Querying for Actions Over Videos

Daren Chao
University of Toronto
drchao@cs.toronto.edu

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

Abstract

Several recent works address issues related to declarative query processing over videos. Research to date encompasses issues related to the presence of certain types of objects on video frames. In this paper we introduce algorithms to enable queries involving objects as well as actions over videos.

We first consider the case of queries involving a number of objects and an action over a streaming video assuming no preprocessing is possible. In this case we introduce algorithms (SVAQ and SVAQD) that are able to identify on the fly video segments that satisfy the query predicates. We then consider the offline case, during which a number of videos are amenable to a one time preprocessing in order to answer adhoc queries of the type considered in this paper. In this case we detail the types of preprocessing required and introduce algorithm RVAQ that produces effectively top-k video segments satisfying the query for user specified ranking functions. Our overall approach and algorithms are grounded on sound statistical principles, being adaptive to the underlying video properties, while making use of state of the art black box object and action detection models.

We present the results of a thorough experimental evaluation utilizing real benchmark videos and real life movies. Our results indicate that our algorithms attain superior accuracy while offering clear performance advantages compared to other applicable approaches.

PVLDB Reference Format:
Daren Chao and Nick Koudas. Querying for Actions Over Videos. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Availability Tag:
The source code of this research paper has been made publicly available at http://vldb.org/pvldb/format_vol14.html.

1 Introduction

Video data are ubiquitous. Video streaming dominates internet traffic (nearly 80% of internet traffic is video equivalent to 282 EB per month, increasing rapidly [9]). Cameras that are prevalent in computing devices have transformed video production into a commodity. As a result, it is easy to generate, upload, and disseminate video data, whose automated analysis becomes a pressing concern.

Given the importance and significance of video data, numerous recent research efforts aim to develop suitable algorithms and techniques to analyze and query videos [24, 27, 28]. Advances in Machine Learning (ML) are offering highly sophisticated algorithms to classify video frames [29, 39, 42], detect and track objects across video frames [19, 22, 23, 35–37, 47, 52], identify and classify actions and interactions among objects present in video frames [3, 5, 14, 15, 20, 41, 43, 46]. These are fairly active research areas in the Computer Vision community. At the same time, in the data management community, there has been increasing research interest in the development of declarative query processing techniques over videos [6, 24, 25, 27, 28, 48, 49]. The basic idea is to expose the outcome of sophisticated ML vision algorithms (such as object detection, action detection, etc) as first class citizens to a declarative query interface. Such queries are executed either in an *online* manner (as the video is streaming, assuming no pre-processing) or in an *offline* manner (over one or more long videos suitably preprocessed). In either case, the query semantics are the same. In the online case, the query identifies the frames or frame sequences where the query condition(s) are true and in the offline case, the query reports the associated frames or frame sequences where the query is true in one or more videos.

In this paper, we aim to overcome the challenges associated with making *actions* in videos a first class citizen for declarative query processing. Action detection and classification [5, 50] has been an active area of research in the ML Vision community. Such algorithms accept a sequence of video frames and classify the frame sequence as containing a specific action out of a list of possible actions the model is trained on. For example, they can classify a frame sequence as containing a human jumping or a human playing guitar, etc. Equivalently action detection models [2, 14, 18, 40, 51] can detect the precise frame sequence that an action is taking place (typically labeling the frames at the start and the end of the action sequence). Such models are fairly mature in the ML community but still constitute active research topics.

Action detection/classification models are typically trained end to end. As such they require numerous data sets with suitable types of actions for training. The ML community over the years has developed a plethora of such data sets containing different types of actions to train the models. A significant challenge to utilize action detection models as part of query processing lies in the interaction of the action detection model with other related query predicates. Consider for example a query seeking to detect a frame sequence that depicts a robot dancing while a car is visible in the frame sequence as depicted in Figure 1. From an action detection perspective, the typical models for detecting actions are trained on the actions themselves (e.g., Robot Dancing) and are not aware of other objects. Thus, an action detection module that is trained to recognize a robot dancing cannot be used to answer such a query in an effective manner (as it necessitates a post-processing step). One could in principle train a model to recognize actions that also contain a car in the frame sequence. Such an approach is not scalable, however, as it would require a model for any possible

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

combination of query predicates present in queries, which is clearly impractical.

A different approach would be to decouple the detection of the action from the detection of the other objects mentioned in a query. Namely, one could detect a sequence of frames containing the desirable action using an action detection model, then utilize an object detection algorithm to detect frame lists that contain the desired objects in the query and intersect them. Such an approach however, requires several parameters/thresholds to be decided apriori. For example for how many frames the action frame sequence should overlap the various frame lists containing objects in order to declare a query match? Object detection algorithms are typically noisy (yield false positives and negatives), so how would such noise affect the thresholds? Is it possible or even feasible to choose or decide such thresholds before a query executes or to tune such thresholds for each different query predicate? In this paper, we place this problem into perspective and we propose solutions for both an *online* setting in which query results have to be reported as the video streams and an *offline* setting in which queries are issued against a specified repository of videos that have been suitably pre-processed.

For the online case, we present a streaming algorithm, called SVAQ, to identify segments (parts) of a **V**ideo **S**tream that satisfy the **Q**uery predicates both in terms of the **A**ction specified and in terms of any additional query specified object predicates. To eliminate the burden of setting query thresholds manually, and cope with the inaccuracies inherent to action and object detection models, we introduce a theoretically grounded approach based on scan statistics [21, 32, 44]. This approach first estimates the distribution of predictions made by each individual model involved in the query when the query predicates are not satisfied. Then, for each query predicate, it computes what constitutes a significantly large number (a critical value, k_{crit}) of positive predictions (the query predicate is satisfied) conducted by each individual model in a sequence of frames. These are utilized to synthesize an answer and determine whether a query is satisfied in a sequence of frames. Intuitively, if the number of positive predictions across the models utilized in the query exceeds k_{crit} in a sequence of frames, then this frame sequence has a higher probability of satisfying the query. In addition, instead of determining such statistical properties statically, we propose a **D**ynamic method to adjust them as the video stream evolves. Our proposed method, called SVAQD, is able to detect sudden changes in the video stream properties and adjust its estimations accordingly while capable of ignoring gradual changes over time, thus dealing with concept drift in a natural manner [44].

For the offline case, in which queries are issued against video repositories we present a framework to **R**ank the relevant **V**ideo segments in the answers of the **A**ction **Q**uery. Our ranking framework, called RVAQ, can easily adopt any monotonic ranking scheme. We present a sample one incorporating various query specific signals such as the length of the answer video segments and the number of the detected objects and actions across relevant video parts. Our proposal consists of two steps: an ingestion phase and a query phase. During the ingestion phase which is executed only once when videos are added to the repository (i.e., a pre-processing step), several metadata are extracted in a query independent manner, which are later utilized during query processing. During the query phase, our algorithm extracts query specific metadata from each video (materialized in the ingestion phase) and utilizes a top-k query processing framework to produce the k most relevant results to the query. Given the specific nature of our problem, we demonstrate that a very different query processing methodology is required, compared to traditional top-k algorithms [12] for best performance.

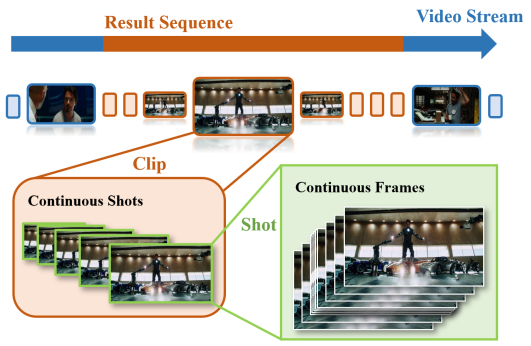


Figure 1: An example video stream. Green rectangles represent shots; rounded rectangles represent clips (red ones represent positive clips).

In this paper, we make the following contributions:

- We formalize problems in the context of processing queries on video streams containing objects and actions present in the videos. To the best of our knowledge our work is the first to present a declarative framework combining objects and actions on such data.
- For the case of processing such queries over streaming videos, we introduce algorithm SVAQD that dynamically adjusts its estimates as the stream evolves. Our approach makes use of *scan statistics* combining object and action joint occurrences on frames in a principled manner.
- For the case of processing queries over videos that are amenable to preprocessing, we present algorithm RVAQ. The algorithm consists of an ingestion phase during which videos are preprocessed, extracting and materializing metadata that is subsequently utilized in a querying phase. The algorithm is able to process queries involving objects and any action supported by object/action detection models and efficiently return top-k frame sequences (for a user specified k) that satisfy the query.
- Finally we conduct experiments utilizing real videos demonstrating the accuracy and performance benefits of our overall approach.

This paper is organized as follows. Section 2 presents background material and introduces terminology utilized in the remainder of the paper. Section 3 presents our solution and associated algorithms for the online case. In Section 4 we present algorithm RVAQ and detail our proposal for processing top-k queries involving objects and actions over preprocessed videos. Section 5 presents our experimental evaluation. Section 6 reviews related work, followed by section 7 which concludes the paper and points to interesting problems for future work.

2 Background

A video is a sequence of frames $V = \{v_1, \dots v_n\}$. The number of frames (length) of a video n can be fixed or unbounded. If n is unbounded we refer to V as a video stream. A video repository is a collection of videos each of fixed length. In a video V we provide the following definitions.

(1) Frame, v: this is the building block of a video. Object detection algorithms provide their predictions accepting a frame as an input. Each frame in V has an index (e.g., v_i) which indicates its position in V.
(2) Shot, s: a continuous (with respect to the frame index) collection of frames of set length (see Figure 1) . Action classification models [5] accept a shot as input and yield their predictions for the action the shot includes.
(3) Clip, c: continuous (with respect to the frame index) collection of shots of set length (see Figure 1). Clips contain several shots that may yield multiple action and object predictions.
(4) Sequence, z: continuous collection of clips. Sequences are the results of our query evaluation and can be of any length as determined by our algorithms. Query result sequences are represented as $P = \{(c_l, c_r)\}$, which is a set of the pairs of start and end clip identifiers for each result sequence returned by our algorithms.

Consider the example of Figure 1, depicting a movie with more than 100k frames. The video is divided into non-overlapping clips, each of which contains a number (e.g., fifty continuous) frames. Then each clip is divided into a number (e.g., five) shots with each shot containing a number of continuous frames. The length of shots in frames, is decided by action classification algorithms [5, 50] (typical values in the literature range from 10-30). The clip length is a parameter in our setting. Smaller clip lengths may fragment a long result sequence into multiple sequences; while a larger clip length may not. We will evaluate the effect of the length of the clips in our experiments.

