Semantic Fast-Forwarding for Video Training Set Construction

Ziqiang Feng*
Carnegie Mellon University

Mahadev Satyanarayanan
Carnegie Mellon University

Abstract

We introduce the concept of *semantic fast-forwarding* of video streams for efficient labeling of training data for activity recognition. We show that this concept can be realized by combining deep learning within individual frames, with spatial and temporal entity-relationship reasoning about detected objects. We describe a prototype that implements this concept, and present preliminary experimental results on its feasibility and value.

CCS Concepts: • **Human-centered computing** → *Interactive systems and tools*; *Empirical studies in ubiquitous and mobile computing*; • **Computing methodologies** → *Computer vision*; *Machine learning*; • **Information systems** → *Information retrieval*.

Keywords: Eureka, edge computing, activity recognition, search, deep learning

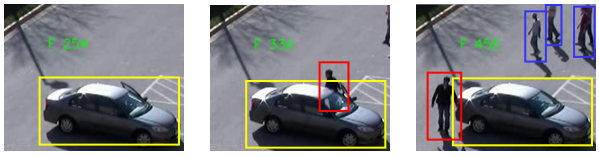
ACM Reference Format:
Ziqiang Feng and Mahadev Satyanarayanan. 2023. Semantic Fast-Forwarding for Video Training Set Construction. In *The 24th International Workshop on Mobile Computing Systems and Applications (HotMobile '23), February 22–23, 2023, Newport Beach, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3572864.3580331

1 Introduction

An astonishing amount of video data is captured every day via smartphones, drones, vehicular cameras, and other sources. In 2020, it was estimated that 720,000 hours of video are shared online daily [22]. What will we do with all this video? Its long-term value as archival data lies in the ability to search it retrospectively. This implies content-based indexing.

At the granularity of individual video frames, the creation of object detectors/classifiers based on deep neural networks (DNNs) is a mostly solved problem. While extensive research continues, the state of the art is already good enough for practical use. However, there are some concepts that cannot be reduced to the granularity of a single frame. They only make sense in a temporal context, across multiple frames.

Consider the task of finding instances of the event "a person getting out of a vehicle." Figures 1(b) and 1(c) show two examples. A DNN can be created to reliably detect "person" and "vehicle" in individual frames, but the action of opening a car door to get out is inherently temporal — Figures 1(b) and 1(c) could equally well show persons getting *into* (instead of *out of*) a car. There are countless examples of this kind, where the temporal dimension is an integral aspect of the concept: e.g., pitching a ball in baseball versus bowling it in cricket; dancing a waltz versus a tango; a fist bump versus a punch; and so on. *Activity Recognition* [5, 9, 24, 30] focuses on



(a) Frame 254 (b) Frame 336 (c) Frame 456

Yellow box: vehicle; Red box: person getting out of vehicle;
Blue box: irrelevant person; Source: VIRAT dataset [18]

Figure 1. Examples of "Get-Out-of-Vehicle" Event

creating DNNs to detect such concepts. Unfortunately, constructing a training set for activity recognition is painful today. It requires human labelers to spend many hours watching raw video. Can we do better? Can we increase their productivity?

In this paper, we introduce the concept of *semantic fast-forwarding:* i.e., automatically skipping a long stretch of video that is definitely irrelevant, and only stopping at the beginning of a plausibly relevant video segment. Note that "plausibly relevant" is a weak descriptor. It implies that the human labeler may have to tolerate many false positives (FPs) relative to the number of true positives (TPs) discovered. But that is better than having to also watch the long irrelevant stretch of video preceding the FP. Once a large-enough training set has been assembled, an accurate DNN for activity recognition can be trained. Our focus is on bootstrapping this training set.

We show that semantic fast-forwarding can be implemented by unifying two well-known mechanisms:

- DNN-based object detection within individual frames,
- entity-relationship (ER) reasoning about detected objects, both spatially within individual frames and temporally across multiple frames.

Unifying these two very different mechanisms, one grounded in machine learning and the other in classical databases, provides the right balance of specificity and flexibility needed for expressing new video concepts. We have built a system called *Eureka* that embodies this approach, and present early results in this paper.

