Efficient Calculation of Frame Level Complex Predicates in Video Analytics

Aubhro Sengupta

December 2022

1 Project Information

My research project is in the Georgia Tech Databases Group under Professor Joy Arulraj. I am mentored by Dr. Arulraj and Gaurav Tarlok Kakkar, a graduate student. I plan on taking the peliminary results of this work and writing a short paper for a systems journal. I hope to build on the results of this work and submit a conference paper to VLDB or SIGMOD.

Figure 1: Example traffic image from dataset.

2 Introduction

Lets consider a scenario in which we have a large amount of video from a traffic camera at a certain busy intersection and we are looking for a black sedan. The number and black sedans and the location of each sedan within the frame would be an example of a frame-level predicate. State of the art object detectors such as FastRCNN typically utilize convolutional neural networks that act on a frame and return the locations of every instance of the type of object (in this case a black sedan) in the frame. These functions that calculate frame-level predicates are often referred to as UDFs, or user defined functions.

UDFs are often computationally expensive to calculate and thus it is not practical to run on every single frame in a video per query. Therefore it becomes paramount to reduce the number of frames evaluated by the UDF to execute a query faster and with an acceptable level of accuracy. The preexisting work in the field focus on using techniques such as sampling, filtering, and clustering to narrow the number of frames that are evaluated by the UDF in order to execute a query more efficiently [1].

3 Background

The field of video analytics focuses on extracting useful information from video. Lets consider a scenario in which we have a large amount of video from a traffic camera at a certain busy intersection and we are looking for a black sedan. State of the art object detectors such as FasterRCNN [3] utilize computationally expensive methods like convolutional neural networks that analyze a frame of video and estimate the number of the object of interest and the locations of every instance of that object in the frame. The most basic approach to solving this problem would simply be to execute the object detector on all frames of the video and collect the frames which contain at least one black sedan to return to the user. However, this approach is impractical on longer videos as CNNs are computationally expensive and thus too slow. Instead the number of frames evaluated by the object detector must be limited. This field focuses on developing strategies for doing so, such as sampling, filtering, proxy models, and clustering.

4 Problem Statement

Most of the current approaches focus on the execution of simple predicates, which involve a single UDF. Complex predicates are conjunctions of multiple simple predicate. For example, if we wanted to find all the frames with a black sedan in which the traffic light was green, that would be a complex predicate and involve multiple UDFs. Most state of the art approaches are meant for executing a single predicate and can just be utilized multiple times for each predicate to evaluate a complex predicate. The problem with this approach is as you add

more predicates, the accuracy will drop considerably and the computation time will explode.

5 Solution Design

My proposed solution involves a combination of adaptive sampling with a proxy modelling and ordering the search to look at sections of the video most likely to contain the complex predicate. The first requirement is a proxy model that provides a good estimate for the number of objects in a frame. This proxy is not predicate specific and will not rely on any UDFs to calculate. A list of UDFs for each possible predicate must also be provided.

First comes the pre-processing step. The video will be split into temporal chunks. The proxy model will be run on all frames and the standard deviation will be calculated for each of the chunks. The standard deviation will determine the sampling rate, and the sampling budget will be split among the chunks such that the chunks with the higher standard deviations will receive more of the sampling budget [2]. Every frame sampled will be evaluated by all UDFs.

A complex predicate query consists a conjunction of multiple simple predicates and as such require some subset of the list of UDFs provided to calculate. The sampled data from the preprocessing step will be used to estimate mean and standard deviations of each predicate and use it to rank the predicates by relative rarity.

The regions of the video with the closest pythagorean distance to the complex predicates in which each simple predicate is a dimension are searched first. Those regions are evaluated by the UDFs in the order of least likely to most likely.

Overall, I will implement this approach to evaluate queries containing complex predicates quicker and more accurately than combining state of the art approaches for simple predicate query optimization.

6 Priorities

First I will collect video data used by others in this field and write a naive solution. This will help me develop a baseline for my own solution. Then I will implement my solution and run it to collect data on the improvements made with my solution.

~~I will take the data collected and describe the methods used closely in order to write my thesis.~~

7 Experiment Setup

7.1 Hardware

In order to run optimization experiments with complex predicates, we must first construct an experimental platform. All experiments will be run on ada-01, a server owned an operated by the Georgia Tech Databases group. This server contains two Nvidia Quadro GPUs, which will enable me to run experiments quickly. The server is shared, and the other workloads may slow down certain experiments. However, this will not matter as the speed will be measured by the number of frames each model reads.

7.2 Software Tools

All code will be written using Python and commonly-used libraries such as Numpy, Pandas, PyTorch, etc. Python and it's library ecosystem is popularly used in the data science and machine learning community. The field of video analytics aims to create Database Management Systems to allow the efficient querying using machine learning models on large volumes of video. Since most of these models are written in Python, this system will also be written in Python. Other tools that will be used are Jupyter notebooks to run experiments, and Petastorm to read and write the data.

8 Methodology

I will train 3 models to detect cars, busses, and the color of the traffic light. I will formulate the queries to detect return the frames with busses when the traffic light is green, cars when the traffic light is red, and cars and busses together. These queries will be executed with all 3 models and the results will be stored.

