Challenges and Opportunities in DNN-Based Video Analytics: A Demonstration of the BlazeIt Video Query Engine

Daniel Kang, Peter Bailis, Matei Zaharia
Stanford DAWN Project, InfoLab

ABSTRACT

As video volumes grow, analysts are increasingly able to query the real world. Since manually watching these growing volumes of video is infeasible, analysts have increasingly turned to deep learning to perform automatic analyses. However, these methods are: costly (running up to 10x slower than real time, i.e., 3 fps) and cumbersome to deploy, requiring writing complex, imperative code with many low-level libraries (e.g., OpenCV, MXNet). There is an incredible opportunity to leverage techniques from the data management community to automate and optimize these analytics pipelines. In this paper, we describe our ongoing work in the Stanford DAWN lab on BlazeIt, an analytics engine for scalable and usable video analytics that currently contains an optimizing query engine. We propose a demonstration of BlazeIt's query language, FrameQL, its use cases, and our preliminary work on debugging machine learning, which will show the feasibility of video analytics at scale. We further describe the challenges that arise from large-scale video, progress we have made in automating and optimizing video analytics pipelines, and our plans to extend BlazeIt.

1 Introduction

Video can be used to answer queries about the real world, both over large historical datasets (e.g., how many people were in Times Square in July?) and in real time (e.g., which streets are the busiest now?) in a variety of domains ranging from urban planning, autonomous vehicle planning, and ornithology. For example, an urban planner working on traffic meter setting or city planning [5] may be interested in whether Mondays have notably different traffic volumes than Tuesdays, and counts the number of cars that enter and exit on various highway exits. An analyst at an autonomous car company may notice the car behaves strangely at lane divers with poor markings and searches for events at lane dividers with poor lane markings [20]. An ornithologist may be interested in the feeding patterns of birds and monitors various bird feeders.

It is not cost effective and is too time-consuming to manually watch these growing quantities of video (London alone has over 500k CCTVs [1]) to answer these questions. Automated methods of video analysis are increasingly important in answering such queries.

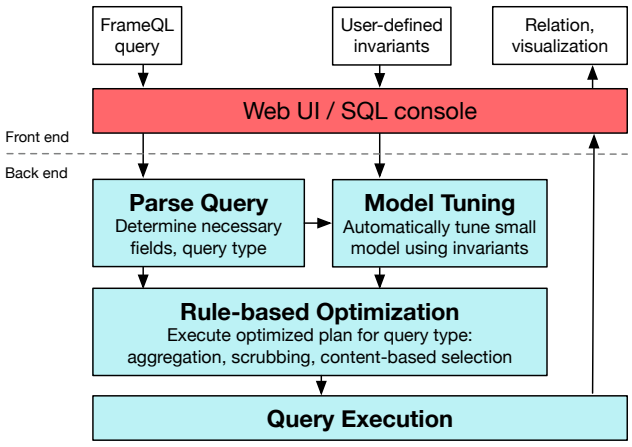

Figure 1: Architecture diagram for BlazeIt.

Fortunately, modern computer vision (CV) techniques have made great strides in automating these tasks, with near human-levels of accuracy for some restricted tasks [9] and rapid progress on others [8] in the form of deep neural networks and models. These trends present an incredible opportunity for visual data management research. We believe that applying data management techniques to video will enable analyses over the real world, as relational DBs have enabled analyses over structured data.

Unfortunately, it is prohibitively expensive to naively deploy these deep models at scale. While these models could be run exhaustively over video to extract relevant information and subsequently be used to answer queries via a traditional query engine, they come at great computational cost. Running a deep model over a single frame of video can take up to hundreds of billions of FLOPs (and run up to 10× slower than real-time), making it infeasible to exhaustively extract information from video. Systems builders have created custom pipelines for specific tasks (e.g., binary detection) [11, 15], but these systems cannot adopt to user's queries. As a result, we believe it to be necessary to rethink analytics for video.

In addition to the computational expense, analysis with deep networks poses several other challenges:

- **Scalability**: we believe the trend of deeper and more computationally expensive models will continue. As networks today are already infeasible at scale, we view scalability as a major obstacle to video analytics.
- **Usability**: using these deep networks requires knowledge

9th Biennial Conference on Innovative Data Systems Research (CIDR '19) January 13-16, 2019 , Asilomar, California, USA.

of CV, deep learning, and programming, and often requires writing complex, ad-hoc code.