2 Modeling Spatial-Temporal Events

Our work focuses on actions that can be characterized as *coarse-grained inter-object spatial-temporal relationships.* Complex events are recursively defined in terms of simpler events. The starting point is object detection on a single frame, which discovers the simple event "Presence of object O at position (x, y) in frame t." Grouping all such events temporally across consecutive frames is a *tracking* problem, and produces the object's motion trajectory over time. This can be interpreted as the complex event: "Object O appears in the video at time-space sequence (t, x_t, y_t), $(t{+}1, x_{t+1}, y_{t+1}), \ldots, (t{+}n, x_{t+n}, y_{t+n})$."

Building on this, more complex events can be specified. For example, we can characterize a "Get-Out-of-Vehicle" event as "An object of type *person* first appears from an object of type *vehicle* located at (x, y) in frame t." Notice that "from" expresses a spatial relationship, while "first" expresses a temporal relationship. In

*Now at Google.

HotMobile '23, February 22–23, 2023, Newport Beach, CA, USA

ACM ISBN 979-8-4007-0017-0/23/02.
https://doi.org/10.1145/3572864.3580331

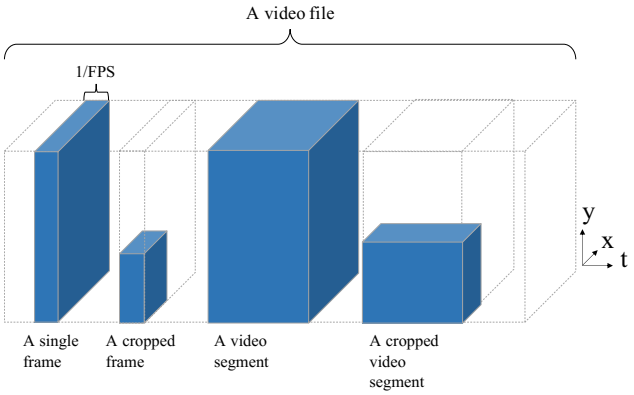


Figure 2. Spatial-Temporal Interval

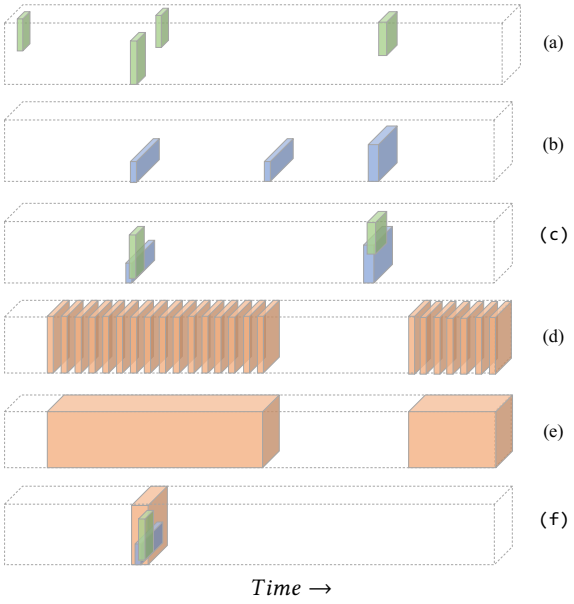


Figure 3. Complex Event

real-world video data, there are likely many persons and many vehicles, but few of them constitute the above event. Finding the matches requires comparing them under a set of constraints. These constraints may be not only on their time and space, but also on their visual features and specific instance identities.

The basic abstraction in Eureka is a *spatial-temporal interval*, abbreviated to just "interval." It can be viewed as the "container" of an event. It specifies some key-value attribute that remains invariant throughout a volume defined by a *cropped video segment* (Figure 2). This attribute can be any one of a wide of range possibilities such as: (a) raw pixels arrays representing visual content; (b) feature vectors computed on the pixel content; (c) meta-data extracted by computer vision algorithms (e.g., labels and confidence scores); (d) pointer to another interval instance; or nested structure of any of the above. An *interval stream* is a sequence of intervals that are in ascending order of time. An *operator* (Table 1) takes interval stream(s) as input, and outputs interval stream(s). A complex query can be composed as a DAG (directed acyclic graph) of operators.