Object Detection. A frame may contain multiple object types. An instance of each type may be detected one or multiple times in the same frame by an object detection model. Each detection of an instance of an object type at a frame is assigned a score by the model. Let O be the set of all the objects that can be recognized by the deployed object detection model. For each object type, $o_i \in O$, we denote ${}_{\max}S_{o_i}{}^{(v)}$ as the maximum score for all the instances of o_i on frame v given by the object detection model,

$$ {}_{\max}S_{o_i}{}^{(v)} = \max\{S^* \in O(o_i|v)\}, $$

where $O(o_i|v)$ represents the set of the scores of each detection of the instance of object type o_i at frame v. In accordance to prior work [22, 35, 37, 48], one imposes a threshold $\mathcal{T}_{\text{obj}}$, on the scores to filter out the predictions of all object instances that the detector assigns a score below the threshold, keeping only detections with sufficient score. Such a threshold can be configured per object type but to ease notation, without loss of generality, we fix a threshold across all object types an object detector supports. We denote $\mathbb{1}_{o_i}{}^{(v)}$ as the prediction indicator of the object type o_i on frame v provided by the object detection model,

$$ \mathbb{1}_{o_i}{}^{(v)} = \mathbb{1}[{}_{\max}S_{o_i}{}^{(v)} \geq \mathcal{T}_{\text{obj}}], $$

We say that the object type o_i has a positive prediction on frame v, iff $\mathbb{1}_{o_i}{}^{(v)}$=1.

Different instances of each object type can appear in the same frame. In our framework we enable an object tracking module [52] that can track the same object instance across different frames. Each object instance gets assigned a unique tracking identifier by the object tracking module the first time it is detected. The identifier remains the same as the object is detected across frames for as long as it is present. We denote $S^t_{o_i}{}^{(v)}$ as the score of the instance of object type o_i with tracking ID t on the frame v given by the object tracking model.

Action Classification. Action classification takes place on shots; utilizing an action classification model [5], each shot is predicted to have zero or more actions from a universe of action categories the model is trained on. We denote A the set of all the action categories that can be recognized by the deployed action classification model. The action classifier will provide scores for all action categories predicted on each shot. For each action type $a_j \in A$, we denote $S_{a_j}{}^{(s)}$ the score of a_j detected on shot s. For action classification, a threshold, $\mathcal{T}_{\text{act}}$, will also be set to filter out the action predictions with scores below the threshold [4, 5, 43]. We denote $\mathbb{1}_{a_j}{}^{(s)}$ as the prediction indicator of the action type a_j on shot s,

$$ \mathbb{1}_{a_j}{}^{(s)} = \mathbb{1}[S_{a_j}{}^{(s)} \geq \mathcal{T}_{\text{act}}]. $$

We say the action type, a_j, has a positive prediction on the shot s, iff $\mathbb{1}_{a_j}{}^{(s)}$=1.

Query. Generally, a query specification encompasses several predicates, which can be specific actions (e.g., robot dancing from Kinetics-700 [4]), presence of objects in frames (e.g., human, car), or relationships between objects in frames (e.g., human left of the car). In this paper we consider queries consisting of conjunctions of query predicates. A query q is:

$$ q : \{p_1; \ldots; p_I; a \in A\}, $$

where I is the number of query predicates p_i involving object types, a represents the query action and A is the set of all actions the action detection model is trained on. Predicates p_i may request the presence of a specific object type o_i on frames or evaluate the existence of a constraint (e.g., spatial relationship) among object types on frames [28]. In either case the evaluation of each p_i on a frame has a binary output (true/false). The techniques used to evaluate each p_i on a frame are orthogonal to our approach. For example, the existence of an object type can be conducted by any object detection model [22]. The evaluation of spatial relationships among objects can be conducted evaluating spatial predicates on the detected objects [28].

Without loss of generality and to simplify our notation, for the remainder of the paper we assume that predicates p_i are mainly concerned with the presence of specific, query specified, object types in frames. Thus a query specification becomes $q = \{o_1 \in O; \ldots; o_I \in O; a \in A\}$ where o_i represents the type of the i_{th} query object; a represents the type of the query action; O and A represent the set of all the objects and actions that can be recognized by the deployed object/action detection models. For example, for the query requesting a human jumping while a car is present in the scene, we have:

$$ q : \{o_1 = \text{human}; o_2 = \text{car}; a = \text{human jumping}\}. \tag{1} $$

An SQL-like query for the online (video streaming) case that we will support in this work would be of the form:

```
SELECT MERGE(clipID) AS Sequence, I1(humanDetections) AS
      humanIndicator, I2(carDetections) AS carIndicator, I3(
      jumpingDetections) AS jumpingIndicator,
FROM (PROCESS inputVideo PRODUCE clipID, humanDetections,
      carDetections USING ObjectDetector, jumpingDetections USING
      ActionClassifier)
WHERE humanIndicator=1 AND carIndicator=1 AND jumpingIndicator=1
```

The query employs an object detector and an action classifier to make the predictions of human, car and jumping, uses three binary indicators (I_i) to determine whether the objects and the action appear on each clip, and then MERGEs the continuous clips that contain all query objects and the action to form result sequences. Similarly, an SQL-like query that we will support over a video repository (offline case) is:

```
SELECT MERGE(clipID) AS Sequence, RANK(clipID, humanDetections,
      carDetections, jumpingDetections) AS sequenceScore
FROM (PROCESS inputVideo PRODUCE clipID, humanDetections,
      carDetections USING ObjectTracker, jumpingDetections USING
      ActionClassifier)
WHERE I1(humanDetections)=1 AND I2(carDetections)=1 AND I3(
      jumpingDetections)=1
ORDER BY sequenceScore LIMIT K
```

Similar to the online query execution, the offline query also first determines whether the objects and the action appear on each clip with an object tracker and an action classifier, and MERGEs the continuous clips that contain all query objects and the action to generate sequences. Then, it groups together the scores in a sequence with a user defined aggregation function, RANK, to provide an overall score for the sequence. The top K sequences will be returned ordered by their scores. For the offline case *inputVideo* in the `FROM` clause ca nrefer to one or more video suitably preprocessed as we detail in Section 4.

3 Online Case

In this section, we first propose an online model, SVAQ, for processing and accelerating queries with complex predicates on video streams. We then propose an improved online model, SVAQD, equipped with dynamic parameter adjustment yielding a more robust processing framework.

3.1 Algorithm SVAQ

For the online case addressed in our work, algorithm SVAQ, is responsible for processing a query $q : \{o_1; ...; o_I; a\}$. For each object type, o_i in q, we consider the positive object detection ($\mathbb{1}_{o_i}{}^{(v)}$=1) on frames as an event in a probabilistic sense. In order to cope with object detection model noise (inaccuracies introducing false positive and negatives) and also determine what constitutes a suitable number of object detections of a specific type in a sequence of frames, we will estimate (section 3.2) a statistically sound number of detections ($k_{\text{crit\_}o_i}$, for $1 \le i \le I$). This will enable us to compute all the clips considered to contain the object type o_i. The indicator of o_i on clip c is defined as,

$$\mathbb{1}_{o_i}{}^{(c)} = \mathbb{1}[\sum_{v \in V(c)} \mathbb{1}_{o_i}{}^{(v)} \ge k_{\text{crit\_}o_i}], \tag{2}$$

where $V(c)$ represents the set of frames in clip c.

Similarly, for the action type a specified in the query, we consider a positive action classification ($\mathbb{1}_a{}^{(s)}$=1) on shots as an event. We will also determine (section 3.2) a statistically sound number of action classifications ($k_{\text{crit\_}a}$) to deal both with classification inaccuracies and also infer what constitutes a suitable number of action classifications of a specific type a in a sequence of shots. That way we will obtain the clips that are considered to contain action a. The indicator of a on clip c then is,

$$\mathbb{1}_a{}^{(c)} = \mathbb{1}[\sum_{s \in S(c)} \mathbb{1}_a{}^{(s)} \ge k_{\text{crit\_}a}], \tag{3}$$

where $S(c)$ represents the set of shots in clip c.

For the query q, a clip c is considered to contain all the query specified object types and action (referred to as *positive clips*) if all have positive predictions on clip c, that is,

$$\mathbb{1}_q{}^{(c)} = \mathbb{1}_{o_1}{}^{(c)} \wedge \cdots \wedge \mathbb{1}_{o_I}{}^{(c)} \wedge \mathbb{1}_a{}^{(c)}. \tag{4}$$

Algorithm 1 Clip Indicator

```
Input: query $q$: $\{o_1; ...; o_I; a\}$; object detection model $\mathcal{O}$ and action classification
    model $\mathcal{A}$; thresholds $\mathcal{T}_{\text{obj}}$ and $\mathcal{T}_{\text{act}}$; clip $c$;
Output: indicator of clip $c$: $\mathbb{1}_q{}^{(c)}$;
 1: for each $i$ in $1, \ldots, I$ do
 2:     for each frame $v$ in the clip $c$ do
 3:         ${}_{\max}S_{o_i}{}^{(v)} \leftarrow \max\{S^* \in \mathcal{O}(o_i|v)\}$.
 4:         $\mathbb{1}_{o_i}{}^{(v)} \leftarrow \mathbb{1}[{}_{\max}S_{o_i}{}^{(v)} \ge \mathcal{T}_{\text{obj}}]$.
 5:     end for
 6:     $\mathbb{1}_{o_i}{}^{(c)} \leftarrow \mathbb{1}[\sum_{v \in V(c)} \mathbb{1}_{o_i}{}^{(v)} \ge k_{\text{crit\_}o_i}]$.
 7:     if $\mathbb{1}_{o_i}{}^{(c)} = 0$ then
 8:         $\mathbb{1}_q{}^{(c)} \leftarrow 0$.
 9:         continue outer for loop.
10:     end if
11: end for
12: for each shot $s$ in clip $c$ do
13:     $S_{a_j}{}^{(s)} \leftarrow \mathcal{A}(a|s)$ .
14:     $\mathbb{1}_a{}^{(s)} \leftarrow \mathbb{1}[S_{a_j}{}^{(s)} \ge \mathcal{T}_{\text{act}}]$.
15: end for
16: $\mathbb{1}_a{}^{(c)} \leftarrow \mathbb{1}[\sum_{s \in S(c)} \mathbb{1}_a{}^{(s)} \ge k_{\text{crit\_}a}]$.
17: $\mathbb{1}_q{}^{(c)} \leftarrow \mathbb{1}_{o_1}{}^{(c)} \wedge \cdots \wedge \mathbb{1}_{o_I}{}^{(c)} \wedge \mathbb{1}_a{}^{(c)}$.
18: return $\mathbb{1}_q{}^{(c)}$.
```

