtration rate) and (2) filtration rate at DNN inference stage (inference filtration rate).

Dataset	Decode Filtration Rate (%)	Inference Filtration Rate (%)
amsterdam	87.16	99.60
archie	72.94	99.15
jackson	94.81	99.79
shinjuku	77.18	99.26
taipei	74.03	99.81

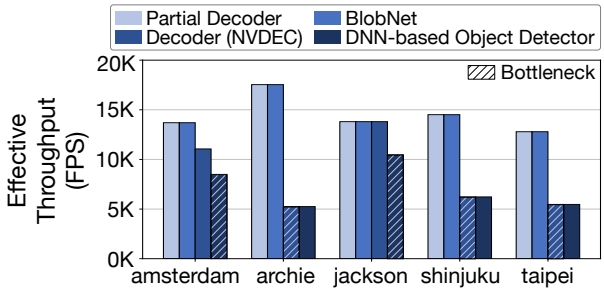

Figure 9: Effective throughput of CoVA stages. The lowest bar represents the bottleneck of CoVA pipeline, which is marked with hatching lines.

CoVA pipeline stages, which eventually engenders a different bottleneck point. To better understand the throughput implication of these stages, we delve into the interplay of algorithms and system in the CoVA pipeline below.

Effectiveness of frame selection. Frame selection is the key to alleviate the decoding bottleneck since it determines the computational load for decoder. Table 3 reports the filtration rates at decoding stage (decode filtration rate) and DNN inference stage (inference filtration rate). The decode filtration rate is calculated based on the number of decoded frames that include both anchor frames and their dependent frames, while the inference filtration rate only considers the anchor frames that are passed to the DNN object detection stage. Intuitively, various semantics of datasets cause different filtration rates. If video contains many objects having lots of motions, blob tracking would produce numerous tracks, which would require many anchor and dependent frames to proceed to the decoder. For crowded video streams such as archie, CoVA sees lower decode filtration rate of 72.94%, while the uncongested ones like jackson capture less activity and provide higher decode filtration rate of 94.81%. Across all datasets, CoVA filters out over 73% to deliver over 3.7× (=100/(100-73)) throughput boost for decoder. At the same time, the inference filtration rate closely reaches 100%, which addresses the DNN bottleneck since the object detector only sees a handful of frames.

Bottleneck analysis. To understand the throughput variation

Table 4: Accuracy results of the four evaluated queries for the video datasets. The acronyms for accuracy metric are specified below.

Dataset	Object	BP (ACC)	CNT (AE)	LBP (ACC)	LCNT (AE)
amsterdam	Car	85.79	0.15	81.61	0.09
archie	Bus	86.96	0.04	90.06	0.01
jackson	Car	86.13	0.10	92.01	0.05
shinjuku	Car	90.15	0.30	91.31	0.05
taipei	Car	87.74	1.10	83.98	0.37
average	-	87.34	N/A	87.69	N/A

* ACC: Accuracy (%), AE: Absolute Error

of CoVA stages across different datasets, we measure the performance of individual stages. Figure 9 reports the effective throughput of each stage by starting from the first partial decoding stage and adding successive stages one by one to the system. The *effective* throughput is defined as the product of the absolute throughput of stage and the accumulated filtration rates. Note that since we measure the throughput from the pipelined system, the effective throughput of a stage cannot be larger than that of the previous stage. The results suggest that different datasets experience bottleneck at different stages. The datasets that attain lower decode filtration rate than the others (i.e., archie, shinjuku, and taipei) are still bottlenecked at the decoder, while the other two datasets are bounded by the DNN object detector. We observe that the inference of BlobNet never becomes a bottleneck and always matches the throughput of the preceding partial decoding stage.

8.3 Accuracy Implication of CoVA

Table 4 reports the accuracy results of evaluated queries. For the BP query, CoVA achieves on average 87.3% accuracy. For the CNT query, CoVA experiences absolute errors from 0.04 (archie) to 1.10 (taipei), respectively. For spatial queries (LBP and LCNT), we do not observe a noticeable difference in accuracy with the temporal queries. The lack of difference is intuitive since CoVA processes the spatial queries by simply restricting the focus of analysis to a certain region of frames. Therefore, the results of spatial variants are merely a subset of temporal query results.