Relational	map, reduce, filter, sort, join, flatten
Data transformation	VideoToFrames, FramesToVideo, ImageCrop, VideoCrop
Spatial	IoU, merge_span, above, below
Temporal	during, before, after, coalesce
Visual features	RGB color histogram, perceptual hashing, SIFT key points, image classification, object detection

Table 1. Example Operators and Predicates

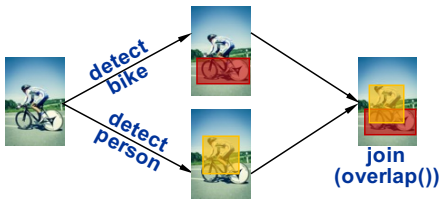
Figure 3(a) through (f) illustrate these steps towards expressing the complex query "a person riding a bike during a red traffic light." DNN-based object detection on individual video frames produces the atomic events "person instances" (3(a)) and "bike instances" (3(b)). A `Join` operator with the spatial predicate `Overlap()` finds matching person-bike pairs that indicate the event "bicyclist" (3(c)). Likewise, red lights in individual frames are marked (3(d)) (based on object detection and color filter) and then coalesced into "periods of red light" (3(e)). Finally, using the temporal predicate `During()` to join the "bicyclist" event stream (3(c)) with the "periods of red light" event stream (3(e)) selects exactly those bicyclists who ride during a red light (3(f)).

3 Coping with Uncertainty

Classic ER reasoning, based on relational logic, is devoid of uncertainty. In spite of many efforts to incorporate uncertainty and probabilistic reasoning into ER models [23], they remain esoteric. In contrast, DNNs inherently embody uncertainty — the result is expressed as a distribution of probabilities across classes.

Hence, Eureka offers two different approaches to describing within-frame relationships. The first approach describes events by *construction*. We start by detecting by individual objects, and then we examine whether they relate to each other in a certain manner. Figure 4(a) illustrates this approach.

The second approach is *guess-and-verify*, where the presence of one object leads to a guess regarding the identity and location of a second object. This is illustrated in Figure 4(b). This approach has two advantages. First, we can use image classification instead of object detection in the verify step. Since classification is typically an order of magnitude faster than detection, this can be a significant



(a) By Construction

(b) Guess and Verify

Figure 4. Alternative Approaches to Detecting Bicyclist

speedup. It is also useful in situations where only a classifier, but not a detector, is available for a target. Second, the verify step only needs to focus on a small region of interest rather than the whole frame, making it less prone to clutter in the background.

4 Implementation and Optimization

Eureka separates *expression* and *execution* of semantic fast-forwarding. Using a GUI, the user (i.e., the human labeler) describes the desired target concept using the abstractions of Section 2. Like a database query optimizer, Eureka compiles the query into an execution plan that optimally utilizes multi-core CPUs, main memory, and GPUs on multiple servers across which the raw video data has been sharded. However, Eureka's goal is not to run a query to completion in the shortest time, but to present the next result as soon as possible for the user to inspect and label. Starving the user of results, or overwhelming her with FPs are both to be avoided. We describe below the video-specific features that distinguish Eureka from classic SQL database optimizers.

4.1 Maintaining the Stream Invariant

The interval stream model in Eureka maps natively to live video processing, as discovered events arrive in temporal order and may never end. Moreover, maintaining the stream assumption throughout the system allows for optimized implementation of certain operators. For example, in the worst case, the `Join` operator may produce the Cartesian product of its inputs. Yet, we observe that the more constrained `WindowedJoin` operator is sufficient in most use cases, as one typically wants to relate an event to other events that happen in proximate time. The `WindowedJoin` operator only considers input pairs that are within a small temporal window of each other. As Algorithm 1 shows, the stream assumption of the join inputs allows an efficient implementation that only needs to maintain a small input buffer, and releases obsolete buffered inputs as soon as new ones run out of their window.