Algorithm 2 SVAQ

```
Input: video stream $\mathcal{X}$; $q$: $\{o_1; ...; o_I; a\}$; $k_{\text{crit\_o\_init}}$, $k_{\text{crit\_a\_init}}$;
Output: set of result sequences $P_q$;
 1: $k_{\text{crit\_}o_1}, k_{\text{crit\_}o_2}, \ldots, k_{\text{crit\_}o_I} \leftarrow k_{\text{crit\_o\_init}}$.
 2: $k_{\text{crit\_}a} \leftarrow k_{\text{crit\_a\_init}}$.
 3: $P_q \leftarrow \emptyset$.                          # initialize the result sequence set.
 4: while !$\mathcal{X}.end()$ do
 5:     $c \leftarrow \mathcal{X}.next()$.                # grab the next clip in video stream $\mathcal{X}$.
 6:     $\mathbb{1}_q{}^{(c)}$ ← get the indicator of clip $c$ through Algorithm 1 w.r.t. values
        $k_{\text{crit\_}o_1}, \ldots, k_{\text{crit\_}o_I}$ and $k_{\text{crit\_}a}$.
 7: end while
 8: $P_q \leftarrow \{(c_l, c_r) \mid \forall c \in [c_l, c_r] : \mathbb{1}_q{}^{(c)} = 1; \mathbb{1}_q{}^{(c_l - 1)} = 0; \mathbb{1}_q{}^{(c_r + 1)} = 0; \}$.
```

These positive clips are merged together to produce result sequences; let P_q denote the set of all result sequences for the query q,

$$P_q = \{(c_l, c_r) \mid \forall c \in [c_l, c_r] : \mathbb{1}_q{}^{(c)} = 1; \mathbb{1}_q{}^{(c_l - 1)} = 0; \mathbb{1}_q{}^{(c_r + 1)} = 0; \}, \tag{5}$$

where (c_l, c_r) represents a result sequence that starts from the clip c_l and ends at c_r.

Algorithm SVAQ is presented as Algorithm 2. The input consists of continuous sequence of frames $\mathcal{X}$, the query q, and the values $k_{\text{crit\_o\_init}}/k_{\text{crit\_a\_init}}$. The derivation of values $k_{\text{crit\_o\_init}}/k_{\text{crit\_a\_init}}$ will be discussed in section 3.2. For purposes of exposition in Algorithm 2 we initialize all values $k_{\text{crit\_}o_i}$ the same, but each may have its own initial values. The algorithm outputs a set of result sequences P_q. The algorithm determines clips from the video stream that satisfy the query (Lines 4-7) and then merges them if they are continuous into result sequences (Line 8).

When a new clip is considered, SVAQ conducts object detection and action classification on its frames and shots respectively and determines whether the clip satisfies query q. Algorithm 1 depicts the process of determining whether a clip c satisfies the query. For each object type specified in the query, the algorithm determines the indicator of this object type on clip c (Lines 2-6). Next, it determines the indicator of the query action type on clip c (Lines 12-16). Finally, it determines whether clip c satisfies the query (Line 17). As query predicates are evaluated in sequence, a new predicate is evaluated only when the former predicate is determined to be positive; otherwise this clip can be skipped (Lines 7-10).

3.2 Statistical Properties of Event Sequences

Detection of objects taking place in frames, is considered a statistical event. Every-time an object specified in the query is detected on a frame, we consider the presence of this object in the frame as an event associated with this object type occurring on the frame. Similarly when a shot is classified to contain the action specified by the query, we consider this an event associated in the action taking place at the shot. As a result, as the video stream progresses, we generate sequences of events associated with the objects and the actions included in the query.

Our discussion in Section 3.1 left unspecified a formal derivation of a statistically significant number of events (objects or action) in a sequence, which we will specify in this section. Scan statistics [21, 32] is a powerful method to detect unusually high rates of events in a sequence. In our setting we can deploy principles of scan statistics to detect a high concentration of positive predictions by the object detector (${\mathbb{1}_{o_i}}^{(v)} = 1$) and action classifier (${\mathbb{1}_{a_j}}^{(s)} = 1$).

We refer to the lowest granularity an event may occur (object detected or action predicted) as the *occurrence unit* (OU, which can be a frame or a shot in our case). Since OU are discrete, we model event occurrence by a Bernoulli distribution. Thus, occurrence of events can be modelled as N Bernoulli trials with a set background probability of success on a given trial p.

Let $n_{y,y+w}$ denote the number of successes in trials y, y+1, y+2, ..., y+w−1. After N OUs have been observed, define $S_w(N)$ as the maximum number of OUs to be found in any *scanning interval* (sequence of OUs) of length w, $S_w(N) = \max_{1\leq y\leq N-w+1} n_{y,y+w}$. When $S_w(N) \geq k$, we say that a quota of at least k successes within some w consecutive trials has occurred. Following [32] we approximate the distribution $P(S_w(N) \geq k|p,w,L)$, where $L = N/w$. Let $b_k = b(k;w,p) = \binom{w}{k}p^k(1-p)^{w-k}$ and $F_b(r;s,p) = \sum_{i=0}^{r} b(i;s,p)$ as

$$P(S_w(N) \geq k|p,w,L) \approx 1 - Q_2(Q_3/Q_2)^{L-2}.$$
$$Q_2 = F_b^2(k-1;w,p) - (k-1)b_kF_b(k-2;w,p) + wpb_kF_b(k-3;w-1,p).$$
$$Q_3 = F_b^3(k-1;w,p) - A_1 + A_2 + A_3 - A_4.$$

where

$$A_1 = 2b_kF_b(k-1;w,p)((k-1)F_b(k-2;w,p) - wpF_b(k-3;w-1,p));$$
$$A_2 = \frac{1}{2}b_k^2((k-1)(k-2)F_b(k-3;w,p) - 2(k-2)wpF_b(k-4;w-1,p) + w(w-1)p^2F_b(k-5;w-2,p));$$
$$A_3 = \sum_{r=1}^{k-1} b_{2k-r}F_b^2(r-1;w,p);$$
$$A_4 = \sum_{r=2}^{k-1} b_{2k-r}b_r((r-1)F_b(r-2;w,p) - wpF_b(r-3;w-l,p)).$$

Based on this, assuming that the background probability is $p = p_0$ (for a given p_0) we can compute the smallest k, *a critical value* for objects and actions for which the following holds:

$$P(S_w(N) \geq k_{\text{crit}}|p_0,w,L) \leq \alpha. \tag{6}$$

If the number of positive predictions in a scanning interval is greater than the critical value, then we say that at significance level α the event (e.g., object type, an action) is present at the scanning interval. For the case of object types which are part of the query, the OU is a frame and for the case of actions it is a shot. Using the suitable OU in each case (frame for object and shot for actions), we can compute critical values for objects $k_{\text{crit\_o\_init}}$ (one per object used in a query) and actions $k_{\text{crit\_a\_init}}$ using this methodology (as utilized in Algorithm 2)

As a result, we can now define indicator functions for objects and action to be present at a clip c as follows:

$$\sum_{v\in V(c)} {\mathbb{1}_{o_i}}^{(v)} \geq k_{\text{crit\_}o_i} \Rightarrow {\mathbb{1}_{o_i}}^{(c)} = 1,$$
$$\sum_{s\in S(c)} {\mathbb{1}_{a_j}}^{(s)} \geq k_{\text{crit\_}a} \Rightarrow {\mathbb{1}_{a_j}}^{(c)} = 1,$$

where $V(c)$ is the set of all frames in a clip and $S(c)$ is the set of all shots in a clip. A query will then be satisfied on a clip as per equation 4.

Such an approach however has clear limitations. Notably the background probability (either a single one across all object predicates and the action or one per object predicate separately and the action) has to be set apriori. This is not easy to do in practise as one has no context or guidance on how to set them. We will remove this constraint next.

3.3 Dynamic Parameter Updates (SVAQD)

We assumed so far, that the Bernoulli probability of positive predictions, p_0, is fixed across predicates of query q (objects and action) or equivalently that each object predicate and the action have their own (possibly different) fixed probability of positive predictions set apriori. In practice, however, the background probability p_0 can change suddenly. For instance, a surveillance camera at a crossroad can experience peak traffic at certain times of the day, that is, the probability of a vehicle being detected varies over time. Thus, deciding the value of p_0 for the entire video is not acceptable.

We aim to estimate the suitable value of p utilizing data from observations over a period of time. We present our derivation assuming the granularity of an event is an OU (as in section 3.2). The maximum likelihood estimates of p is $\frac{N^*}{N}$, where N^* is the number of events (positive predictions for an object type or action) and N is the total number of OUs. An intuitive method to estimate p is to define a view length $\mathcal{V}$. When updating the value of p, only the events in $\mathcal{V}$ OUs before and after the current OU are considered. However, this will introduce new parameters. Thus, we use a more feasible parameter update method to update p automatically, while ignoring gradual enough changes.

Let t represent an occurrence unit and $\mathcal{R}$ a region around it; such a region will contain occurrence units before and after t in sequence. We observe that the probability of an event, drawn from a Binomial distribution $p(t)$, will fall in a region $\mathcal{R}$ is: $P = \sum_{i\in\mathcal{R}} p(t_i)$ where t_i is the i-th occurrence unit in region $\mathcal{R}$. If we have N^* events drawn from Binomial distribution $p(t)$, the probability that k of these N^* events fall in the region $\mathcal{R}$ is $\mathbb{P}(k) = \binom{N^*}{k}P^k(1-P)^{N^*-k}$. It is known [1, 44] that the expectation and variance of $\frac{k}{N^*}$ is $\mathbb{E}[\frac{k}{N^*}] = P$, $Var[\frac{k}{N^*}] = \frac{P(1-P)}{N^*}$. Thus, when $N^* \rightarrow \infty$, $P \approx \frac{k}{N^*}$. If we assume $\mathcal{R}$ is so small that $p(t)$ does not vary appreciably within it, then $P = \sum_{i\in\mathcal{R}} p(t_i) = p(t)u$ where u is the volume enclosed by region $\mathcal{R}$. Thus, we obtain

$$\hat{p}(t) = \frac{k}{N^*u} = \frac{1}{N^*u}\sum_{n=1}^{N^*} K(\frac{t-t_n}{u}),$$

K is a kernel function; N^* is the number of events; t_n is the occurrence unit that the n_{th} event occurs.