The results show that the approximate nature of compressed domain analysis introduces accuracy loss. However, we argue that such modest level of accuracy degradation (10∼20%) is tolerable to retrospective video analytics, which aims to process large corpus of video data interactively at query time. The video analytics also inherently produce approximate results due to the nature of noisy analog video data and predictive object detection models. Moreover, our accuracy results are conservatively calculated by treating the YOLOv4 detection results as ground truth and marking the CoVA results as error

Table 5: Raw throughput of four different video codecs on the libavcodec and NVDEC decoders.

Codec	Full Decoding (FPS)		Partial Decoding (FPS)
	NVDEC	libavcodec	
VP8	1,590	1,802	32,774
H.264	**1,431**	**1,230**	**16,761**
VP9	3,249	1,179	35,349
H.265	3,888	2,026	25,862

if they do not match. However, we empirically observe that there are many cases where YOLOv4 misses small objects when the objects are faraway from the shooting point, while CoVA can correctly detect them by successfully tracking blobs even for the small objects and propagating the correct labels to the tracks. In this case, the correct results of CoVA would be marked as false positives due to the erroneous ground truth.

Discussion. As discussed above, approximation is fundamentally inevitable for video analytics, because even the best effort results are still imperfect. Thus, our goal in designing CoVA is to achieve *acceptable* approximation accuracy loss for video analytics. According to a study [50], the level of acceptable approximation accuracy loss is higher when the users consider contexts such as application purpose and cost. We believe that CoVA could be a useful tool where analysts can quickly and cost-efficiently extract high-level insights from a large corpus of videos. For instance, consider an application that monitors traffic in a harbor in Amsterdam (see Table 2). For binary predicate query, it suffers from 15% accuracy loss. However, CoVA does not miss the cars completely from the video in most cases since the cars stay in the video for at least several tens of frames (only 2% of cars are eventually missed). Hence, if analysts merely wanted to estimate traffic, CoVA would be able to offer sufficiently precise results. We also believe that if an application requires more accurate results, CoVA could serve as an initial scanning tool that quickly identifies "worth-to-be-further-analyzed" video clips.

8.4 Sensitivity Study

Implication of video codecs. We implement the CoVA system based on H.264, one of the most widely used video codecs. However, to demonstrate applicability of CoVA to other block-based compression standards, we take three alternatives, VP8, VP9, and HEVC (i.e., H.265), and develop metadata extraction in their partial decoding implementations. Table 5 reports throughput results when using the four different codecs with 720p videos and 32 cores. The throughput of NVDEC for the four codecs ranges from 1,431 FPS to 3,888 FPS, which is significantly lower than the effective throughput of existing cascade systems and thus our problem statement regarding decoding bottleneck still holds true. In addition, we observe

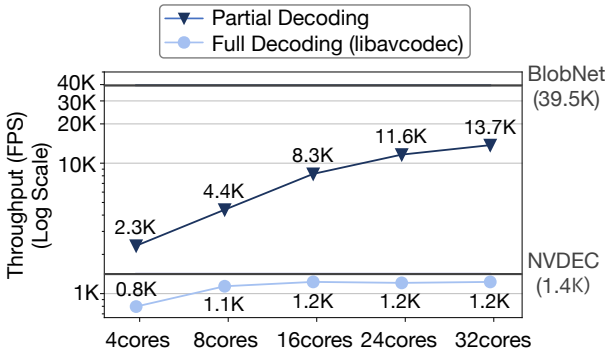

Figure 10: Throughput of partial and full video decoding (libavcodec) on CPU, as the number of cores changes from 4 to 32. For comparison, we also report the throughput of BlobNet and NVDEC, while they have constant throughput since they run on GPU.

that for all codecs, the full decoding throughput in both software and hardware significantly falls short of throughput of the partial decoding. This throughput gap allows CoVA to construct a cascade architecture that enables blob tracking to run at a higher throughput than the vanilla decoder and effectively lowers the full decode workload.