Algorithm 1: Windowed Join of Two Interval Streams

```
Input: IntervalStream leftIn, IntervalStream rightIn
Input: int window, function predicateFn, function mergeFn
leftBuffer ← List();
rightBuffer ← List();
while True do
    // Get a new input from the left upstream
    iL ← leftIn.getNext();
    for iR in rightRuffer do
        if iR.t2 < iL.t1 − window then
            Remove iR from rightBuffer ;
        else if iR.t1 < iL.t2 + window then
            if predicateFn(iL, iR) = True then
                Output(mergeFn(iL, iR));
    leftBuffer.append(iL);
    // Get a new input from the right upstream
    iR = rightIn.getNext();
    ... // Mirror operations above
```

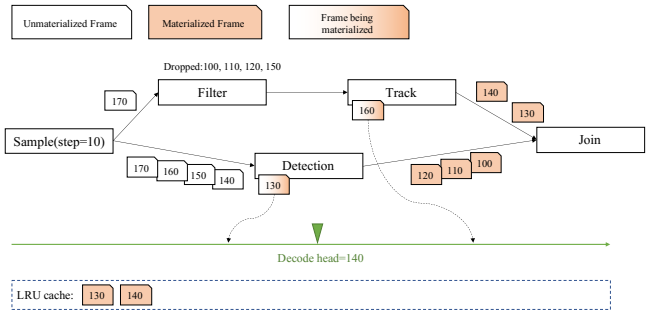


Figure 5. Avoiding Redundant Decoding and Seeks

4.2 Exploiting Parallelism

Eureka exploits three levels of parallelism, while remaining constrained by sequential dependency between frames:

- First, it exploits *inter-file/inter-stream parallelism*, by creating separate instances of the user query for each input file or live stream, and executes them in parallel.
- Second, it exploits *inter-operator parallelism* by running each operator in its own thread. Using a producer-consumer execution model, the operator threads run in parallel, optimally overlapping CPU-, GPU-, and memory-bound processing.
- Third, Eureka exploits two sources of *intra-operator parallelism:* (1) multi-threaded implementation of computer visions algorithms (e.g., from OpenCV and OpenBLAS); (2) data-parallel versions of generic operators (e.g., `ParallelMap`, `ParallelFilter`). These data-parallel operators typically use a thread pool to concurrently process inputs, thus exploiting *inter-interval parallelism.*

4.3 Provenance and Late Materialization

Eureka queries often derive visual data by spatial-temporal cropping. Such derivations may be recursive: i.e., crops inside a crop. Eureka tracks provenance via reference to the upper-level element from which a crop is created, much like ".." in a Linux directory.

Tracking provenance enables *late materialization* of decoded visual data. The crop made by an operator is *logical.* A new `Interval` object is created with new (reduced) `bounds`, and its `parent` pointing to the source. The RGB array is not materialized immediately. Instead, the reserved attribute name `rgb` is replaced by a callback function that performs the actual cropping using its source's RGB data. Since the source itself may not have been materialized, callback and cropping are performed recursively, on demand.

4.4 Video Decoder and LRU Frame Cache

Parallel executing operators and late materialization pose challenges for decoding video frames. First, the cost of video decoding is non-trivial. Second, for a large query graph, it requires a lot of DRAM to hold "on-the-fly" decoded frames that await processing by downstream operators. Third, video decoders are stateful and have tape-like performance: decoding forward in time is efficient, while winding-back or random seek is costly.

We address these issues by sharing a single video decoder between multiple operators, and introducing an LRU frame cache. Figure 5 shows how these mechanisms work together during the execution of a simple query. While the decode head is at frame 140, the `Track` and `Detection` operators are trying to decode frame 130

(a) *get-out*: person getting out of a vehicle

(b) *get-in*: person getting into a vehicle

(c) *loading*: person (un)loading objects from/into vehicle

(d) *carry-bag*: person carrying bag

(a) *get-out* (b) *get-in*

(c) *loading* (d) *carry-bag*

Figure 6. VIRAT Dataset Events and Examples

and 160, respectively. Without the frame cache, an expensive "wind-back" is necessary and frame 130 needs to be decoded again. With the frame cache, cached frame 130 can be returned to `Detection` without moving the decode head; similarly for the subsequent frame 140. The frame cache not only reduces redundant decoding of the same frame, but also reduces DRAM used by "on-the-fly" intervals by passing cached frames by reference, rather than by value. A callback function materializes the frame's RGB data on demand.