- **Integration with the data ecosystem**: many video queries (e.g., searching for actions or events) are not easily expressed in standard SQL.
- **Storage and indexing**: As running deep models over every frame of video is infeasible, we can only index a small fraction of the video. Additionally, existing video storage formats are not well suited for indexing.
- **Real-time analysis and actuation**: as cameras proliferate, so will the demand for real-time analysis and actuation (e.g., decision making). Due to the size of video, streaming the data back to a central server is infeasible.

We have begun to build a system called BlazeIt for usable and scalable video analytics to demonstrate the feasibility of video analytics at scale and address the challenges we have listed. BlazeIt currently contains an optimizing query engine and its system diagram is shown in Figure 1.

In the remainder of this paper, we describe our current progress in BlazeIt, our plans to extend BlazeIt, and a description of our demonstration of BlazeIt.

2 Related Work

BlazeIt builds on a long tradition of data management for multimedia and video, and on recent advances in CV. We outline some relevant parts of the literature below.

Visual data management. Visual data management has aimed to organize and query visual data [6, 24]. These systems were followed by a range of "multimedia" database for storing [4, 19], querying [3, 18], and managing [13, 27] video data. These systems use classic CV techniques such as low-level image features (e.g., colors, textures) and/or rely on textual annotations for semantic queries. However, recent advances in CV allow the *automatic* population of semantic data and we believe it is critical to reinvestigate these systems.

Many query languages for visual data have been developed [12, 22]. However, much of the prior work has assumed the data is provided (e.g., actors in a movie) or that low-level image features are queried. In this work, we describe how these fields can be *automatically populated* and optimized.

Modern video analytics. Systems builders have created specific pipelines for video analytics, but have yet to built a complete query system with a rich query language. For example, [2, 11, 15] optimize binary detection (the presence or absence of a particular object class in video). However, these systems cannot adapt to user's queries. In BlazeIt, we augment these pipelines with a rich query language and novel optimizations which these systems do not support.

Other systems reduce the latency of live queries [28] or increase the throughput of batch analytics [25] when the computation is pre-defined *as a black-box computation graph*. These systems cannot perform certain optimizations such as in BlazeIt, as they do not have access to the computation semantics. BlazeIt could be run on such systems for live queries or scale-out.

A range of contemporary systems aim to optimize complementary functions, including arbitrary UDFs [23], visualization [26], and patch-based visual analytics [17]. To the best of our knowledge, BlazeIt is the first system with a declarative query language and query optimizer for visual analytics that automatically extracts information from pixel data directly, without requiring knowledge of the neural networks used to process the pixel data.

```
SELECT FCOUNT(*)
FROM taipei
WHERE class = 'car'
ERROR WITHIN 0.1
AT CONFIDENCE 95%
```

(a) The FrameQL query for counting the frame-averaged number of cars.

```
SELECT timestamp
FROM taipei
GROUP BY timestamp
HAVING SUM(class='bus')>=1
   AND SUM(class='car')>=5
LIMIT 10 GAP 300
```

(b) The FrameQL query for selecting 10 frames of at least one bus and five cars, with each frame at least 10 seconds apart.

```
SELECT *
FROM taipei
WHERE class = 'bus' AND redness(content) >= 17.5
  AND area(mask) > 100000
GROUP BY trackid
HAVING COUNT(*) > 15
```

(c) The FrameQL query for selecting all the information of red buses at least 100,000 pixels large, in the scene for at least 0.5s (at 30 fps, 0.5s is 15 frames). The last constraint is for noise reduction.

Figure 2: Three FrameQL example queries.

3 BlazeIt Overview

Over the past two years, we have built systems for scalable and usable video analytics in the DAWN lab at Stanford University, beginning with NoScope [15]. NoScope leveraged model specialization, in which a smaller, cheaper model is trained to mimic a larger reference model on a reduced task (e.g., counting cars in a frame vs full object detection). However, our work on the initial NoScope system only optimizes binary detection.

In response, we have initiated work on BlazeIt [14], which significantly expands NoScope by including an optimizing query engine and a query language, FrameQL. BlazeIt's primary goal is to execute FrameQL queries efficiently, as materializing the records is the primary computational bottleneck. Despite our progress, several challenges remain. We describe BlazeIt below and describe our plans to extend BlazeIt in Section 4.