Implication of CPU parallelism. To further analyze the scalability of our parallelization scheme, Figure 10 compares the throughput of partial and full decoding as we parallelize them using the varied number of cores from 4 to 32. We also show the throughput of BlobNet and NVDEC for comparison. Note that these results are averaged across the datasets. The results show that the parallelized partial decoder not only scales significantly better than the full decoder when using the same number of cores (i.e., 1.5× vs. 5.9×), but also largely outperforms the throughput of NVDEC. Currently, we use all the available cores (32) for partial decoding to optimize for throughput. However, one may be able to revise the objective function such that it also takes into account resource utilization and energy efficiency, which we leave as a future work.

9 Related Work

A growing body of literature [2–9, 14, 15, 29, 51–57] aims to address the computational challenges in video analytics. CoVA differentiates itself by addressing video decoding bottleneck, exploiting compressed-domain analysis. Further, CoVA does not require pre-processing, transcoding, or profiling to obtain the benefits.

Cascade architectures for binary predicate queries. NoScope [2] and Lu et al. [5] use a series of approximate pixel-domain filtering stages to build their cascade. Tahoma [3] and Shen et at. [29] use multiple pipelined neural networks to build their cascade architecture. BlazeIt [8] builds on top

of NoScope to support Aggregate and Limit Queries. All five works aim to increase the effective throughput of the system for raw video frames by filtering a majority of the frames using pixel-domain operators. Alternatively, Thia [51] splits up the DNN-inference model using exit points for early termination, similar to the stages of cascade architecture. In contrast, CoVA splits the cascade computation between compressed domain and pixel domain to alleviate the decoding bottleneck.

Spatial queries for video analytics. An emerging class of video analytics systems aim to enable queries based on spatial relationship between labeled objects. Koudas et al. [7] accelerate spatial queries using separate stages for inexpensive DNN-based classification followed by expensive DNN-based object detection. TASM [15] dynamically adapts the layout of tiles, which partition compressed video frames, based on the spatial location of objects to improve performance. Unlike the above works, CoVA uses compressed domain blob tracking to accelerate spatial queries. Unlike TASM, CoVA does not need to update the compression to gain performance benefits.

Storage-accuracy trade-off for decoding bottleneck. VStore [4] uses a search space of fidelity and encoding/decoding knobs (frame sampling rate, resolution, etc) to optimize for query performance and storage cost. SMOL [6] jointly optimizes complexity of the reference DNN for inference and the resolution of data (360p, 720p, etc), for accuracy-performance trade-off. VSS [52] proposes optimizations for video storage to yield higher read rates and compression ratios. CoVA takes an orthogonal approach of performing approximate blob tracking using compression metadata at *query time*. Nevertheless, CoVA is complementary to the above approaches.

Ingest time analysis. Focus [9] generates approximate labels using an inexpensive DNN and Boggart [53] tracks objects at ingest time to generate additional metadata. At query time, both Focus and Boggart use the stored metadata to yield improved performance. Scanner [54] identifies sampling frames offline for pixel domain analysis and skips decoding for all other frames. In contrast, CoVA does not require additional metadata and can operate on standard video compression formats. VideoStorm [55] uses offline profiling data for dynamic load balancing and Chameleon [14] uses inexpensive online profiling to improve accuracy-resource tradeoff at query time. These two profiling approaches are orthogonal and complementary to compressed-domain query processing in CoVA.

Compressed domain object detection. Many prior works [23, 25, 58, 59] have proposed object detection from compression metadata using classical approaches such as predefined kernels [24] and statistical models [20,60,61]. Further, the prior works impose restrictions on the compression-time parameters (e.g., 4 frames per GoP), which limit their applicability [20, 23–25]. Liu et al. [62], Wang et al. [27], and Wu et al. [26] employ DNNs to detect moving objects using *both* pixel and compressed domain data, training a single model for all datasets. BlobNet differs from prior works in the following aspects: (1) BlobNet does not require any pixel data; (2) BlobNet does not impose restrictions on the compression parameters; and (3) BlobNet is trained for given video to compensate the accuracy.

