Querying Video Interactions

Leave Authors Anonymous

ABSTRACT

As deep neural nets enabled sophisticated information extraction out of images, including video frames, there has been recent interest in techniques and algorithms to allow interactive declarative query processing on objects and their interactive constraints on video feeds. In this demo, we present a system for declarative querying on real-time video streams involving objects and their interactions. The system utilizes a sequence of inexpensive and less accurate models (filters), called Progressive Filters (PF), to detect the presence of the query specified objects on frames. Since selectivities may vary as the video evolves, the system deploys a dynamic statistical test to determine when to trigger the re-optimization of the filters. In the end, the system presents a filtering approach, called Interaction Sheave (IS), which utilizes learned spatial information about objects and interactions to effectively prune frames that are not likely to contain interaction. We demonstrate that this system can efficiently identify frames in a streaming video in which an object is interacting with another in a specific way, increasing the frame processing rate dramatically and speed up query processing by at least two orders of magnitude depending on the query.

ACM Reference Format:
Leave Authors Anonymous. 2020. Querying Video Interactions. In *SIGMOD '20: ACM International Conference on Management of Data, June 14 - 19, 2020, Portland, Oregon, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

1 INTRODUCTION

Recent advances in computer vision - in the form of deep neural networks - have made it possible to query increasing volumes of video data with high accuracy. However, deep neural network inference is computationally expensive at scale. Although some have proposed systems for accelerating deep neural network queries on videos, there are still many limitations to apply them to real-world scenarios.

To understand the visual world, a machine must not only recognize individual object instances but also how they interact. Most of the state-of-the-art video query processing models focus on the queries about the number of classified objects and their location relationships [6, 7, 9, 15]. Semantic information, such as human-object interaction information, is often ignored. Fortunately, several state-of-the-art algorithms in the fields of object detection, object classification and object tracking in images and videos are proposed [3, 4, 10, 13]. They provide the ability to classify objects and detect object locations in a frame as well as track objects from frame to frame with adequate accuracy.

Figure 1: Examples of interactions.

Video monitoring and surveillance applications are of particular interest in our system, where a static camera records activity in its receptive range. Several recent works [6, 7, 9, 15] focus in these domains, to provide declarative queries based on streaming video feeds. Our work complements and enhances this research line with a focus on efficiently executing interaction query primitives, i.e., capturing interactions among query objects. Many interesting queries become possible when we can query object interactions. For example, automatically detecting frames in which a human holds a gun or a human breaks a window would be of great interest in surveillance and security applications. Figure 1 presents examples of interactions between a human and a ball (videos from Kinetics Dataset [1]). The corresponding query in SQL for the interaction at the frame of Figure 1(a) would be:

```
SELECT cameraID, frameID, C1 (F1 (HumanBox1)) AS
    HumanType1, C2 (F2 (Ballbox1)) AS BallType1,
FROM (PROCESS inputVideo PRODUCE cameraID, frameID,
    HumanBox1 USING HumanDetector, BallBox1 USING
    BallDetector)
WHERE HumanType1 = human AND BallType1 = baseball AND
    INTERACTION(HumanType1, BallType1) = THROW
```

The query employs two classifiers (C_i) to detect a *human* and a *baseball*, using features F_i extracted from the frame and checking whether the objects, once identified, are related via a THROW interaction. Obviously, from an execution perspective, it makes sense to invoke the INTERACTION predicate once the operands have been identified on the frame. We seek to automatically identify frames in a video stream where query objects interact in a specific way. The main emphasis of this system is, given a query involving objects and their interaction on a video stream, to propose algorithms to efficiently determine which frames are expected to be part of the query answer and filter out all irrelevant frames. Therefore, we try to increase the frame processing rate as frames that are considered irrelevant will not be processed further but skipped quickly. Even without filtering, the application of deep neural networks for object and interaction detection entails false positives/negatives. We will

SIGMOD '20, June 14 - 19, 2020, Portland, Oregon

ACM ISBN 978-x-xxxx-xxxx-x/xx/xx…$15.00
https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