By choosing a smooth exponential kernel, for example, $K(\frac{t-t_n}{u}) = \exp(-\frac{t-t_n}{u})$, the parameter update can be made very efficient even

Algorithm 3 SVAQD

```
Input: video stream $\mathcal{X}$; query $q$: $\{o_1; ...; o_I; a\}$; initialized background probabilities $p_{\text{obj}_0}, p_{\text{act}_0}$;
Output: set of result sequences $P_q$;
1: $p_{o_1}, ..., p_{o_I} \leftarrow p_{\text{obj}_0}$.
2: $p_a \leftarrow p_{\text{act}_0}$.
3: $k_{\text{crit\_}o_1}, \ldots, k_{\text{crit\_}o_I}, k_{\text{crit\_}a} \leftarrow$ calculate critical values through Scan Statistics (Equation 6) w.r.t. $p_{o_1}, ..., p_{o_I}, p_a$.
4: $P_q \leftarrow \emptyset$.
5: while $!\mathcal{X}.end()$ do
6:     $c \leftarrow \mathcal{X}.next()$.
7:     $\mathbb{1}_q{}^{(c)} \leftarrow$ get indicator of clip $c$ through Algorithm 1 w.r.t. $k_{\text{crit\_}o_1}, \ldots, k_{\text{crit\_}o_I}$ and $k_{\text{crit\_a}}$.
8:     if $\mathbb{1}_q{}^{(c)} = 1$ then
9:         $p_{o_1}, ..., p_{o_I}, p_a \leftarrow$ update background $p$s (Equation 7).
10:        $k_{\text{crit\_}o_1}, \ldots, k_{\text{crit\_}o_I}, k_{\text{crit\_}a} \leftarrow$ calculate the new critical values through Scan Statistics w.r.t. $p_{o_1}, ..., p_{o_I}, p_a$.
11:    end if
12: end while
13: $P_q \leftarrow \{(c_l, c_r) \mid \forall c \in [c_l, c_r] : \mathbb{1}_q{}^{(c)} = 1; \mathbb{1}_q{}^{(c_l-1)} = 0; \mathbb{1}_q{}^{(c_r+1)} = 0; \}$.
```

for a large N^*. Given the estimate $\hat{p}(t)$, we can update it after Δt OUs:

$$\hat{p}(t+\Delta t) = \frac{1}{N^* u} \sum_{n=1}^{N^*} \exp\left(-\frac{t+\Delta t - t_n}{u}\right) = \exp\left(-\frac{\Delta t}{u}\right)\hat{p}(t).$$

If an event occurs when updating the estimate $\hat{p}(t)$, a bias will be generated. Edge correction [10] can help to remove this bias,

$$\begin{aligned}\hat{p}(t) &= \frac{1}{N^* u} \sum_{n=1}^{N^*} \exp\left(-\frac{t-t_n}{u}\right) \Big/ \sum_{j=1}^{N} \exp\left(-\frac{t-t_j}{u}\right) \\ &= \frac{1}{N^* u} \sum_{n=1}^{N^*} \exp\left(-\frac{t-t_n}{u}\right) \frac{1-\exp(-\frac{1}{u})}{1-\exp(-\frac{t}{u})},\end{aligned}$$

where N is the total number of OUs observed. This estimator is unbiased in the case when the background probability is constant. The overall update equations with edge correction are

$$\hat{p}(t+\Delta t) = \hat{p}(t) \frac{1-\exp(-\frac{t}{u})}{\exp(\frac{\Delta t}{u}) - \exp(-\frac{t}{u})} + \frac{1-\exp(-\frac{1}{u})}{u(1-\exp(-\frac{t+\Delta t}{u}))}. \quad (7)$$

This equation refines the background probability by smoothing events over time. Thus for each object type specified in the query and the action, we can now adjust the background probability dynamically over time.

The resulting algorithm, named SVAQD, employs the method of dynamic parameter adjustment for each object type present in query predicates and for the action predicate (Equation 7). For the case of object types, the occurrence unit is fixed to a frame, and for the action, it is a shot. The process is shown in Algorithm 3. Different from SVAQ, SVAQD is initialized with values p_{obj_0} (can be the same for all objects present in the query or different) and p_{act_0} as input. The update of p can be performed every time a new event occurs (Lines 8-11), or after processing a fixed number of clips. This algorithm will eliminate the influence of p_{obj_0} and p_{act_0} naturally, adjusting them as the statistical properties of the video change, increasing the robustness of the online model.

4 Offline Case

We now turn our attention to answering queries over one or more videos in an offline manner. In this case, each video is amenable to pre-processing to extract certain metadata which are fully available during query processing. During an *ingestion phase* we process each video utilizing object detection and action detection models. Since the queries are not known in advance, we process each frame extracting meta data for all possible object types recognized by the deployed object detection model and for all possible actions, supported by the action classification model utilized. At query time, when desired objects and actions are specified, the applicable meta data are utilized for query processing.

We propose algorithm RVAQ to process queries utilizing metadata extracted during the ingestion phase. In order to effectively limit the number of results returned, RVAQ utilizes scoring functions to rank the results, returning only the K highest scoring ones, for a user specified K provided at query time. We first detail a general class of scoring functions supported by RVAQ as well as detail a sample one which we utilize in our experiments for purposes of exposition (Section 4.1). We then present the ingestion phase in Section 4.2 followed by algorithm RVAQ in Sections 4.3 and 4.4.

4.1 Scoring Functions

The scoring functions utilized in our work, map a result sequence z to a real number. Let d be the maximum number of scores that can be used as an input to our scoring function [1]. In our work we support any scoring function $f: \mathbb{R}^d \to \mathbb{R}$ to rank a result sequence z that satisfies the following *monotonicity* properties:

- the score of a sequence, z, is monotonic to each prediction score over z given by the deployed object/action detection/-classification models.

$$\begin{aligned}&\partial f / \partial S_{o_i}^{t}{}^{(v)} \geq 0 \qquad \forall o_i \in q, \forall t \in T_{o_i}(z), \forall v \in V(z);\\ &\partial f / \partial S_a{}^{(s)} \geq 0 \qquad \forall s \in S(z),\end{aligned}$$

  where $V(z)$ is the set of all frames in sequence z; $S(z)$ is the set of all shots in sequence z; $T_{o_i}(z)$ is the set of all tracking IDs of object type o_i in sequence z. $S_{o_i}^{t}{}^{(v)}$ is available by the deployed object detection algorithm for object type o_i on frame v with tracking identifier t. $S_a{}^{(s)}$ is available by the action classification algorithm.
- The score of a sub-sequence of a sequence z is always lower than that of z. Let z' represent a sub-sequence of z,

$$f(z) \geq f(z') \quad \forall z' \subseteq z.$$

- If a sequence is divided into multiple parts we can obtain the score of the sequence by manipulating the scores of its parts. Let $\odot$ represent an aggregation operator on scores,

$$\begin{aligned}&\text{if } z_1 \cup z_2 \cup \ldots \cup z_m = z,\\ &z_i \subseteq z, z_j \subseteq z, z_i \cap z_j = \phi, \quad \forall i{\neq}j, 1{\leq}i{\leq}m, 1{\leq}j{\leq}m\\ &f(z) = f(z_1) \odot f(z_2) \odot \ldots \odot f(z_m).\end{aligned}$$

To realise things concrete we outline below a sample scoring function which we use in our experiments, but any function that adheres to the properties above can be easily adopted. Each sequence is composed of several continuous positive clips (Section 3.1). Let $S_{o_i}{}^{(c)}$ denote the score of object type o_i on a clip c:

$$S_{o_i}{}^{(c)} = \sum_{v \in V(c)} \sum_{t \in T_{o_i}(c)} S_{o_i}^{t}{}^{(v)}, \quad (8)$$

where $V(c)$ is the set of all frames in clip c and $T_{o_i}(c)$ represents the set of all tracking IDs of the object type o_i that appear in clip c. Similarly, let $S_{a_j}{}^{(c)}$ denote the score of clip c for action a_j,

$$S_{a_j}{}^{(c)} = \sum_{s \in S(c)} S_{a_j}{}^{(s)}. \quad (9)$$

where $S(c)$ is the set of shots in c.

[1]The features used as the input to our scoring functions include all the prediction scores over the sequence given by the deployed object/action detection/classification models.

For a query $q : \{o_1; ...; o_I; a\}$, we define the score for the clip c under the query q as

$$S_q{}^{(c)} = \sum_{i=1}^{I} S_{o_i}{}^{(c)} + S_a{}^{(c)}. \quad (10)$$

Let $S_q{}^{(z)}$ represent the score of sequence, z, under query q. It is the sum of the scores of all clips in this sequence,

$$S_q{}^{(z)} = \sum_{c \in C(z)} S_q{}^{(c)}, \quad (11)$$

where $C(z)$ represents the set of all the clips in the sequence z.

4.2 Ingestion Phase

4.2.1 Clip Score Tables During the ingestion phase, meta data are extracted from each video in a query independent manner. We describe the process for a single video to simplify notation. Multiple videos are handled in the same manner by associating a *video identifier* to each clip identifier (cid) below.

We process the video one clip c at a time and calculate the scores for each object type and each action type on the clip, materialising them in the following tables,

$$table_{o_i} : \{cid, Score\} \quad \forall o_i \in O; \quad table_{a_j} : \{cid, Score\} \quad \forall a_j \in A.$$

For each object type detected by the object detection model, we extract the score of each instance of this object type on each frame, calculate $S_{o_i}{}^{(c)}$ by Equation 8, and store it in $table_{o_i}$ as $(c, S_{o_i}{}^{(c)})$. Similarly, For each action type that is recognized by the action classification model, we obtain the score of this action type on each shot, calculate $S_{a_j}{}^{(c)}$ by Equation 9, and store it in $table_{a_j}$ as $(c, S_{a_j}{}^{(c)})$. The tuples in all these tables are ordered by *Score*. Table 1 presents examples of tables generated at the ingestion phase. Although we present the tables for a single video for brevity, notice that it is very easy to add more videos or delete videos in this setting, by manipulating the information in these tables. We just associate a video identifier for each *cid* in the tables.

Table 1: Examples of clip score tables materialized at ingestion phase.

$table_{\text{human}}$

CID	Score
3	0.94
2	0.92
1	0.91
...	...

$table_{\text{car}}$

CID	Score
2	0.95
4	0.92
1	0.92
...	...

$table_{\text{jumping}}$

CID	Score
2	0.72
1	0.70
5	0.67
...	...