3.1 FrameQL Overview

BlazeIt queries are specified via FrameQL, a SQL-like language. FrameQL allows users to query the objects appearing in a video feed over space and time by content and location at the frame level. We choose a declarative language for two reasons. First, by providing a table-like schema using the standard relational algebra, we enable users with only familiarity with SQL to query videos, whereas extracting the data necessary for these queries manually would require expertise in deep learning, CV, and programming. Second, the separation of the specification and implementation enables new forms of optimizations.

In FrameQL's data model, each video (stored and compressed in formats such as H.264) is represented as a relation, so they can be composed with relational operators. FrameQL supports selection, projection, and aggregation. FrameQL's data schema contains fields relating to the time, location, and class of objects, scene and global identifiers, the box contents, and the features from the object detection method. BlazeIt can populate these fields automatically.

We show three examples in Figure 2, with full details in [14].

3.2 BlazeIt's Architecture

To materialize FrameQL records, BlazeIt uses object detection, which, given a frame of video, returns the set of bounding boxes and class information of the objects in the video. These

object detectors come in the form of deep networks.

In traditional query processing engines, the records are typically cheap to process (e.g., 100s of cycles to apply a predicate), but materializing the records for a frame of video, i.e., performing object detection, is extremely expensive (e.g., billions of FLOPs for Mask R-CNN [8]). This computational cost also makes the solution of materializing all the records infeasible.

When the user issues an FrameQL query, BlazeIt's query engine optimizes and executes the query. BlazeIt's primary challenge is executing the query *efficiently*: naive methods, such as performing object detection on every frame or using NoScope [15] as a filter, are often prohibitively slow. To optimize and execute the query, BlazeIt inspects the query contents to see if optimizations can be applied. For example, BlazeIt cannot optimize `SELECT *`, but it can optimize computing the average number of cars per frame by sampling and applying model specialization [15].

Because object detection is the major computational bottleneck, BlazeIt's optimizer primarily attempts to reduce the number of object detection calls while achieving the target accuracy. BlazeIt leverages three novel optimizations and existing techniques from NoScope to reduce the computational cost of object detection. One key primitive we use in BlazeIt is called *model specialization*, which we developed in NoScope [15].

BlazeIt currently targets aggregation (e.g., Figure 2a), scrubbing (e.g., Figure 2b), and content-based selection (e.g., Figure 2c):

- **Aggregation**: to optimize aggregation, BlazeIt uses specialized networks to directly compute the answer or to reduce the variance in sampling.
- **Scrubbing**: to optimize scrubbing, BlazeIt trains specialized networks to search for frames matching the predicate and adapts importance sampling for fast search.
- **Content-based selection**: to optimize content-based selection, BlazeIt infers, from the query, a set of filters that discards irrelevant frames.

These optimizations can give up to three orders of magnitude speedup. [14] provides full details.

4 Ongoing Work

While the core ideas in FrameQL and in BlazeIt's current optimizations offer significant speedups in video analytics, many challenges remain. We describe several of these challenges, as well as how we are incorporating solutions into BlazeIt.

Scalability. While model specialization currently can speed up restricted tasks (e.g., counting, classification), some queries require the full functionality of object detection, the overwhelming bottleneck for many queries. We are exploring ways of automatically specializing object detection methods.

Off-the-shelf object detectors are trained on generic datasets, but are often deployed in specific scenarios, e.g., on a street corner in New York or inside a store. While powerful object detection methods (e.g., Mask R-CNN [8]) may be able to be deployed in all such scenarios, weaker models (e.g., SSD [21] with a MobileNet [10] backbone) may not be as accurate. However, similar to how specialized models forego the full generality for inference speed, we plan to explore fast methods of specializing these weaker object detection models in restricted scenarios (e.g., a model inside a store may not need to detect cars).

Storage and indexing. While storage of video has been extensively studied for efficient compression, there are new opportunities for storage and indexing for neural network-based analytics engines. For efficient storage, videos are compressed using spatial information. For example, many storage formats (including H.264) largely consist of two types of frames: I-frames, which capture full frames of video, and P-frames, which capture deltas between frames. We believe learning format-specific neural networks for compressed data is promising and can avoid decoding P-frames in many situations, similar to [7].

Additionally, we have found that specialized neural networks can operate on significantly reduced representations of the video frames (e.g., scaled-down frames or the results of specific convolution layers). We plan to explore methods of indexing via specialized networks for exploratory queries, while maintaining high accuracy for downstream queries.