5 Evaluation

5.1 Metrics

For rare events, labeling effort is dominated by the time taken to collect a sufficient number of TPs for the training set, as negative examples are usually easy to find. This leads to the concept of *productivity* as a key metric:

$$\text{Productivity [\#/hr]} = \frac{\text{\# TPs discovered}}{\text{Total length of video viewed}}$$

If ground truth is known, we can also calculate *(event) recall:*

$$\text{Event recall [\%]} = \frac{\text{\# TPs discovered}}{\text{\# Ground Truth Event instances}}$$

Note that *precision,* which is a simple ratio of counts, is a misleading metric for video. An FP that is just one minute long is less painful than an FP that is an hour long. This is in contrast to frames, for which the pain of an FP is constant. The productivity metric takes the length of FP video into account. An ideal method would achieve 100% recall with the highest possible productivity. Nonetheless, this state of affairs could only exist at the end of the Eureka workflow, *after* sufficient data has been labeled to train an accurate model. Without a golden model, event-based Eureka provides a richer tradeoff space between recall and productivity, and allows more effective tradeoffs to be made relative to alternative approaches.

5.2 Datasets and Tasks

Our evaluation uses two datasets. The first is the VIRAT [18] dataset (329 files totaling 8.6 hours) of video captured from real-life public surveillance cameras at parking lots, school campus, streets, etc. The second is the Okutama [1] dataset (33 files totaling 0.55 hours) of drone-captured video of actors performing scripted scenes in a playground. Both datasets have multiple actions happening simultaneously in a scene. We define four events of interest for VIRAT (Figure 6) and two for Okutama (Figure 7).

(a) *pushing*: person pushing or pulling object

(b) *handshake*: persons shaking hands

(a) *pushing* (b) *handshake*

Figure 7. Okutama Dataset Events and Examples

5.3 Frame-based and Event-based Heuristics

Semantic fast-forwarding requires the user to guide the fast-forwarding using heuristics. These heuristics can be frame-based (i.e., only involving spatial relationships on a single frame) or event-based (i.e., involving temporal invariance across multiple frames of a single-frame concept). Table 2 and 3 present these heuristics for the events from the VIRAT and Okutama datasets. As discussed in Section 2, we expect event-based heuristics to be more effective, but that needs to be experimentally validated.

Event	Frame-based	Event-based
get-out	Person + vehicle	Person emerges from a stopped vehicle
get-in	Person + vehicle	Person disappears into a stopped vehicle
loading	Person + vehicle	Person stays next to vehicle for > 2s
carry-bag	Person	Person attached to bag for > 2s

Table 2. VIRAT Semantic Fast-Forwarding Heuristics

Event	Frame-based	Event-based
pushing	Person + non-person object	Person + non-person object move together for > 2 seconds
handshake	≥ 2 persons	Two persons' trajectories meet for > 2 seconds

Table 3. Okutama Semantic Fast-Forwarding Heuristics

5.4 Preliminary Case Study Results

We conducted a preliminary case study on the effectiveness of semantic fast-forwarding. Its goal is to provide initial insights for designing a formal user study. Since the subjects of the case study were the authors themselves, no IRB approval was needed. We used the Eureka operators in Table 1, which include a common (but not exhaustive) set of computer vision building blocks. Emerging computer vision techniques (e.g., pose estimation) can be incorporated into Eureka, but are orthogonal to our focus in this paper.

We use three labeling methods:

- *Brute Force:* a user views all video data, and labels all event instances in it. This is painfully slow and laborious. However, assuming that the user does not tire or make mistakes, it defines ground truth.
- *FB-Eureka:* the video fast-forwards until a frame embodying the specified heuristic is encountered. The user begins viewing at that frame, and continues until the event is complete (in case of a positive), or until it is clear to the user that this is a negative.
- *EB-Eureka:* the video fast-forwards until a segment embodying the event-based heuristic is encontered. That segment is then shown to the user.