*4.2.2 Individual Sequences*Utilizing algorithm SVAQD (Section 3.3), for each object type and action type, we determine the positive clips utilizing Equations 2 and 3 respectively. As a result for each object and for each action type, the entire video is divided into sequences by merging the corresponding consecutive positive clips. Let P_{o_i} denote the individual sequences for object type o_i, P_{a_j} denote the individual sequences for action type a_j. Let $C(V)$ be the total number of clips in video V; then for $1 \le c_i \le C(V)$ we have:

$$P_{o_i} = \{(c_l, c_r) \mid \forall c \in [c_l, c_r] : \mathbb{1}_{o_i}{}^{(c)} = 1; \mathbb{1}_{o_i}{}^{(c_l-1)} = \mathbb{1}_{o_i}{}^{(c_r+1)} = 0\};$$
$$P_{a_j} = \{(c_l, c_r) \mid \forall c \in [c_l, c_r] : \mathbb{1}_{a_j}{}^{(c)} = 1; \mathbb{1}_{a_j}{}^{(c_l-1)} = \mathbb{1}_{a_j}{}^{(c_r+1)} = 0\}.$$

For each object and action type, we first traverse all the clips in the target video, compute the indicators on each clip (Equation 2 and 3). Then, utilizing the equation above, we traverse the indicators on all clips to extract the individual sequences for each object type and each action type. The resulting sequences are stored as $\{(c_l, c_r)\}$, a set of the pairs of start and end clip identifiers. That way P_{o_i} and P_{a_j} are materialized during ingestion for each object type i and action type j in the deployed object detection and action classification models. This is easily accomplished with a single pass over all the clips of a video.

A clip c satisfies query q, if all the query object types and the action have positive predictions on this clip (Equation 4). Let $\otimes$ denote the *intersection* of two individual sequences defined in this paper. For two individual sequences P_1 and P_1, $P_1 \otimes P_2$ represents new sequences that contain the clips both in P_1 and P_2,

$$P_1 \otimes P_2 = \{(c_l, c_r) \mid \forall c \in [c_l, c_r] : c \in C(P_1) \cap C(P_2);$$
$$c_l - 1 \notin C(P_1) \cap C(P_2); c_r + 1 \notin C(P_1) \cap C(P_2)\}$$

where $C(P_1)$ and $C(P_2)$ represent the set of all the clip identifiers that appear in the sequences of P_1 and P_2 respectively. Thus, one can obtain sequences that satisfy query q, as:

$$P_q = P_a \otimes P_{o_1} \otimes P_{o_2} \otimes \ldots \otimes P_{o_I} \quad (12)$$

For any query q, P_q can be computed very efficiently; since all the P_{o_i} and P_a involved in q are set of clip identifier ranges, each range can be viewed as an interval. All intervals can be sorted by their start point and P_q can be computed in a single pass by an interval sweep [45].

Consider the following sequences generated at ingestion for query (1):

$$P_{\text{human}} : \{(1, 20), (150, 300), \ldots\}; \quad P_{\text{car}} : \{(1, 30), (200, 250), \ldots\};$$
$$P_{\text{jumping}} : \{(1, 25), (120, 240), \ldots\}. \quad (13)$$

We identify the result sequences containing object types *human*, *car* and action *jumping* by

$$P_q = P_{\text{jumping}} \otimes P_{\text{human}} \otimes P_{\text{car}} = \{(1, 20), (200, 240), \ldots\}.$$

A straightforward way to identify the K sequences in P_q with highest score is to retrieve each clip in the sequences of P_q, compute each sequence score (Equation 11) and report the K sequences with the highest score. This however will involve a large number of random accesses to the underlying clip score tables $table_{o_i}$, $table_{a_j}$. We will propose next a more efficient algorithm to identify the result sequences that effectively prunes early unnecessary sequences effectively limiting the number of clips that need to be accessed.

4.3 The RVAQ Algorithm

The basic idea of RVAQ is to estimate bounds (upper and lower) on the scores of each sequence in P_q and refine them progressively in a way that the top K sequences with the highest scores can be computed much faster without the need to inquire about the scores of all clips across all sequences in P_q.

The efficient access of the clips in P_q is conducted with the help of an iterator TBClip (Section 5) that progressively delivers clips of the sequences in P_q along with their scores and only those; clips not belonging to sequences in P_q are excluded. In particular, TBClip returns two clips each time (with scores calculated by Equation 10) clip c_{top} which is the clip in P_q with the largest score among those clips not already processed by the iterator and clip c_{btm} which is the clip in P_q with the lowest score among those not already processed by the iterator. Each time we obtain new values for c_{top}, c_{btm}, the upper and lower bounds of the scores for all sequences in P_q are re-estimated; the process continues until a stopping condition is encountered and the K sequences with the highest scores are identified.

Let $(c_l{}^{(i)}, c_r{}^{(i)}) \in P_q$ be a sequence with $1 \le i \le |P_q|$. For a sequence $(c_l{}^{(i)}, c_r{}^{(i)})$ we will denote $B_{\text{up}}^{c_l{}^{(i)}}$, $B_{\text{lo}}^{c_l{}^{(i)}}$ the estimates for its upper and lower bound scores.

$$B_{\text{lo}}^{c_l^{(i)}} \le S_q{}^{(i)} \le B_{\text{up}}^{c_l^{(i)}}, \quad 1 \le i \le |P_q|.$$

Given c_{top} and c_{btm}, we refer to the clips in a sequence $(c_l^{(i)}, c_r^{(i)}) \in P_q$ that have a larger/lower score than c_{top} and c_{btm} respectively as top/bottom *processed* clips in this sequence; we refer to all other clips in this sequence as *not processed*. Let $L_{\text{up}}^{c_l^{(i)}}/L_{\text{lo}}^{c_l^{(i)}}$ represent the length of a sequence $(c_l{}^{(i)}, c_r{}^{(i)}) \in P_q$ minus the number of *processed* top/bottom clips belonging to this sequence,

$$L_{\text{up}}^{c_l^{(i)}} = c_r^{(i)} - c_l^{(i)} - \left|\{\text{processed top clip } c \in [c_l^{(i)}, c_r^{(i)}]\}\right|, 1 \le i \le |P_q|;$$
$$L_{\text{lo}}^{c_l^{(i)}} = c_r^{(i)} - c_l^{(i)} - \left|\{\text{processed bottom clip } c \in [c_l^{(i)}, c_r^{(i)}]\}\right|, 1 \le i \le |P_q|.$$

where $S_{\text{up}}^{c_l^{(i)}}/S_{\text{lo}}^{c_l^{(i)}}$ represent the overall scores of top/bottom clips that have been processed in sequence $(c_l{}^{(i)}, c_r{}^{(i)})$,

$$S_{\text{up}}^{c_l^{(i)}} = \sum\nolimits_{\text{processed top clip } c \in [c_l{}^{(i)}, c_r{}^{(i)}]} S_q{}^{(c)}, \qquad 1 \le i \le |P_q|,$$
$$S_{\text{lo}}^{c_l^{(i)}} = \sum\nolimits_{\text{processed bottom clip } c \in [c_l{}^{(i)}, c_r{}^{(i)}]} S_q{}^{(c)}, \quad 1 \le i \le |P_q|.$$

After obtaining a new, c_{top} from the iterator, the upper bounds for all the sequences in P_q will be re-estimated,

$$B_{\text{up}}^{c_l^{(i)}} = S_q{}^{(c_{\text{top}})} \times L_{\text{up}}^{c_l^{(i)}} + S_{\text{up}}^{c_l^{(i)}}, \quad 1 \le i \le |P_q|. \tag{14}$$

where $S_q{}^{(c_{\text{top}})}$ represents the score of clip c_{top}; this is also the highest score of a clip among the not processed top clips of all result sequences. If $c_{\text{top}} \in [c_l{}^{(j)}, c_r{}^{(j)}]$, for $1 \le j \le |P_q|$, $L_{\text{up}}^{c_l^{(j)}}$ and $S_{\text{up}}^{c_l^{(j)}}$ will also be updated for the sequence $(c_l{}^{(j)}, c_r{}^{(j)})$.

$$L_{\text{up}}^{c_l^{(j)}} = L_{\text{up}}^{c_l^{(j)}} - 1, \quad S_{\text{up}}^{c_l^{(j)}} = S_{\text{up}}^{c_l^{(j)}} + S_q{}^{(c_{\text{top}})}.$$

Similarly, after obtaining a new, c_{btm} from the iterator, the estimation of the lower bound for each sequence $(c_l{}^{(i)}, c_r{}^{(i)})$ will also be refined:

$$B_{\text{lo}}^{c_l^{(i)}} = S_q{}^{(c_{\text{btm}})} \times L_{\text{lo}}^{c_l^{(i)}} + S_{\text{lo}}^{c_l^{(i)}}, \quad 1 \le i \le |P_q|. \tag{15}$$

We also update $L_{\text{lo}}^{c_l^{(j)}}$ and $S_{\text{lo}}^{c_l^{(j)}}$ for the sequence $(c_l{}^{(j)}, c_r{}^{(j)})$ if $c_{\text{btm}} \in [c_l{}^{(j)}, c_r{}^{(j)}]$:

$$L_{\text{lo}}^{c_l^{(j)}} = L_{\text{lo}}^{c_l^{(j)}} - 1, \quad S_{\text{lo}}^{c_l^{(j)}} = S_{\text{lo}}^{c_l^{(j)}} + S_q{}^{(c_{\text{btm}})}.$$

For each successive invocation of the iterator TBClip, the upper and lower bounds of all sequences are refined and converge to the exact values unless this process halts earlier triggered by a stopping condition.