10 Conclusion

Existing cascade systems for video analytics assume to pay significant compute and storage cost for addressing the decoding bottleneck. Further, the systems are specialized for temporal query to achieve otherwise-unachievable throughput. To tackle the two limitations, this paper proposes CoVA, which splits cascade computation between compressed and uncompressed pixel domain. Leveraging the unique characteristics of video analytics and video compression algorithm, CoVA effectively unclogs the decoding bottleneck while additionally supporting spatial queries. Our experiments demonstrate that CoVA reduces the decoding workload by 83.6% and offers 4.8× system speedup compared to state-of-the-art query-time retrospective video analytics systems, while compromising modest accuracy.

11 Acknowledgements

We thank the anonymous reviewers and our shepherd for their comments and feedback. This work was supported by National Research Foundation of Korea (NRF-2020R1A2C1103088) and Information Technology Research Center (ITRC) support program (IITP-2022-2020-0-01795), both of which are funded by the Ministry of Science and ICT, Korea. This work was also partly supported by Samsung Electronics Co., Ltd.

References

[1] Mark Nowell. Cisco VNI Forecast update. https://www.ieee802.org/3/ad_hoc/bwa2/public/calls/19_0624/nowell_bwa_01_190624.pdf, 2021.

[2] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. NoScope: Optimizing Neural Network Queries over Video at Scale. In *PVLDB*, 2017.

[3] Michael R Anderson, Michael Cafarella, German Ros, and Thomas F Wenisch. Physical Representation-Based Predicate Optimization for a Visual Analytics Database. In *ICDE*, 2019.

[4] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. VStore: A Data Store for Analytics on Large Videos. In *EuroSys*, 2019.

[5] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, , and Surajit Chaudhuri. Accelerating Machine Learning Inference with Probabilistic Predicates. In *SIGMOD*, 2018.

[6] Daniel Kang, Ankit Mathur, Teja Veeramacheneni, Peter Bailis, and Matei Zaharia. Jointly Optimizing Preprocessing and Inference for DNN-Based Visual Analytics. In *PVLDB*, 2020.

[7] Nick Koudas, Raymond Li, and Ioannis Xarchakos. Video Monitoring Queries. In *ICDE*, 2020.

[8] Daniel Kang, Peter Bailis, and Matei Zaharia. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics. In *PVLDB*, 2019.

[9] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In *OSDI*, 2018.

[10] Ioannis Xarchakos and Nick Koudas. SVQ: Streaming Video Queries. In *SIGMOD*, 2019.

[11] Jingjing Wang and Magdalena Balazinska. Deluceva: Delta-Based Neural Network Inference for Fast Video Analytics. In *SSDBM*, 2020.

[12] Yuhao Zhang and Arun Kumar. Panorama: A Data System for Unbounded Vocabulary Querying over Video. In *PVLDB*, 2020.

[13] Favyen Bastan, Oscar Moll, and Sam Madden. Vaas: Video Analytics At Scale. In *PVLDB*, 2020.

[14] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodík, Siddhartha Sen, and Ion Stoica. Chameleon: Scalable Adaptation of Video Analytics. In *SIGCOMM*, 2018.

[15] Maureen Daum, Brandon Haynes, Dong He, Amrita Mazumdar, and Magdalena Balazinska. TASM: A Tile-Based Storage Manager for Video Analytics. In *ICDE*, 2021.

[16] NVIDIA. DeepStream SDK. https://developer.nvidia.com/deepstream-sdk, 2021.

[17] NVIDIA. Video Codec SDK. https://developer.nvidia.com/nvidia-video-codec-sdk, 2021.

[18] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H.264/AVC Video Coding Standard. *TCSVT*, 2003.

[19] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, 2016.