Real-time analytics and actuation. While batch analytics can be useful for understanding aggregate behavior, many scenarios require real-time analytics and actuation, e.g., real-time traffic meter setting.

Deep learning raises a new set of concerns for real-time analytics. Many robots (e.g., autonomous vehicles, drones) and sensors (e.g., street cameras) will contain accelerators that run deep networks for a restricted set of tasks and possibly reduced accuracy. For example, an autonomous vehicle may have an object detection network, but no model for collecting the demographic information of the pedestrians it sees. Additionally, it will become increasingly costly to transfer data to a central server for storage and processing.

We believe model specialization for real-time analytics on edge devices and bandwidth reduction are promising. For example, edge devices have a number of competing constraints (e.g., power, bandwidth, accuracy) and we believe extending inference-aware model search to account for multiple constraints is promising.

Usability via debugging. Despite advances in deep learning and specialization, deep and specialized models are not always accurate, often in predictable ways. The accuracy of these methods is a major obstacle to accurate analysis.

We have begun work on a model debugging system that will help identify and fix model issues via *model assertions* [16]. Model assertions are boolean statements over the output of models and are similar to program assertions. For example, an analyst may code an assertion that there should not be 3 cars within another car bounding box (a common error for lower-quality object detection methods). These assertions can be used to fix behavior at run-time, or collect data to improve models.

5 Demonstration

To highlight the ease of use of FrameQL for video analytics, we will present a demonstration of our UI and accompanying query engine. Our demonstration will contain two parts: a textual interface for users to issue FrameQL queries and a visualization tool to display the frames and metadata of the returned results.

The textual interface will accept FrameQL queries such as in Figure 2. For queries that return summary statistics (i.e., aggregation), the demo will display the output as is. For queries that return frame numbers and box information, we will provide a visualization tool that can show frames or a sequence of frames with the boxes (and metadata) drawn on the frames. An example of the interface is shown in Figure 3 and an example of a visualized frame is shown in Figure 4b.

We will demonstrate two use cases of BlazeIt in the urban planning scenario: 1) exploratory queries and 2) model debugging via user-defined invariants.

Exploratory query use case. We will first demonstrate the

Figure 3: An example query with example answer and visualized frame.

(a) An example when the assertion triggers.

(b) Simply collecting instances of objects is not sufficient.

Figure 4: An example of an error an object detection method can make on `night-street`.

ease of use of FrameQL for an urban planning use case. The analyst has access to a set of cameras at street corners.

First, we demonstrate how the urban planner can count the number of cars in the video, which can be accomplished with the query in Figure 2a. The query specifies an error bound via `ERROR` for fast execution. This count information can be used to set traffic metering or for understanding traffic patterns.

Second, we demonstrate how the urban planner can look for rare events by querying for events of at least five cars and at least one bus, which can be accomplished with the query in Figure 2b. The urban planner may be interested in such events to understand how public transit interacts with congestion.

Finally, we demonstrate how to perform content-based selection by searching for red buses, which can be accomplished by the query shown in Figure 2c. The urban planner may be interested in searching for such events to understand how tourism affects traffic and looks for red buses as a proxy for tour buses. This query shows how to exhaustively select frames with red buses. Here, `redness` and `area` return the measure of redness of an image and the area of a box respectively.

Model debugging use case. We will additionally demonstrate ongoing work [16] to extend BlazeIt with debugging capabilities via user-defined invariants.

An analyst may view a video of a street corner and notices a common error is multiple overlapping boxes of a car when there should only be one box. Collecting instances of this error is prohibitively expensive with random sampling or by collecting instances of objects, as shown Figure 4b.

The analyst writes an assertion that several boxes of cars should not overlap. By running this assertion, the analyst is able to collect training instances (Figure 4a) to improve the model. We will show the improved model performance after it is trained on the result of the triggered assertions.

6 Conclusions

Video is a rich, rapidly growing source of high value, high volume data. We have outlined several key challenges in querying video, ranging from scalability to storage and indexing. In response, we are developing BlazeIt to demonstrate the feasibility of usable video analytics at scale. We show that that a large class of queries regarding the spatiotemporal information of objects in video can be answered in FrameQL and that BlazeIt can deliver several orders of magnitude speedup over baselines by leveraging model specialization, sampling, and learned filters. However, much work remains to make video analytics usable and scalable. We believe BlazeIt will be a valuable platform to build scalable and usable video analytics engines.