Table 4 presents the VIRAT results. In this dataset, there are often long periods when no event-relevant object is present in a frame — e.g., a parking lot is empty of people and vehicles. As a result, FB-Eureka is able to improve productivity significantly: up to 1.8X that of BruteForce. EB-Eureka is able to eliminate many more FPs, resulting in up to 5.0X the productivity of BruteForce. The imperfect recall of DNN-based object detection affects the recall of both FB-Eureka and EB-Eureka. For *get-out* and *get-in*, the event-based predicates are based on the start/end point of a person's trajectory. These require object tracking, which is not perfect. This leads to an additional 17–19% loss of recall in EB-Eureka, relative to FB-Eureka. This small sacrifice in recall is likely tolerable for creating a training set, because it comes with a large gain in productivity. Our qualitative impression on examining the missed TPs is that they do not substantially enlarge the diversity of the training set. Quantitatively establishing this would require training and comparing the AUC (Area Under the Curve) scores of a range of DNN models. This remains future work.

For *loading*, the object being loaded/unloaded is often occluded by the person or the vehicle. Therefore, we exclude it from our predicate, leading to many FPs and hence low productivity for FB-Eureka. For *carry-bag*, only a 5.8-hour subset of the VIRAT videos are annotated with ground truth, so our evaluation only used that subset. The bag being carried is often so small that it cannot be detected on the frame level. Therefore, with FB-Eureka we only predicate on the presence of persons. This explains FB-Eureka's high event recall but low productivity. By contrast, EB-Eureka allows detecting the bags in a zoomed-in region, which has a higher chance of success.

The Okutama dataset is more challenging than VIRAT for two reasons. First, because the content of the videos is performed by actors according to a script, there are no long stretches of "boring" footage as in typical real-life video. Second, the aerial viewpoint and abrupt turns of drone-captured video make detection and tracking challenging. The results in Table 5 reflect these observations. For FB-Eureka, poor recall offsets the benefit gained from pruning non-event frames; this hurts productivity slightly for both *pushing* and *handshake*. EB-Eureka is still able to achieve 1.6x productivity for *pushing*, and essentially breaks even for *handshake*.

These results suggest that semantic fast-forwarding is especially valuable for real-life video rather than curated video, such as that found in movies or uploaded to YouTube. The latter have already been curated to strip out long "boring" stretches, thus resulting in video with less opportunity for simple heuristics. Also, as mentioned earlier, a more rigorous user study to validate our results will be worth the effort.

	Labeling effort [hr]	TPs found [#]	Event recall [#/GT]	Productivity [#/hr]
get-out				
BruteForce	8.60	97	100%	11
FB-Eureka	3.86	76	78%	20
EB-Eureka	1.05	59	61%	56
get-in				
BruteForce	8.60	111	100%	13
FB-Eureka	3.86	77	69%	20
EB-Eureka	1.06	55	50%	52
loading				
BruteForce	8.60	80	100%	9
FB-Eureka	3.86	37	46%	10
EB-Eureka	0.76	33	41%	43
carry-bag				
BruteForce	5.80	822	100%	141
FB-Eureka	4.08	809	98%	198
EB-Eureka	0.97	556	68%	573

Table 4. Labeling Effort for VIRAT Dataset Events

	Labeling effort [hr]	TPs found [#]	Event recall [#/GT]	Productivity [#/hr]
pushing				
BruteForce	0.55	134	100%	242
FB-Eureka	0.43	100	75%	234
EB-Eureka	0.20	77	57%	395
handshake				
BruteForce	0.55	71	100%	128
FB-Eureka	0.39	48	68%	124
EB-Eureka	0.37	48	68%	130

Table 5. Labeling Effort for Okutama Dataset Events

6 Context and Related Work

Research in video analytics systems can be viewed within a 4-level taxonomy, in which higher levels involve extraction of deeper knowledge. Each level poses unique challenges in terms of system design and algorithms.

Level 0 (L0): treats video as a collection of independent static images. No temporal relationships are considered. Standard image analytics tasks, such as classification and object detection, can be applied to video data with little change. Diamond [7] and GigaSight [20, 21] are examples of L0 systems.