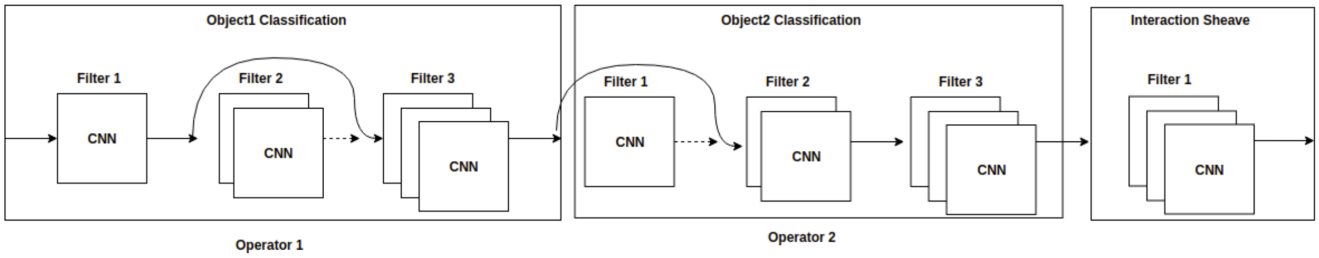

Figure 2: Example of the overall architecture.

also show the effect that filtering has on the false positive/negative rate of the techniques on this demo.

To overcome these challenges, we present a set of algorithms that dramatically increase the frame processing rate when executing the query while maintaining high accuracy. In particular, the set of algorithms are as follows.

- **Progressive Filters.** Progressive filters are a set of filters of increasing cost (i.e., an increasing number of network layers). For a particular object type, progressive filters use the selectivity of the filters to derive the optimal filter sequence. The filters should be applied to minimize the total cost of processing the video stream.
- **Triggering Re-optimization.** Since we observe that the statistical properties of certain types of objects vary over time on a video stream, we adopt a dynamic version of the Kolmogorov-Smirnoff test [12] that can trigger progressive filters algorithm when the statistical properties of the filter selectivity change. We are effectively adjusting the filter sequence, in anticipation of stable (predictable) object statistics, until the next re-optimization. Such a strategy can improve accuracy while maintaining a high frame processing rate.
- **Interaction Sheave.** Interaction sheave is a filtering mechanism for object interaction queries. It can inspect the spatial location of objects on frames and filter out frames that although contain objects relevant to the query but not promising to encompass the suitable interaction among the query objects.

In this demonstration, we showcase our Querying Video Interactions system, which is capable of handling a wide range of interaction queries. Section 2 presents the architecture of our system. Section 3 presents a description of this demonstration. Section 4 concludes this demo and discusses future work in this area.

2 SYSTEM ARCHITECTURE

Deep learning filters is an extensible module that implements our proposed filter predicates, which encompasses popular recent deep learning algorithms for object detection, texture/shape extraction, as well as algorithms for precise object localization on a video frame. It enables count estimation of objects of a specific class and location estimation of the objects in the frames. These algorithms are embedded in the parsed query representation and relayed to the query execution engine. The execution module utilizes popular deep learning frameworks to execute the query with the assistance of available GPUs. Frames that pass the filters instantiated in the query are subsequently checked with deep learning predicates and then routed to the front end for display.

An example of the architecture of our proposed filter technologies is shown in Figure 2. There are two operators, each consisting of a number of filters. The application of progressive filters for operator 1 determined that filters 1 and 3 are currently in use. Similarly, for operator two, filters 2 and 3 are in use. Frames that successfully pass the operators with a positive determination[1] that they contain the objects of interest, are tested by the Interaction Sheave. Our focus in this work is how to order filters within an object detection operator effectively. Typically, a query will check for the presence of more than one object (as in the case of the sample SQL query presented). The progressive filters can derive the least cost sequence of filters given the current selectivities of query objects, avoiding costly object detection and localization operations, and minimizes the cost per frame for object detection. For the filters employed, we can maintain their selectivities up to date by observing the result of each frame tested by the filter. For the filters that are not part of the optimal solution, we periodically (every few seconds) route a frame from the input sequence through them and obtain a selectivity estimate to trigger re-optimization. For frames that pass through the filter sequences determined by progressive filters for each object specified by the query, we have confidence that they contain the required objects. These models will derive a bounding box that encloses the location of each object as specified by the query on the frame. We then test the frame whether it relates the objects via the query specified interaction.