The algorithm utilizes two priority queues to efficiently determine whether a stopping condition is satisfied. Let PQ_{lo}^K be a priority queue containing the K result sequences in P_q with the highest lower bounds, organized in non decreasing order of the bound estimates.

$$PQ_{\text{lo}}^K = \left\{(c_l{}^{(i)}, c_r{}^{(i)}) \text{ with } K \text{ highest } B_{\text{lo}}^{c_l^{(i)}} \mid (c_l{}^{(i)}, c_r{}^{(i)}) \in P_q\right\}.$$

Each time a new lower bound estimate for a sequence (Equation 15) is available, we update the priority queue PQ_{lo}^K reflecting the new score. Similarly, let $PQ_{\text{up}}^{\neg K}$ be a priority queue for all result sequences not in PQ_{lo}^K ranked in non increasing order of their upper bounds,

$$PQ_{\text{up}}^{\neg K} = \left\{(c_l{}^{(i)}, c_r{}^{(i)}) \text{ ranked by } B_{\text{up}}^{c_l^{(i)}} \mid (c_l{}^{(i)}, c_r{}^{(i)}) \in (P_q \setminus PQ_{\text{lo}}^K)\right\}.$$

Each time a new upper bound estimate is available (Equation 14), we update $PQ_{\text{up}}^{\neg K}$. If a sequence z is inserted in PQ_{lo}^K removing sequence z' from it, z is removed from $PQ_{\text{up}}^{\neg K}$ and z' is inserted instead. Let B_{lo}^K be the minimum of the lower bounds among the sequences in PQ_{lo}^K,

$$B_{\text{lo}}^K = \min\{B_{\text{lo}}^{c_l^{(i)}} \mid (c_l{}^{(i)}, c_r{}^{(i)}) \in PQ_{\text{lo}}^K\}.$$

Let $B_{\text{up}}^{\neg K}$ be the maximum of the upper bounds of all the result sequences not currently in PQ_{lo}^K,

$$B_{\text{up}}^{\neg K} = \max\{B_{\text{up}}^{c_l^{(i)}} \mid (c_l{}^{(i)}, c_r{}^{(i)}) \in PQ_{\text{up}}^{\neg K}\}.$$

Formally, the stopping condition becomes

$$B_{\text{lo}}^K \ge B_{\text{up}}^{\neg K}. \tag{16}$$

Algorithm RVAQ is presented as Algorithm 4. It first calculates P_q (Line 1) and initializes the statistics and the priority queues (Lines 3-10). It then updates the estimation of the upper/lower bound for each sequence and the upper/lower bound priority queues (Lines 12-30) after each invocation of the iterator. Subsequently, the minimum of the lower bounds among current top-K sequences and the maximum of the upper bounds of other sequences can be extracted (Lines 31-32) and we can check the stopping condition by Equation 16. The algorithm will repeat the invocation of the iterator TBClip until the stopping condition is triggered (Line 33). The result top-K sequences, stored in PQ_{lo}^K, will then be returned.

4.3.1 Skipped Clips During the execution of algorithm RVAQ, a lot of information is gained regarding which of the not processed clips are still relevant in estimating the upper/lower bounds and which should be *skipped* by TBClip. The set C_{skip} dynamically adjusts as the algorithm executes containing clips that the iterator TBClip can safely skip as they cannot contribute or alter the final result. The set contains two types of clips:

(1) The clips that are in the target video but not in P_q. At the beginning of Algorithm RVAQ (Line 2), C_{skip} will be initialized as $C(\mathcal{X}) \setminus C(P_q)$, where $C(\mathcal{X})$ represents all the clips in the target video; $C(P_q)$ represents the clips in the sequences of P_q.

(2) As algorithm RVAQ progresses some sequences are placed in the top-K result conclusively and some are conclusively excluded from the top-K result. More specifically, for each sequence $(c_l{}^{(i)}, c_r{}^{(i)}) \in P_q$, if its upper bound is smaller than B_{lo}^K, it will never be a top-K sequence; all the clips in this sequence are added into C_{skip} (Lines 16-18):

$$B_{\text{up}}^{c_l^{(i)}} < B_{\text{lo}}^K \Rightarrow C_{\text{skip}} = C_{\text{skip}} \cup \text{range}(c_l{}^{(i)}, c_r{}^{(i)}),$$

where $\text{range}(c_l{}^{(i)}, c_r{}^{(i)})$ is the set of all the clips in this sequence. Similarly, if the lower bound of a sequence is larger than $B_{\text{up}}^{\neg K}$, it is one of the top-K sequences; if the final exact scores of top-K sequences are not required, all the clips in this sequence will also be added into C_{skip} (Line 25-27),

$$B_{\text{up}}^{c_l^{(i)}} < B_{\text{lo}}^K \Rightarrow C_{\text{skip}} = C_{\text{skip}} \cup \text{range}(c_l{}^{(i)}, c_r{}^{(i)}).$$

C_{skip} is used by the iterator TBClip (detailed in Section 4.4) to return only clips relevant for further processing.

Algorithm 4 RVAQ

```
Input: query $q$: $\{o_1; ...; o_I; a\}$; $\forall i \in [1, I]$: $P_{o_i}$; $P_a$; $K$;
Output: top-K sequences: $P_{\text{fix}}^{\text{top}}$;
 1: $P_q \leftarrow P_a \otimes P_{o_1} \otimes P_{o_2} \otimes \ldots \otimes P_{o_I}$.
 2: $C_{\text{skip}} \leftarrow C(\mathcal{X}) \setminus C(P_q)$.
 3: $B_{\text{lo}}^{K}, B_{\text{up}}^{\neg K} \leftarrow -1, \infty$.
 4: for each $(c_l{}^{(i)}, c_r{}^{(i)})$ in $P_q$ do
 5:     $B_{\text{up}}^{c_l{}^{(i)}}, B_{\text{lo}}^{c_l{}^{(i)}} \leftarrow \infty, -1$.
 6:     $L_{\text{up}}^{c_l{}^{(i)}}, L_{\text{lo}}^{c_l{}^{(i)}} \leftarrow c_r{}^{(i)} - c_l{}^{(i)}, c_r{}^{(i)} - c_l{}^{(i)}$.
 7:     $S_{\text{up}}^{c_l{}^{(i)}}, S_{\text{lo}}^{c_l{}^{(i)}} \leftarrow 0, 0$.
 8: end for
 9: $PQ_{\text{lo}}^{K} \leftarrow \left\{(c_l{}^{(i)}, c_r{}^{(i)}) \text{ with } K \text{ highest } B_{\text{lo}}^{c_l{}^{(i)}} \mid (c_l{}^{(i)}, c_r{}^{(i)}) \in P_q\right\}$.
10: $PQ_{\text{up}}^{\neg K} \leftarrow \left\{(c_l^{(i)}, c_r^{(i)}) \text{ ranked by } B_{\text{up}}^{c_l{}^{(i)}} \mid (c_l^{(i)}, c_r^{(i)}) \in (P_q \setminus PQ_{\text{lo}}^{K})\right\}$.
11: repeat
12:     $c_{\text{top}}, S_q{}^{(c_{\text{top}})}, c_{\text{btm}}, S_q{}^{(c_{\text{btm}})} \leftarrow$ get the next top/bottom clip through iterator
        TBClip (Algorithm 5) w.r.t. $C_{\text{skip}}$.
13:     for each $(c_l{}^{(i)}, c_r{}^{(i)})$ in $P_q$ do
14:         $B_{\text{up}}^{c_l{}^{(i)}} \leftarrow S_q{}^{(c_{\text{top}})} \times L_{\text{up}}^{c_l{}^{(i)}} + S_{\text{up}}^{c_l{}^{(i)}}$.          # update upper bounds.
15:         update $PQ_{\text{lo}}^{K}$ and $PQ_{\text{up}}^{\neg K}$ w.r.t. $B_{\text{up}}^{c_l{}^{(i)}}$.
16:         if $B_{\text{up}}^{c_l{}^{(i)}} < B_{\text{lo}}^{K}$ then
17:             $C_{\text{skip}} \leftarrow C_{\text{skip}} \cup \text{range}(c_l{}^{(i)}, c_r{}^{(i)})$.
18:         end if
19:     end for
20:     $c_l{}^{(j)} \leftarrow c_l{}^{(i)}|_{(c_l{}^{(i)}, c_r{}^{(i)}) \in P_q:\ c_l{}^{(i)} \le c_{\text{top}} \le c_r{}^{(i)}}$.
21:     $S_{\text{up}}^{c_l{}^{(j)}}, L_{\text{up}}^{c_l{}^{(j)}} \leftarrow S_{\text{up}}^{c_l{}^{(j)}} + S_q{}^{(c_{\text{top}})}, L_{\text{up}}^{c_l{}^{(j)}} - 1$ .
22:     for each $(c_l{}^{(i)}, c_r{}^{(i)})$ in $P_q$ do
23:         $B_{\text{lo}}^{c_l{}^{(i)}} \leftarrow S_q{}^{(c_{\text{btm}})} \times L_{\text{lo}}^{c_l{}^{(i)}} + S_{\text{lo}}^{c_l{}^{(i)}}$.          # update lower bounds.
24:         update $PQ_{\text{lo}}^{K}$ and $PQ_{\text{up}}^{\neg K}$ w.r.t. $B_{\text{lo}}^{c_l{}^{(i)}}$.
25:         if $B_{\text{lo}}^{c_l{}^{(i)}} > B_{\text{up}}^{\neg K}$ then
26:             $C_{\text{skip}} \leftarrow C_{\text{skip}} \cup \text{range}(c_l{}^{(i)}, c_r{}^{(i)})$.
27:         end if
28:     end for
29:     $c_l{}^{(j)} \leftarrow c_l{}^{(i)}|_{(c_l{}^{(i)}, c_r{}^{(i)}) \in P_q:\ c_l{}^{(i)} \le c_{\text{btm}} \le c_r{}^{(i)}}$.
30:     $S_{\text{lo}}^{c_l{}^{(j)}}, L_{\text{lo}}^{c_l{}^{(j)}} \leftarrow S_{\text{lo}}^{c_l{}^{(j)}} + S_q{}^{(c_{\text{btm}})}, L_{\text{lo}}^{c_l{}^{(j)}} - 1$ .
31:     $B_{\text{lo}}^{K} \leftarrow \min\{B_{\text{lo}}^{c_l{}^{(i)}} \mid (c_l{}^{(i)}, c_r{}^{(i)}) \in PQ_{\text{lo}}^{K}\}$.
32:     $B_{\text{up}}^{\neg K} \leftarrow \max\{B_{\text{up}}^{c_l{}^{(i)}} \mid (c_l{}^{(i)}, c_r{}^{(i)}) \in PQ_{\text{up}}^{\neg K}\}$.
33: until $B_{\text{lo}}^{K} \ge B_{\text{up}}^{\neg K}$.
34: return $PQ_{\text{lo}}^{K}$.
```

4.4 TBClip

We now discuss TBClip, an iterator utilized by Algorithm 5 to return incrementally the highest and lowest ranking clips that satisfy query q. The iterator deploys a variant of popular top-k query algorithm processing algorithm (e.g., [12]) with some important differences as detailed below.

After some initialization the first time it is invoked, the iterator proceeds in two steps. First, for each $table_{o_i}$, $table_{a_j}$ involved in q, conducts sorted access in parallel starting from the row in each table, accessed last in the previous invocation, until at least one *new* clip is identified in all the tables (Lines 8-14). Second, performs random accesses to obtain the scores of all retrieved clips (Lines 17-25) to fully compute their scores. At the same time the iterator conducts the same process, starting from the end of all the tables involved in query q to compute the clip with the current lowest score. Similarly in each invocation of the iterator, the process to compute the clip with the lowest score, continues from the row in each table last accessed during sequential access in the previous invocation. The clip with the highest score along with the clip with the lowest score identified, are returned by the iterator in this invocation.

