CoVA: Exploiting Compressed-Domain Analysis to Accelerate Video Analytics

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

KAIST, *Google
Growing Video Data

Video data makes up **82% of global IP traffic*** as of 2022, and is **growing**

* CISCO Annual Internet Report
Video Analytics

Video Analytic System analyzes video to extract high-level information and answers user queries.

Example

User Query: How many cars per hour?

Video Analytics System: 1,200 cars
Using Object Detector for Video Analytics

Frame 1

Frame 2

Frame 3

When does a car appear?

Answer: Frame 3!

<table>
<thead>
<tr>
<th>Frame #</th>
<th>Class</th>
<th>X, Y, W, H</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Person</td>
<td>(140, 550, 130, 100)</td>
</tr>
<tr>
<td>3</td>
<td>Person</td>
<td>(870, 570, 140, 100)</td>
</tr>
<tr>
<td>3</td>
<td>Car</td>
<td>(150, 410, 24, 64)</td>
</tr>
</tbody>
</table>
Challenge and Prior Approaches

Challenge

DNN-based object detector requires heavy computation
e.g., YOLO take 11 hours to process two weeks long video

Prior Approaches [VLDB’18, ICDE’20, VLDB’20]

Simple neural networks specialized for the user query

Cascade architecture constitute a pipeline of classifiers that trades accuracy and performance
Prior Approach: Specialized Neural Network

Complex Task
Determine location and class of every object

Simple Task
Determine if there is any car or not

Object Detector

When does car appear?

Specialized Neural Network

1.0 car

0.0 no car

confident

not confident

confident
Prior Approach: Cascade Architecture

When does *car* appear?
Prior Approach: Cascade Architecture

When does *car* appear?
Prior Approach: Cascade Architecture

When does car appear?

Answer: Frame 2
Prior Approach: Cascade Architecture

Specialized NN

Frames flowing...

Applied object detection

0.97

Frame 3

Frame 2

Frame 1

Skipped object detection

When does *car* appear?

Answer: Frame 2 and 3!
Two Limitations of Prior Approaches

1. Bottleneck from Decoding

- Prior works ignore a compute-heavy preprocessing stage, video decoding!

* 720p video with HW acceleration, NVDEC

Prior approach speedup

30 FPS

Decoding bottleneck

80,000 FPS

1,400 FPS*

* 720p video with HW acceleration, NVDEC
Two Limitations of Prior Approaches

2. Lack of Support for Spatial Query

Skipped object detection

Applied object detection

When does car appear?
Frame 2 and 3

Where does car appear?
Frame 2 at (150, 410)
Frame 3 at ?
**Contribution 1**: 4.8× end-to-end speedup by addressing decoding bottleneck

**Contribution 2**: Spatial query support
CoVA Overview

Track Detection
- BlobNet
- Blob Tracking

Frame Selection → Decoder → Object Detector → Label Propagation
Track Detection
Goal of Track Detection

**Goal:** without decoding, find track of moving objects

How can we find moving objects from compressed video?
How modern video codecs work

Algorithmic commonality: \textit{Block-based compression}
Block-based Compression: Macroblock

Frames are first divided into a grid of *macroblocks*
Block-based Compression: Motion Vector

Macroblock is compressed by saving relative position to similar block
Challenge in Using Compression Metadata

**Challenge:** Find moving object from noisy compression metadata

**Solution:** Neural network based algorithm
BlobNet

Input
- Compression Metadata
  - Motion vector
  - MB type*
  - MB partition*

Embedding Layer
- Additional layer for neural network to embed compression metadata

Temporal U-Net
- Encoder-decoder architecture for denoising
- Video instance segmentation model architecture running in pixel domain

Output
- Training label generated using background subtraction in pixel domain

* Details omitted in the talk
BlobNet Result

- *blob*: region where moving objects appear
Detecting Tracks from Blobs

Blobs detected by BlobNet are not tracked yet

Tracking with Simple Online and Realtime Tracking (SORT)
Frame Selection
Goal of Frame Selection

**Goal**: select minimal frames to decode

Decoding is required to see what *kinds* of objects they are
Can we just pick any of the frames to decode?
Not every frame has the same decoding cost
Dependency-Aware Frame Selection