2.1 Progressive Filters

We test whether the objects of the class specified by the query are present in the frame before we check a frame for a query specified interaction among objects. Such a test can take place by involving state of the art object classification or detection models [2, 13]. Although such models have high accuracy, they impose large overheads in terms of processing time per frame. Prior work [7, 15] has utilized inexpensive filters that are trained apriori or on-demand to reduce processing overheads at the expense of accuracy. They are typically trained to achieve specific false positive and false negative rates, while relaying any other decision to high accuracy (but more expensive) model. Such a filter quickly determines whether a frame contains an object of a specific type or not. The premise of such filters is that a significant fraction of frames are not relevant to the

[1]Take the first operator as an example, a frame may get a positive determination by filter 1; in that case, there is no need for further processing by other filters in this operator. Filter 3 (a more accurate and expensive model in this example) is involved only if Filter 1 is unable to make a positive or negative determination for the frame.

query and can be dropped quickly. Moreover, relevant frames can be positively decided by the inexpensive filter, and only tough cases will involve an expensive model.

We observe that the pruning efficiency of such a filtering approach can be improved by deploying a series of inexpensive filters that progressively have higher accuracy and higher cost. Therefore, we realize an object detection operator as a sequence of filters. Assessing the selectivity for each of the filter predicates, we propose a practical algorithm that derives the optimal set of filter predicates for an object type to deploy. Thus, we address the inter-operator scheduling of filters to minimize detection cost. Assuming the future statistical characteristics of the video stream remain the same, the filtering combination will reduce the cost (alternatively maximize frame processing rate) to process the video stream and determine frames that contain the specific object type.

2.2 Triggering Re-optimization

Since one cannot expect the distribution of objects on video frames over time to remain the same, we propose an algorithm to assess incrementally when the statistical properties of the underlying object distribution change. When such properties change, we trigger re-optimization, deriving a new optimal filter sequence to deploy. The basic idea behind our proposal is to treat the selectivity of filters as a statistical population and employ a dynamic version of the popular Kolmogorov-Smirnov test [12]. In that way, we monitor the underlying distribution of objects relevant to the query between two epochs and trigger re-optimization when the statistical properties change.

2.3 Interaction Sheave

In order to address our ultimate objective, each frame that contains the specified objects has to be processed by a deep learning model that determines the type of interaction between objects [4, 8, 11, 14]. However, such object interaction models impose their own overheads per frame. To improve the frame processing rate even further, we develop a filtering mechanism for object interactions. The basic observation is that the interaction (e.g., human throwing a baseball) typically takes place at a specific spatial region of the frame involving the first object (i.e., the human). For each type of interaction, we could propose a model to identify such regions. Only when the second object is located within a spatial region relative to the first object, the frame is relayed to an expensive object interaction model for further processing. Thus, by processing the objects spatially, we obtain a filtering mechanism for interaction. Our final proposal combines these three techniques delivering an effective framework to process object interaction queries, efficiently improving the frame processing rate in a video stream dramatically.

3 DEMONSTRATION SETUP

This demo simulates the interaction querying process and provides various knobs for video source selection, performance monitoring and query results display and comparisons. It currently incorporates the layout of [15] with few new features. Our layout, will allow users to express human object interaction queries and compare the time and accuracy performance of our approach with the full model's performance [4].

Figure 3 presents the current state of the front end. SQL queries are defined through query definition which is shown in the left sidebar (Area A) and enhanced with UDFs to manipulate video object primitives. The Demo provides four video clips for interaction querying. Each video clip corresponds to a specific predicate. For the baseball video clip, for instance, the predicate will be *human THROW baseball*. The Demo also provides two modes - normal and slow mode. The normal mode shows the real video processing speed of our system, while it may run too fast to show the results of interaction detection. For better implementation, it also provides a slow mode to show the resulting video which is slower than the real processing speed.