Level 1 (L1): exploits certain attributes of video data, such as temporal similarity, in order to reduce frame processing cost and improve application-level utility. The majority of recent work falls in this category. Table 6 lists some of the most important recent L1 systems. It can be seen that search targets in this table are all nouns and not temporal in nature. These systems also assume a golden

System	Search targets considered
NoScope [13]	Person, bus, car
BlazeIt [12]	Bus, car, boat
Chameleon [11]	Targets from the COCO data set [15]
Focus [6]	Targets from the COCO data set
Mainstream [10]	Pedestrian, bus, car, train
FilterForward [2]	Pedestrian, pedestrian wearing red
Drone split compute [26]	Person, car, raft, elephant
Gabriel [27]	Lego blocks, ping-pong table, human face
EVA [29]	Vehicle

Table 6. Recent L1 Video Analytics Systems

model exists as a starting point for finetuning and optimization, whereas such a model does not exist in Eureka's use cases.

Level 2 (L2): systems treat video as a spatial-temporal container of events. Because of the need to maintain temporal information, L2 analytics poses additional challenges in terms of algorithmic design and system optimization. Eureka is an L2 system. Snorkel [19] offers part of EB-Eureka's functions that only deal with metadata, without streaming or content-based processing. Activity recognition [5, 9, 24, 30] based on DNNs trained via supervised learning can be seen as a form of L2 analytics. Object tracking [28] also maps to L2.

Level 3 (L3): involves fusion of multi-sensor multi-modality data. This includes video data from multiple cameras, as well as other sensor types such as audio sensors and thermometers. Some researchers have started to consider this problem [8], but overall it is still a nascent area.

In the above 4-level taxonomy, tasks of a lower level can be formulated as a degenerated case of a higher level. For example, L1 frame-based analysis can be seen as finding one-frame L2 events (e.g., "object X appears in frame t"); L2 event analysis in a video stream can be seen as L3 n-camera fusion where $n = 1$.

While Eureka builds on recent work in deep learning, it also leverages much older work. As mentioned in Section 1, Eureka combines DNN-based object detection within individual frames with ER reasoning about detected objects, both spatially within individual frames and temporally across multiple frames. The idea of specifying an object detector as a set of geometric relationships between parts dates back nearly 50 years to Fischler and Elschlager's pictorial structures [4]. A more modern realization of this approach is the 2009 work on deformable part models (DPM) [3]. Although deep learning has muted the value of these old techniques, Eureka shows that they are still valuable in its heuristics.

Finally, Eureka complements work in *few-shot learning (FSL)* [14, 16, 17]. The goal of FSL is to "make do" with a tiny training set. Eureka, on the other hand, is all about enlarging the training set. One can view a weak FSL model for spatio-temporal events as an excellent heuristic for Eureka. Once a large training set is assembled, supervised learning in activity recognition can complete the job.

7 Conclusion and Future Work

Indexing the vast quantities of video that are captured daily is a herculean effort. Well-known mechanisms such as map-reduce, combined with large GPU-equipped computing clusters (possibly edge-based) are necessary, but not sufficient. Also needed are DNNs for activity recognition. Over time, we expect that an ever-growing collection of activities of increasing complexity will be of interest. We have focused on the construction of training sets for creating these DNNs. We have introduced the concept of semantic fast-forwarding, and described a system that is able to significantly improve human productivity in labeling. Our preliminary results suggest that this is a promising path forward. Techniques in this paper may be repurposed for pre-filtering in mobile-edge computing systems such as [25, 26]. We leave it as future work to explore their applications in those contexts.

Our preliminary evaluation in this paper has focused on the productivity improvement enabled by semantic fast-forwarding. This evaluation can be extended in many ways. There is further opportunity for improving performance using techniques from systems such as GigaSight [20] and EVA [29]. There is also an opportunity to study the kinds of heuristics that are used by Eureka users on real tasks. This would likely entail a formal user study, which is important future work in this domain.

Acknowledgements

We thank Anh Nguyen, our shepherd, and the anonymous reviewers for their guidance in improving this paper. This research was supported by the National Science Foundation (NSF) under grant number CNS-2106862. This work was done in the CMU Living Edge Lab, which was supported by gifts from Intel, ARM, Vodafone, Deutsche Telekom, CableLabs, Crown Castle, InterDigital, Seagate, Microsoft, VMware and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the view(s) of their employers or the above funding sources.

References

[1] M. Barekatain and et al. Okutama-action: An aerial view video dataset for concurrent human action detection. In *IEEE Computer Vision and Pattern Recognition Workshops*, pages 2153–2160, 2017.