[20] Mohammadsadegh Alizadeh and Mohammad Sharifkhani. Compressed Domain Moving Object Detection Based on CRF. *TCSVT*, 2020.

[21] Wei Zeng, Jun Du, Wen Gao, and Qingming Huang. Robust Moving Object Segmentation on H.264/AVC Compressed Video Using the Block-Based MRF Model. *Real-Time Imaging*, 2005.

[22] R. Babu, Kalpathi Ramakrishnan, and S.H. Srinivasan. Video Object Segmentation: A Compressed Domain Approach. *TCSVT*, 2004.

[23] Marcus Laumer, Peter Amon, Andreas Hutter, and André Kaup. Moving Object Detection in the H.264/AVC Compressed Domain. *APSIPA*, 2016.

[24] Chris Poppe, Sarah De Bruyne, Tom Paridaens, Peter Lambert, and Rik Van de Walle. Moving Object Detection in the H.264/AVC Compressed Domain for Video Surveillance Applications. *Journal of Visual Communication and Image Representation*, 2009.

[25] Dien Van Nguyen and Jaehyuk Choi. Toward Scalable Video Analytics Using Compressed-Domain Features at the Edge. *Applied Sciences*, 2020.

[26] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Compressed Video Action Recognition. In *CVPR*, 2018.

[27] Shiyao Wang, Hongchao Lu, Pavel Dmitriev, and Zhidong Deng. Fast Object Detection in Compressed Video. In *ICCV*, 2019.

[28] Radu Sibechi, Olaf Booij, Nora Baka, and Peter Bloem. Exploiting Temporality for Semi-Supervised Video Segmentation. In *ICCV*, 2019.

[29] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy. Fast Video Classification via Adaptive Cascading of Deep Models. In *CVPR*, 2017.

[30] Seung-Hwan Bae and Kuk-Jin Yoon. Robust Online Multi-Object Tracking Based on Tracklet Confidence and Online Discriminative Appearance Learning. In *CVPR*, 2014.

[31] Min Yang and Yunde Jia. Temporal Dynamic Appearance Modeling for Online Multi-Person Tracking. *CVIU*, 2016.

[32] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to Track: Online Multi-Object Tracking by Decision Making. In *ICCV*, 2015.

[33] Alex Bewley, Vitor Guizilini, Fabio Ramos, and Ben Upcroft. Online Self-Supervised Multi-Instance Segmentation of Dynamic Objects. In *ICRA*, 2014.

[34] Wongun Choi. Near-Online Multi-Target Tracking with Aggregated Local Flow Descriptor. In *ICCV*, 2015.

[35] Ju Hong Yoon, Ming-Hsuan Yang, Jongwoo Lim, and Kuk-Jin Yoon. Bayesian Multi-Object Tracking Using Motion Context from Multiple Objects. In *WACV*, 2015.

[36] Alex Bewley, Lionel Ott, Fabio Ramos, and Ben Upcroft. Alextrac: Affinity Learning by Exploring Temporal Reinforcement within Association Chains. In *ICRA*, 2016.

[37] NVIDIA. Gst-nvinfer. https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_plugin_gst-nvinfer.html, 2021.

[38] Webcam Lemmer. Binnenhaven lemmer, youtube. https://www.youtube.com/watch?v=NyzxJMWxDeo, 2019.

[39] See Jackson Hole. Jackson hole wyoming usa town square live cam, youtube. https://www.youtube.com/watch?v=1EiC9bvVGnk, 2018.

[40] KABUKICHO. Shinjuku kabukicho, youtube. https://www.youtube.com/watch?v=EHkMjfMw7oU, 2020.

[41] StarDot Technologies. Taiwan new taipei city, youtube. https://www.youtube.com/watch?v=INR-B7FwhS8, 2020.

[42] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *SIGCOMM*, 2020.

[43] M. Kilger. A Shadow Handler in a Video-Based Real-Time Traffic Monitoring System. In *WACV*, 1992.

[44] Kostia Robert. Video-Based Traffic Monitoring at Day and Night Vehicle Features Detection Tracking. In *ITSC*, 2009.