Frame 40: Object A enters
Frame 120: Object B enters
Frame 140: Object C enters
Frame 170: Object A leaves
Frame 230: Object B leaves

Number of frames to decode vs. Frame number

Optimal frame to decode

Object A enters
Object B enters
Object C enters
Object A leaves
Object B leaves

Object A
Object B
Object C

Track Detection  Frame Selection  Decoding  Object Detection  Label Propagation
Decoding and Object Detection on Selected Frame
Label Propagation
Goal of Label Propagation

**Goal:** combine results from previous stages to label tracks
Overlap based label propagation

*blob* at the same timestamp

Retrieve *blob* location at the timestamp of object detected frame
Overlap based label propagation

Assigned labels are *propagated* throughout the track, including not decoded frames
CoVA Summary

- **Track Detection**
  - BlobNet
  - Blob Tracking

- **Frame Selection**

- **Decoder**

- **Object Detector**

- **Label Propagation**

---

**Compressed Domain**
- Perform Filtering
- Extract Tracking Information

**Pixel Domain**
- Extract Label Information
# Evaluation Setup

Datasets: five live stream videos / Average 28 hours long

<table>
<thead>
<tr>
<th>Query specification</th>
<th>System specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Predicate (BP)</td>
<td>Software</td>
</tr>
<tr>
<td>Frames where querying object appears</td>
<td>C++ &amp; Rust / CUDA 11.5</td>
</tr>
<tr>
<td>Global Count (CNT)</td>
<td>Decoder</td>
</tr>
<tr>
<td>Average count of querying object</td>
<td>FFmpeg v4.41 / NVDEC v5</td>
</tr>
<tr>
<td>Local Binary Predicate (LBP)</td>
<td>CPU</td>
</tr>
<tr>
<td>BP with spatial constraint</td>
<td>Two Intel Xeon CPU Gold 6226R</td>
</tr>
<tr>
<td>Local Count (LCNT)</td>
<td>GPU</td>
</tr>
<tr>
<td>GC with spatial constraint</td>
<td>NVIDIA RTX 3090</td>
</tr>
</tbody>
</table>
End to End System Throughput Improvement

Achieves 4.8\times higher throughput in average compared to prior work
### Filtration rate

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Decode Filtration Rate (%)</th>
<th>Inference Filtration Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>amsterdam</td>
<td>87.16</td>
<td>99.60</td>
</tr>
<tr>
<td>archie</td>
<td>72.94</td>
<td>99.15</td>
</tr>
<tr>
<td>jackson</td>
<td>94.81</td>
<td>99.79</td>
</tr>
<tr>
<td>shinjuku</td>
<td>77.18</td>
<td>99.26</td>
</tr>
<tr>
<td>taipei</td>
<td>74.03</td>
<td>99.81</td>
</tr>
<tr>
<td>geomean</td>
<td>80.80</td>
<td>99.39</td>
</tr>
</tbody>
</table>

Reduces decoding workload by 80.8%, and inference by 99.4% on average.
Bottleneck Analysis of CoVA

Bottleneck of varies across dataset

Compressed domain filtering never becomes the bottleneck
Implication on accuracy

Degrades accuracy in modest level comparable to prior works

E.g., Degradation in binary predicate query is in the range of 10-15%

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BP (%)</th>
<th>CNT (Err)</th>
<th>Ground Truth*</th>
</tr>
</thead>
<tbody>
<tr>
<td>amsterdam</td>
<td>85.79</td>
<td>0.15</td>
<td>1.40</td>
</tr>
<tr>
<td>archie</td>
<td>86.96</td>
<td>0.04</td>
<td>0.16</td>
</tr>
<tr>
<td>jackson</td>
<td>86.13</td>
<td>0.10</td>
<td>0.56</td>
</tr>
<tr>
<td>shinjuku</td>
<td>90.15</td>
<td>0.30</td>
<td>2.18</td>
</tr>
<tr>
<td>taipei</td>
<td>87.74</td>
<td>1.10</td>
<td>5.03</td>
</tr>
<tr>
<td>geomean</td>
<td>87.34</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Comparison made against YOLOv4 as ground truth
Conclusion

- Novel video analytics pipeline that introduces compressed domain analysis
- 4.8× on average speedup by addressing decoding bottleneck
- Support for spatial query

Opensourced Artifact evaluated