Algorithm 5 iterator: TBClip

```
Input: query $q$: $\{o_1, ..., o_I, a\}$; $\forall i \in [1, I]$: $table_{o_i}$; $table_a$; $C_{\text{skip}}$;
Output: next top and bottom clip identifiers and their scores: $c_{\text{top}}$, $S_{\text{top}}$, $c_{\text{btm}}$, $S_{\text{btm}}$.
 1: if this iterator is initialized then
 2:     $C_{\text{top}}, C_{\text{btm}} \leftarrow \emptyset, \emptyset$.                    # processed top/bottom clips.
 3:     $C_{o_1}{}^{\text{top}}, ..., C_{o_I}{}^{\text{top}}, C_a{}^{\text{top}} \leftarrow \emptyset, ..., \emptyset, \emptyset$.
 4:     $C_{o_1}{}^{\text{btm}}, ..., C_{o_I}{}^{\text{btm}}, C_a{}^{\text{btm}} \leftarrow \emptyset, ..., \emptyset, \emptyset$.
 5:     $stamp^{\text{top}}, stamp^{\text{btm}} \leftarrow 1, 1$.
 6: end if
 7: {Step 1} (8-14) for each table, do sorted access in parallel from the $(stamp^{\text{top}})_{\text{th}}$
    row until a new clip is found in all the tables.
 8: repeat
 9:     $c_{o_1}{}^{*}, ..., c_{o_I}{}^{*}, c_a{}^{*} \leftarrow$ retrieve all the CIDs that appear on the $stamp^{\text{top}}{}_{\text{th}}$ row
        of $table_{o_1}, ..., table_{o_I}, table_a$ respectively.
10:     $C_{o_1}{}^{\text{top}}, ..., C_{o_I}{}^{\text{top}} \leftarrow C_{o_1}{}^{\text{top}} \cup \{c_{o_1}{}^{*}\}, ..., C_{o_I}{}^{\text{top}} \cup \{c_{o_I}{}^{*}\}$.
11:     $C_a{}^{\text{top}} \leftarrow C_a{}^{\text{top}} \cup \{c_a{}^{*}\}$.
12:     $C_{\cap}{}^{\text{top}} \leftarrow C_a{}^{\text{top}} \cap \bigcap_{i=1}^{I} C_{o_i}{}^{\text{top}} - C_{\text{top}} - C_{\text{skip}}$.
13:     $stamp^{\text{top}} \leftarrow stamp^{\text{top}} + 1$.
14: until $|C_{\cap}{}^{\text{top}}| \ge 1$.
15: {Step 2} (17-25) perform random accesses to obtain the scores of all the seen clips
    and return the clip with the highest score.
16: $S_{\text{top}} \leftarrow -1$.
17: $C_{\cup}{}^{\text{top}} \leftarrow C_a{}^{\text{top}} \cup \bigcup_{i=1}^{I} C_{o_i}{}^{\text{top}} - C_{\text{top}} - C_{\text{skip}}$.
18: for each clip $c$ in $C_{\cup}{}^{\text{top}}$ do
19:     $S_{o_1}{}^{(c)}, \ldots, S_{o_I}{}^{(c)} \leftarrow$ retrieve Scores from $table_{o_1}, ..., table_{o_I}$, respectively
        w.r.t. CID $c$.
20:     $S_a{}^{(c)} \leftarrow$ retrieve the $Score$ from $table_a$ w.r.t. CID $c$.
21:     if $\sum_{i=1}^{I} S_{o_i}{}^{(c)} + S_a{}^{(c)} > S_{\text{top}}$ then
22:         $S_{\text{top}} \leftarrow \sum_{i=1}^{I} S_{o_i}{}^{(c)} + S_a{}^{(c)}$.
23:         $c_{\text{top}} \leftarrow c$.
24:     end if
25: end for
26: {Step 3} for each table, do reverse access in parallel from the $(stamp^{\text{btm}})_{\text{th}}$ row
    from the bottom and update $C_{o_1}{}^{\text{btm}}, ..., C_{o_I}{}^{\text{btm}}, C_a{}^{\text{btm}}$, until a new clip is found
    in all the tables.
27: {Step 4} perform random accesses to obtain the scores of all the seen clips and
    return the clip with the lowest score.
28: $C_{\text{top}}, C_{\text{btm}} \leftarrow C_{\text{top}} \cup c_{\text{top}}, C_{\text{btm}} \cup c_{\text{btm}}$.
29: return $c_{\text{top}}, S_{\text{top}}, c_{\text{btm}}, S_{\text{btm}}$.
```

Thus, TBClip obtains both highest ranking and lowest ranking clips every time is called. Moreover, while accessing the tables sequentially in parallel, we can *skip* (Lines 12 and 17) clips from further processing, if we are confident that these clips cannot contribute to sequences in the final top-K result. Such skipped clips belong to C_{skip} detailed in Section 4.3.1. Notice that set C_{skip} expands dynamically as the algorithm progresses. In TBClip, clips belonging to C_{skip} will only be accessed once during parallel sorted access, and be excluded from further processing (thus imposing no random access overhead in the ensuing execution).

5 Experimental Evaluation

In this section we present the results of our experimental evaluation of our proposals on a variety of data sets varying parameters of interest.

5.1 Experimental Setup

5.1.1 Models Our proposals are orthogonal to the underlying object/action detection and tracking models utilized. Our approach can work with any state of the art model utilizing the best in kind

Table 2: Action and object types queried in our evaluation on YouTube and Movie dataset.

Video	Action	Object	Length
YouTube (ActivityNet)	washing dishes	faucet, oven	57min
	blowing leaves	car, person	52min
Coffee and Cigarettes	smoking	wine glass, cup	1h 36min
Iron Man	robot dancing	car, airplane	2h 6min
Star Wars 3	archery	bird, cat	2h 14min
Titanic	kissing	surfboard, boat	3h 14min

models. In our experiments, for purposes of exposition, we select the following models as they have demonstrated solid performance. We stress however that our proposal can work with any model desired.

- **Mask R-CNN**. Mask R-CNN [22] is a state-of-the-art two-stage object detector trained on COCO dataset.
- **I3D**. We apply I3D [5], a state-of-the-art action recognizer trained on Kinetics dataset [4], in our proposed algorithms.
- **CenterTrack**. CenterTrack [52] is a real-time object tracker that localizes objects and predicts their associations with the previous frames.
- **Ideal Model**. In order to evaluate the accuracy of our algorithms excluding the influence of errors introduced by the object/action detection and tracking models utilized we also include experiments utilizing the *ideal models* that essentially generate detections matching the ground truth labels accurately.

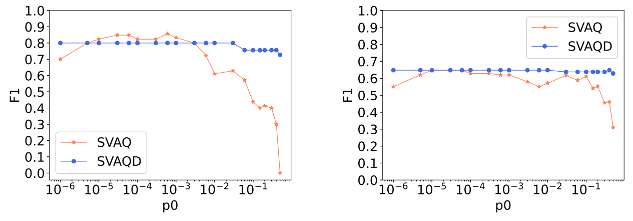
5.1.2 Datasets We use two video datasets in experimental study.

- **YouTube Videos**. ActivityNet [2] provides a large-scale YouTube video benchmark for human activity understanding, with 1.54 activity instances per video. We select 70 videos (about *washing dishes* or *blowing leaves*) in ActivityNet and manually label the range of different objects appearing in each activity for evaluating SVAQ and SVAQD. We evaluate the algorithms on a fixed set of queries depicted in Table 2.
- **Movies**. We select well-known movies and form queries involving actions and objects appearing in them (Table 2) for the evaluation of RVAQ.

5.1.3 Metrics For the evaluation of SVAQ and SVAQD, we present the F1 Score, evaluating the effectiveness of the algorithms to identify the correct sequences compared to the ground truth. We say that a sequence $\tilde{z}$ identified from our algorithm matches a ground truth sequence z, iff the intersection over the union (IOU) of the clips of the two sequences is above a threshold, η. We set the threshold η to 0.5 in our evaluation signifying substantial overlap between the sequences. The same threshold is utilized by object/action detection models [3, 5, 22, 30, 35, 41, 50]. A result sequence, $\tilde{z}$, is considered as a true positive if the IOU between $\tilde{z}$ and any sequence in the ground truth sequences is more than the threshold η; otherwise, it is a false positive. A ground truth sequence $\tilde{z}$, whose IOU with any result sequence is less than η, is considered a false negative.

We evaluate algorithm RVAQ, measuring the number of disk access to the clip score tables required to answer the queries.

5.1.4 Offline Case: Algorithms Compared For the offline case algorithm RVAQ utilizes both a forward and a backward pass on the suitable tables to obtain top/bottom clips and utilizes a *skip* mechanism to avoid accessing clips unnecessarily. For purposes of exposition we also present a comparison with the following algorithms:



(a) Query: {a=blowing leaves; o_1=car} (b) Query: {a=washing dishes; o_1=faucet}

Figure 2: Comparison of SVAQ and SVAQD on diff initial background probabilities (IOU=0.5).

- We utilize Fagin's Algorithm [11] to produce top ranking clips according to equation 12. Each clip as produced is checked against the ranges of clips in P_q. If the clip is not in any of the clip ranges in P_q it is disregarded. The score of each sequence in P_q is computed as its clips are produced and the algorithm stops when the score of each sequence in P_q has been produced and the final K sequences are returned. We refer to this algorithm as *FA*.
- We also include a variant of algorithm RVAQ without activating the *skip* mechanism in order to quantify the effectiveness of our proposal. We refer to this algorithm as RVAQ-noSkip.
- Finally we consider the algorithm that accesses all clips in the sequences of P_q (Equation 12), calculates the scores of each sequence and returns the K sequences with the highest scores. This algorithm accesses only the clips in the result sequences. We refer to this algorithm as the P_q-Traverse.