[45] Tariq Abdullah, Ashiq Anjum, M. Fahim Tariq, Yusuf Baltaci, and Nikos Antonopoulos. Traffic Monitoring Using Video Analytics in Clouds. In *UCC*, 2014.

[46] L. Snidaro, C. Micheloni, and C. Chiavedale. Video Security for Ambient Intelligence. *SMC*, 2005.

[47] Minghu Wu, Xiang Li, Cong Liu, Min Liu, Nan Zhao, Juan Wang, Xiangkui Wan, Zheheng Rao, and Li Zhu. Robust Global Motion Estimation for Video Security Based on Improved K-Means Clustering. *JAIHC*, 2019.

[48] Niels Haering, Péter L. Venetianer, and Alan Lipton. The Evolution of Video Surveillance: An Overview. *MVA*, 2008.

[49] P. Chung, Yung-Ming Kuo, Chin-De Liu, and Chun-Rong Huang. Video Analysis Boosts Healthcare Efficiency and Safety. *Spie Newsroom*, 2011.

[50] Jongse Park, Emmanuel Amaro, Divya Mahajan, Bradley Thwaites, and Hadi Esmaeilzadeh. AxGames: Towards Crowdsourcing Quality Target Determination in Approximate Computing. In *ASPLOS*, 2016.

[51] Jiashen Cao, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. THIA: Accelerating Video Analytics using Early Inference and Fine-Grained Query Planning. *arXiv*, 2021.

[52] Brandon Haynes, Maureen Daum, Dong He, Amrita Mazumdar, Magdalena Balazinska, Alvin Cheung, and Luis Ceze. VSS: A Storage System for Video Analytics. In *SIGMOD*, 2021.

[53] Neil Agarwal and Ravi Netravali. Boggart: Accelerating Retrospective Video Analytics via Model-Agnostic Ingest Processing. In *arXiv*, 2021.

[54] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. Scanner: Efficient Video Analysis at Scale. *TOG*, 2018.

[55] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *NSDI*, 2017.

[56] Ran Xu, Jinkyu Koo, Rakesh Kumar, Peter Bai, Subrata Mitra, Sasa Misailovic, and Saurabh Bagchi. VideoChef: Efficient Approximation for Streaming Video Processing Pipelines. In *ATC*, 2018.

[57] Mengwei Xu, Tiantu Xu, Yunxin Liu, and Felix Xiaozhu Lin. Video Analytics with Zero-streaming Cameras. In *ATC*, 2021.

[58] Orachat Sukmarg and Kamisetty R Rao. Fast Object Detection and Segmentation in MPEG Compressed Domain. In *TENCON*, 2000.

[59] Fatih Porikli, Faisal Bashir, and Huifang Sun. Compressed Domain Video Object Segmentation. *TCSVT*, 2009.

[60] Fernando Bombardelli, Serhan Gül, Daniel Becker, Matthias Schmidt, and Cornelius Hellge. Efficient Object Tracking in Compressed Video Streams with Graph Cuts. In *MMSP*, 2018.

[61] Sayed Hossein Khatoonabadi and Ivan V. Bajic. Video Object Tracking in the Compressed Domain Using Spatio-Temporal Markov Random Fields. *TIP*, 2013.

[62] Qiankun Liu, Bin Liu, Yue Wu, Weihai Li, and Nenghai Yu. Real-Time Online Multi-Object Tracking in Compressed Domain. *IEEE Access*, 2019.

[63] Mark Zhao, Niket Agarwal, Aarti Basant, Buğra Gedik, Satadru Pan, Mustafa Ozdal, Rakesh Komuravelli, Jerry Pan, Tianshu Bao, Haowei Lu, Sundaram Narayanan, Jack Langman, Kevin Wilfong, Harsha Rastogi, Carole-Jean Wu, Christos Kozyrakis, and Parik Pol. Understanding Data Storage and Ingestion for Large-Scale Deep Recommendation Model Training: Industrial Product. In *ISCA*, 2022.