The generated SQL queries, which are shown on the upper left center (Area B), are dispatched to the back end which is responsible for parsing the query and incorporating the supported querying predicates and progressive filters. It currently incorporates our proposed algorithms for filtering frames (based on humans, objects classes and interaction), allowing to express semantically meaningful video frame queries in an interactive fashion. The impact of progressive filters will be analyzed based on the total query processing time (when compared to a query that does not make use of the progressive filters but instead executes the query in a brute force manner [4] as well as the resulting frame processing rate.

The filters' ordering per batch is presented on the upper right corner (Area C). It will be updated at the end of each batch. The three boxes with different colors represent different operators and a node in a box represents a filter in the operator. As introduced in section 2.1, the application of algorithm progressive filters may employ a subset of filters. The links represent the ordering of filters that are used for the current batch. As introduced in section 2.2, for the filters that are not part of the optimal solution, we periodically route a frame from the input sequence through them and obtain a selectivity estimate as well. The links will be bold if they have been changed from the previous batch to the current. Figure 4 presents the filters ordering module. The red box and nodes represent human operators, whose Filter 1, 3, 4 and 6 are used. Similarly, the yellow and blue represent the object operator and interaction sheave respectively.

By clicking on the Optimization button, our approach will be invoked, while by clicking the Brute Force button, state of the art object detectors [5] and action recognition [4] models will evaluate the respective input video. The videos of both our approach and full model's approach will produce bounding boxes for humans and objects which participate in an interaction. Additionally, a heat-map on frames where a specific interaction exists between the human and the object. Areas D and G present the actual query results for our proposed techniques and brute force respectively. F1 score and precision/recall results are reported in Areas E (for our algorithms) and H (for brute force), complemented with detailed performance numbers (response times and frame rates) in areas F and I respectively.

4 CONCLUSION AND FUTURE WORK

This demo showcases our Querying Video Interactions system. The system we proposed constitutes a robust approach to process video streams for query specified object interactions, achieving a very high frame processing rate. This work sheds light on declarative