8.1 Queries

The queries were picked to sample the wide space of possible queries in a VDBMS such as this. It is near-impossible to test all possible queries in a system such as this so a sample will have to do.

8.2 Models

Multiple models will be used in the complex predicates in order to get more generalized data. The models used will be FastRCNN, AlexNet, and VGG. These models are state of the art and widely used.

Figure 2: Experimental Queries

```
--- Q1: Suspicious Vehicle Tracking
SELECT timestamp , bbox , VEHICLE_COLOR (bbox , frame )
FROM VIDEO CROSS APPLY
OBJECT_DETECTOR (frame) ACCURACY 'HIGH '
WHERE timestamp > 6pm AND label = 'car '
AND AREA(bbox) > 0.3 AND
VEHICLE_MODEL (bbox , frame) = 'SUV ';

--- Q2: Suspicious Vehicle Tracking
SELECT timestamp , bbox , LICENSE (bbox , frame )
FROM VIDEO CROSS APPLY
OBJECT_DETECTOR (frame) ACCURACY 'HIGH '
WHERE timestamp > 7pm AND timestamp < 8pm
AND label = 'car ' AND AREA(bbox) > 0.3
AND VEHICLE_COLOR (bbox , frame) = 'red '
AND VEHICLE_MODEL (bbox , frame) = 'SUV ';

-- Q3: Suspicious Vehicle Tracking
SELECT timestamp FROM VIDEO CROSS APPLY
OBJECT_DETECTOR (frame) ACCURACY 'HIGH '
WHERE timestamp > 4pm AND label= 'car ' AND
AREA(bbox) >0.15 AND LICENSE (bbox , frame)= 'XYZ60 ';

--- Q4: Traffic Monitoring
SELECT timestamp , COUNT (*) FROM VIDEO CROSS APPLY
OBJECT_DETECTOR (frame) ACCURACY 'LOW ' WHERE
label = 'car ' AND AREA(bbox) > 0.15
GROUP BY timestamp ;
```

8.3 Dataset

The UA Detrac data-set will be used for all experiments. It is a traffic data-set showing vehicles driving at a busy intersection. Querying traffic footage is a common application in video analytics, and many other papers in this field use this data-set.

9 Methodology

9.1 Architecture

I will write a Python package with an API that allows the user to run a model on a frame given the index. There will also be functions that use this API to

determine what frames to run the model on given the sampling method used. A Jupyter Notebook will be used to run experiments.

9.2 Data Collection

The API will be set up to record the number of frames the model has been run on that this will be the main output of the experiments. The frames returned will also be compared against the source of truth, and this will be used to calculate the accuracy. The results will be stored in a Pandas Dataframe and plotted using Python plotting libraries.

10 Results (new)

Every experiment is run on the UA-Detrac dataset. UA-DETRAC is a challenging real-world multi-object detection and multi-object tracking benchmark. The dataset consists of 10 hours of videos captured with a Cannon EOS 550D camera at 24 different locations at Beijing and Tianjin in China. The videos are recorded at 25 frames per seconds (fps), with resolution of 960×540 pixels. There are more than 140 thousand frames in the UA-DETRAC dataset and 8250 vehicles that are manually annotated, leading to a total of 1.21 million labeled bounding boxes of objects. We also perform benchmark tests of state-of-the-art methods in object detection and multi-object tracking, together with evaluation metrics detailed in this website.

11 Data

When my experiments are completed, I will add the results and a visualization here. I will be showing the execution speed and number of frames checked for all 5 queries defined above on the dataset. I will create a series of bar charts showing that.

12 Discussion

There results show an increase in speed using my method. The issue is if the margin is enough to justify the additional complexity. In many systems applications, simplicity is key and sometimes a method that is theoretically faster may not be worth implementing due to additional constraints such as memory. For example, in operating systems, when running a search in a small block of memory, it is better to use linear search rather than invent a complicated new search algorithm.

The implications of the results of my work are far-reaching. We live in a time of unprecedented surveillance. Cameras are everywhere. The main bottleneck to using the video data in an effective way is the lack of an ability to process the data and gain useful insight. This is what the field of video analytics sets out

to do. By setting up ways to query this video efficiently such as in this project, we are making it easier to spy on people. Although I believe in open scientific progress, it is important to be aware of the ethical consequences of your work.

13 Limitations

The lack of a variety of data is the largest limiting factor. I used UA-Detrac, a standard dataset for benchmarking. However, whether or not the performance benefits generalize to all types of dataset is a separate issue not addressed. These experiments served as a proof of concept rather than a guarantee.

14 Future Work

In the future I will test my methods with other datasets. If similar results are reached then the evidence grows stronger. I will also add more queries to broaden the experiments.

In order to conduct the experiments, I developed a library for simplified video analytics as well as a functional-style API. This library has very few dependencies and makes running these experiments straightforward. I will continue to develop this library as it can possibly be of use to others in the field.

References

[1] Daniel Kang, Peter Bailis, and M. Zaharia. "BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics". In: *Proc. VLDB Endow.* 13 (2019), pp. 533–546.

[2] Oscar Moll et al. "ExSample: Efficient Searches on Video Repositories through Adaptive Sampling". In: *ArXiv* abs/2005.09141 (2020).

[3] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2015), pp. 1137–1149.