Acknowledgments

This research was supported in part by affiliate members and other supporters of the Stanford DAWN project—Google, Intel, Microsoft, NEC, Teradata, and VMware—as well as DARPA under No. FA8750-17-2-0095 (D3M), industrial gifts and support from Toyota Research Institute, Juniper Networks, Keysight Technologies, Hitachi, Facebook, Northrop Grumman, NetApp, and the NSF under grants DGE-1656518 and CNS-1651570.

7 References

[1] Cctv:
Too many cameras useless, warns surveillance watchdog Tony Porter, 2015.

[2] M. R. Anderson, M. Cafarella, T. F. Wenisch, and
G. Ros. Predicate optimization for a visual analytics database. *SysML*, 2018.

[3] W. Aref, M. Hammad, A. C. Catlin,
I. Ilyas, T. Ghanem, A. Elmagarmid, and M. Marzouk. Video query processing in the vdbms testbed for video database research. In *ACM-MMDB*. ACM, 2003.

[4] F. Arman, A. Hsu, and M.-Y. Chiu. Image processing on compressed data for large video databases. In *ACM international conference on Multimedia*. ACM, 1993.

[5] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan,
S. Reddy, and M. B. Srivastava. Participatory sensing. In *WSW*, 2006.

[6] M. Flickner, H. Sawhney, W. Niblack, J. Ashley,
Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, et al.
Query by image and video content: The qbic system. *computer*, 28(9), 1995.

[7] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu,
and J. Yosinski. Faster neural networks straight from jpeg. In *NeurIPS*, 2018.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*. IEEE, 2017.

[9] K. He, X. Zhang, S. Ren,
and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand,
M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[11] K. Hsieh, G. Ananthanarayanan,
P. Bodik, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus:
Querying large video datasets with low latency and low cost. *OSDI*, 2018.

[12] E. Hwang and V. Subrahmanian. Querying video
libraries. *Journal of Visual Communication and Image Representation*, 7(1), 1996.

[13] R. Jain and A. Hampapur. Metadata in video databases. *SIGMOD*, 1994.

[14] D. Kang, P. Bailis, and M. Zaharia. Blazeit: Fast exploratory
video queries using neural networks. *arXiv preprint arXiv:1805.01046*, 2018.

[15] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope:
optimizing neural network queries over video at scale. *PVLDB*, 10(11), 2017.

[16] D. Kang, D. Raghavan, P. Bailis, and M. Zaharia. Model
assertions for debugging machine learning. *NeurIPS MLSys Workshop*, 2018.

[17] S. Krishnan, A. Dziedzic, and E. Aaron.
Deeplens: Towards a visual data management system. In *CIDR*, 2019.

[18] M. La Cascia and E. Ardizzone. Jacob: Just a content-based
query system for video databases. In *ICASSP*, volume 2. IEEE, 1996.

[19] J. Lee, J. Oh, and S. Hwang. Strg-index: Spatio-temporal
region graph indexing for large video databases. In *SIGMOD*. ACM, 2005.

[20] T. Lee. There's growing
evidence Tesla's autopilot handles lane dividers poorly. *Ars Technica*, 2018.

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu,
and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*. Springer, 2016.

[22] C. Lu, M. Liu, and Z. Wu.
Svql: A sql extended query language for video databases. *IJDTA*, 8(3), 2015.

[23] Y. Lu, A. Chowdhery, S. Kandula, and S. Chaudhuri. Accelerating
machine learning inference with probabilistic predicates. In *SIGMOD*, 2018.

[24] V. E. Ogle and M. Stonebraker.
Chabot: Retrieval from a relational database of images. *Computer*, 28(9), 1995.

[25] A. Poms, W. Crichton, P. Hanrahan,
and K. Fatahalian. Scanner: Efficient video analysis at scale. *SIGGRAPH*, 2018.

[26] W. Tao, X. Liu, C. Demiralp, R. Chang, and M. Stonebraker.
Kyrix: Interactive visual data exploration at scale. In *CIDR*, 2019.

[27] A. Yoshitaka and T. Ichikawa. A
survey on content-based retrieval for multimedia databases. *TKDE*, 11(1), 1999.

[28] H. Zhang, G. Ananthanarayanan,
P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In *NSDI*, volume 9, 2017.