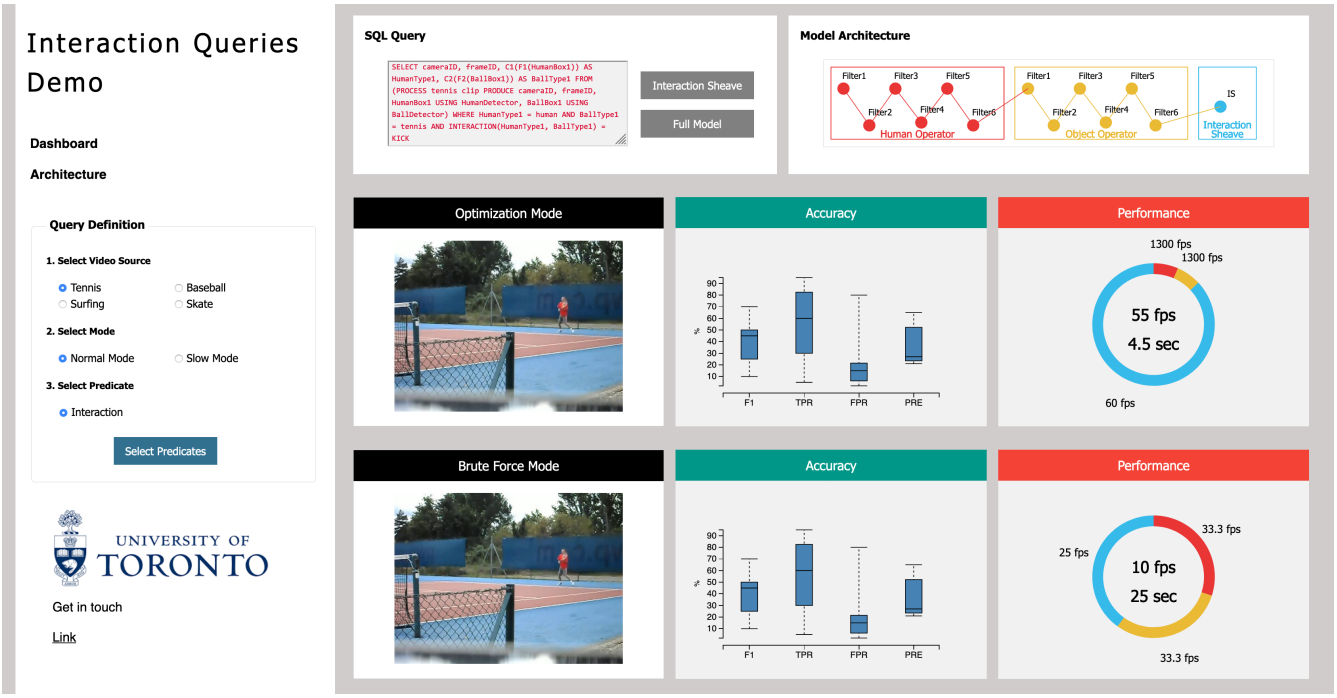

Figure 3: Video Stream Interaction Queries demo.

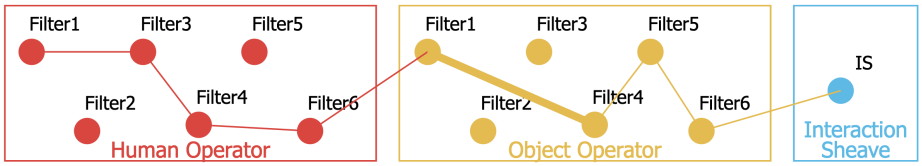

Figure 4: Filters ordering module.

query process for video streams. New query types involving spatial and temporal predicates require intensive future research. The extension of the filters for crowd counting and estimation scenarios is also necessary.

REFERENCES

[1] João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. 2019. A Short Note on the Kinetics-700 Human Action Dataset. *CoRR* abs/1907.06987 (2019). arXiv:1907.06987 http://arxiv.org/abs/1907.06987

[2] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

[4] Georgia Gkioxari, Ross Girshick, Piotr Dollár, and Kaiming He. 2018. Detecting and recognizing human-object interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8359–8367.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick. 2017. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 2980–2988. https://doi.org/10.1109/ICCV.2017.322

[6] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. BlazeIt: Fast Exploratory Video Queries using Neural Networks. *CoRR* abs/1805.01046 (2018). arXiv:1805.01046 http://arxiv.org/abs/1805.01046

[7] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. Noscope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1586–1597.

[8] Alexander Kolesnikov, Christoph H. Lampert, and Vittorio Ferrari. 2018. Detecting Visual Relationships Using Box Attention. *CoRR* abs/1807.02136 (2018). arXiv:1807.02136 http://arxiv.org/abs/1807.02136

[9] Nick Koudas, Raymond Li, and Ioannis Xarchakos. 2020. Video Monitoring Queries. *Proceedings of IEEE ICDE* (2020).

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[11] Chih-Yao Ma, Asim Kadav, Iain Melvin, Zsolt Kira, Ghassan AlRegib, and Hans Peter Graf. 2017. Grounded Objects and Interactions for Video Captioning. *CoRR* abs/1711.06354 (2017). arXiv:1711.06354 http://arxiv.org/abs/1711.06354

[12] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.

[13] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1409.1556

[14] Oytun Ulutan, Swati Rallapalli, Mudhakar Srivatsa, and B. S. Manjunath. 2018. Actor Conditioned Attention Maps for Video Action Detection. *CoRR* abs/1812.11631 (2018). arXiv:1812.11631 http://arxiv.org/abs/1812.11631

[15] Ioannis Xarchakos and Nick Koudas. 2019. SVQ: Streaming Video Queries. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD '19)*. ACM, New York, NY, USA, 2013–2016. https://doi.org/10.1145/3299869.3320230