[2] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. R. Dulloor. Scaling video analytics on constrained edge nodes. In *SysML*, 2019.

[3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 2009.

[4] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1), 1973.

[5] R. Ghosh. Deep learning for videos: A 2018 guide to action recognition. https://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review, 2018. Last Accessed January 23, 2021.

[6] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation*, 2018.

[7] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. R. Ganger, E. Riedel, and A. Ailamaki. Diamond: A storage architecture for early discard in interactive search. In *Proceedins of USENIX Conference on File and Storage Technologies*, 2004.

[8] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez. Scaling video analytics systems to large camera deployments. In *Proc. of the 20th Intl. Workshop on Mobile Computing Sys. and Apps.*, 2019.

[9] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2012.

[10] A. H. Jiang, D. L.-K. Wong, C. Canel, L. Tang, I. Misra, M. Kaminsky, M. A. Kozuch, P. Pillai, D. G. Andersen, and G. R. Ganger. Mainstream: Dynamic stem-sharing for multi-tenant video processing. In *2018 USENIX Annual Technical Conference USENIX ATC 18)*, 2018.

[11] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 253–266. ACM, 2018.

[12] D. Kang, P. Bailis, and M. Zaharia. BlazeIt: Fast Exploratory Video Queries using Neural Networks. arXiv:1805.01046v1, 2018.

[13] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: optimizing neural network queries over video at scale. *Proceedings of the Very Large Data Bases*, 2017.

[14] J. Lei Ba, K. Swersky, S. Fidler, et al. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*. Springer, 2014.

[16] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. In *ECCV*, 2016.

[17] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016.

[18] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*. IEEE, 2011.

[19] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282, 2017.

[20] M. Satyanarayanan, P. Gibbons, L. Mummert, P. Pillai, P. Simoens, and R. Sukthankar. Cloudlet-based Just-in-Time Indexing of IoT Video. In *Proceedings of the IEEE 2017 Global IoT Summit*, Geneva, Switzerland, 2017.

[21] Simoens, P., Xiao, Y., Pillai, P., Chen, Z., Ha, K., Satyanarayanan, M. Scalable Crowd-Sourcing of Video from Mobile Devices. In *Proceedings of the 11th International Conference on Mobile Systems, Applications, and Services (MobiSys 2013)*, June 2013.

[22] T. Thomson, D. Angus, and P. Dootson. 3.2 billion images and 720,000 hours of video are shared online daily. Can you sort real from fake? *The Conversation*, November 2020. https://theconversation.com/3-2-billion-images-and-720-000-hours-of-video-are-shared-online-daily-can-you-sort-real-from-fake-148630.

[23] A. Urrutia, J. Galindo, L. Jimenéz, M. Piattini, and J. Pries-Heje. Data modeling dealing with uncertainty in fuzzy logic. In *The Past and Future of Information Systems: 1976–2006 and Beyond*. Springer, 2006.

[24] A. Vahdat, K. Cannons, G. Mori, S. Oh, and I. Kim. Compositional models for video event detection: A multiple kernel learning latent variable approach. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[25] J. Wang. *Scaling Wearable Cognitive Assistance*. PhD thesis, CMU-CS-20-107, CMU School of Computer Science, 2020.

[26] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 159–173. IEEE, 2018.

[27] J. Wang, Z. Feng, S. George, R. Iyengar, P. Pillai, and M. Satyanarayanan. Towards Scalable Edge-Native Applications. In *Proceedings of the Fourth IEEE/ACM Symposium on Edge Computing (SEC 2019)*, Washington, DC, November 2019.

[28] Z. Wang, L. Zheng, Y. Liu, and S. Wang. Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2019.

[29] Z. Xu, G. T. Kakkar, J. Arulraj, and U. Ramachandran. Eva: A symbolic approach to accelerating exploratory video analytics with materialized views. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD '22, page 602–616, New York, NY, USA, 2022. Association for Computing Machinery.

[30] F. Yang, S. Sakti, Y. Wu, and S. Nakamura. Make skeleton-based action recognition model smaller, faster and better. In *ACM International Conference on Multimedia in Asia*, 2019.