5.2 The Online Case

In our description of algorithms SVAQD and SVAQ there is an initialization of the background probability p_0. We first explore the sensitivity of the algorithms to this initialization. In Figure 2, we illustrate the performance of the two algorithms for two queries on the YouTube dataset, varying the background probability. The depicted queries are (a): {a=blowing leaves; o_1=car} and (b): {a=washing dishes; o_1=faucet}. For each experiment we initialize both algorithms with the same value of p_0 and report the associated F1 score for each query. As is evident in the Figure, SVAQD has very low dependency on the initial value of p_0 due to its adaptive design. In contrast, algorithm SVAQ has high dependency on the background probability value and some initial values affect its accuracy substantially. Thus, the benefits of the adaptive design of SVAQD are clear as the dependency on the initial value of the background probability are immaterial, making it the algorithm of choice.

We next present the F1 score for the two algorithms across several diverse queries. As a basis for comparison, we observe that the F1 scores of SVAQ as p_0 varies in Figure 2 peaks in the range $[10^{-5}, 10^{-4}]$. So we fix p_0 for SVAQ to $p_0 = 10^{-4}$ in this and subsequent experiments. Table 3 presents the F1 score for eight different queries on the YouTube video data set for the SVAQ and SVAQD algorithms. It is evident that SVAQD which dynamically updates p_0 provides superior performance.

Table 3: F1 scores of SVAQ and SVAQD for different queries.

Queries	SVAQ	SVAQD
a=blowing leaves	0.82	0.85
a=blowing leaves; o_1=car	0.83	0.85
a=blowing leaves, o_1=person	0.90	0.93
a=blowing leaves, o_1=car, o_2=person	0.83	0.88
a=washing dishes	0.86	0.88
a=washing dishes, o_1=faucet	0.63	0.65
a=washing dishes, o_1=oven	0.83	0.86
a=washing dishes, o_1=faucet, o_2=oven	0.62	0.66

Table 4: F1 score results of SVAQ and SVAQD with different detection models for the query q: {a=blowing leaves; o_1=car}.

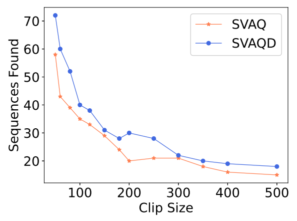
Methods	F1 Score
SVAQ (Mask R-CNN+I3D)	0.83
SVAQD (Mask R-CNN+I3D)	0.85
SVAQ (YOLO+I3D)	0.80
SVAQD (YOLO+I3D)	0.82
SVAQ (Ideal Models)	1.00
SVAQD (Ideal Models)	1.00

In table 4 we present the F1 score for the two algorithms adopting different detection models. As expected lower accuracy models (e.g., YOLO [36]) yield lower overall F1 score when adopted in our algorithms, versus more accurate models (e.g., Mask R-CNN [22]). We also present the accuracy of our algorithms assuming the *ideal models* that have perfect object and action detection accuracy (i.e., match the ground truth in all of their inferences). It is evident that the major source of inaccuracy for our algorithms is the errors introduced by the underlying object and action detection models. When that source of inaccuracy is removed (i.e., utilizing better models) our algorithms improve their accuracy in terms of identifying all results of interest compared to ground truth.

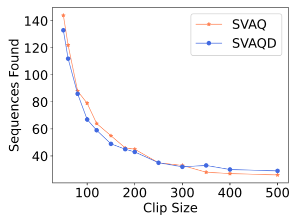
Finally we conduct experiments varying the clip size. The size of clips is a parameter in our setting and we wish to explore how its choice affects the results reported. In Figure 3, we report the number of result sequences identified by our algorithms with varying clip sizes for two different queries. As is depicted in Figure 3, it is evident that the prevailing trend is that with a smaller clip size we generate more result sequences (of smaller length) whereas with a larger clip size we will get fewer result sequences (of larger length). However upon close examination of the actual number of frames reported in each case, irrespective of the clip size, the total number of frames reported for each clip size remains stable. Thus the choice of the size of the clip primarily affects the number of result sequences reported, not the actual content of the sequences compared to group truth. In particular, if we evaluate the algorithms assessing the frame-level F1 score comparing the frames in the sequences retrieved to the ground truth, their accuracy exhibits low dependency on the clip size (as shown in Figure 4), which corroborates the fact that frames in the result sequences obtained by our algorithms are the same even if the clip size varies.

5.3 Performance of RVAQ

Once the query is provided along with the number of results required, the main objective is to produce the results sequences with the highest score fast. Since the video sequences have been preprocessed, the execution of the algorithm involves accessing information in secondary storage to produce the results. As such we report on the total number of accesses to secondary store required to produce the final answer.

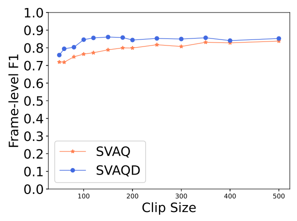


(a) Query: {a=blowing leaves; o_1=car}

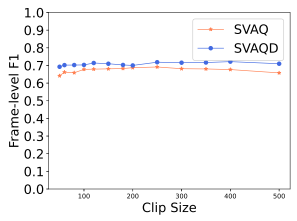


(b) Query: {a=washing dishes; o_1=faucet}

Figure 3: Sequences found by SVAQ and SVAQD on diff clip sizes.

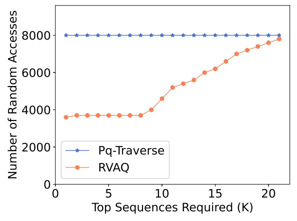


(a) Query: {a=blowing leaves; o_1=car}

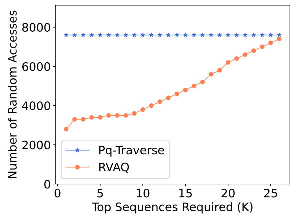


(b) Query: {a=washing dishes; o_1=faucet}

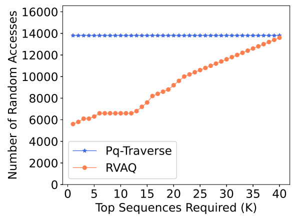
Figure 4: Frame-level F1 scores of SVAQ and SVAQD on diff clip sizes.



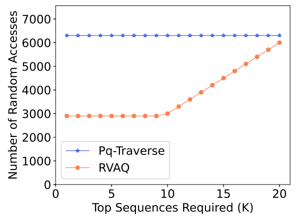
(a) Coffee and Cigarettes



(b) Iron Man



(c) Star Wars 3



(d) Titanic

Figure 5: The RVAQ Case

Table 5 presents the total number of disk accesses conducted by all approaches we consider (Section 5.1.4) for the movie, *Coffee and Cigarettes*, as per Table 2, as the number K of highest ranking results varies. Some observations are immediate; for algorithm FA as no lower bounds can be obtained as well as there is no way to skip unnecessary clips, the overall performance is substantially worse than all other approaches. In Algorithm RVAQ-noSkip, we observe that, although upper and lower bounds are obtained for the clips accessed, the inability to skip irrelevant clips is detrimental to the performance of the algorithm. For algorithm P_q-Traverse

Table 5: Number of disk accesses of diff methods on a movie, *Coffee and Cigarettes.*

Methods	Number of Disk Accesses (×1000)												
	K=1	K=2	K=3	K=4	K=5	K=7	K=9	K=11	K=13	K=15	K=17	K=19	K=21
FA	39.7	45.8	47.6	48.0	50.6	48.9	51.5	51.5	54.1	55.4	54.1	54.7	55.4
RVAQ-noSkip	24.9	25.9	26.2	27.4	27.4	27.4	27.4	27.4	27.4	27.4	27.4	27.4	27.4
P_q-Traverse	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0
RVAQ	3.6	3.7	3.7	3.7	3.7	3.7	4.0	5.2	5.6	6.2	7.0	7.4	7.8

the number of disk accesses is a constant irrespective of the value of K and proportional to the total number of clips contained in the result sequences. Overall, our proposed RVAQ is the most efficient and progressively its performance approaches closely that of P_q-Traverse when all sequences in the move are requested as expected (this video has 21 ground truth result sequences). Since the performance of FA and RVAQ-noSkip is almost an order of magnitude smaller that of RVAQ for small values of K we do not consider these algorithms further.

Figure 5 presents the total number of disk accesses conducted by algorithm RVAQ compared to P_q-Traverse as the number K of highest ranking results varies for four popular movies. The associated queries we issue are depicted in Table 2. We observe that RVAQ is at least two times faster compared to p_q-Traverse for small number of results K. As K increases gradually, since the query requires accessing all the clips of top-K sequences to obtain their exact scores, the number of accesses required by RVAQ increases to reach progressively that of P_q-Traverse, when all the results sequences in the video are required in the final answer. Note that in all experiments depicted, the largest value of K contains all the results sequences for the query in each video.

In terms of accuracy for all these results, since the movies considered in our evaluation have not been annotated by a third party, we manually annotated the results for the queries considered and inspect the ranked result sequences of RVAQ. In all cases for all sequences returned by RVAQ, the precision is above 81.0% and the associated F1 score is above 82.9%. These results are in line with the accuracy of SVAQD presented in the previous section (as compared to annotated ground truth in benchmark data sets). Moreover, in all cases, when $K = 10$, that is for the top-10 sequences ranked by Equation 11, the precision and F1 score are both 1, attesting to the superior accuracy (and associated performance) attainable by our proposals.

6 Related Work

In recent years, the application of deep learning to computer vision has received considerable attention. Many emerging solutions for object detection are presented, which can be roughly divided into two two main categories: one-stage methods (such as YOLO [36] and SSD [31]) and two stage-methods (such as Faster R-CNN [38] and Mask R-CNN [22]). YOLO [35, 36] is a unified detector casting object detection as a regression problem from image pixels to spatially separated bounding boxes and associated class probabilities. Mask RCNN [22] is proposed to tackle pixel-wise object instance segmentation by extending Faster RCNN. Several recent works address the problem of recognizing actions in videos [5, 14, 18, 51]. I3D [5] extends the network structure from two to three dimensions and proposes a two-stream inflated 3D ConvNet for action recognition. SlowFast [14] involves a Slow pathway to capture spatial semantics, and a Fast pathway to capture motion at fine temporal resolution. These works primarily train deep architectures of varying complexities end to end yielding models to classify and detect actions in video frame sequences. Our work readily utilizes object detection, action detection and tracking models [47, 52] in an online or offline manner to enable declarative query processing scenarios.

Several recent works present query processing frameworks utilizing frame content (objects, spatial locations in frame, etc) as first class citizens to query processing [7, 8, 16, 26, 33]. NoScope and BlazeIt [25, 27] utilize special purpose build neural networks (NNs) to detect objects accelerating queries via inference-optimized model search. Focus [24] implements low-latency search over large video data sets aiming to balance precision and query speed. SVQ [28, 48] provides a series of filters to accelerate video monitoring queries involving count and spatial constraints on objects present in the frames. Most current works focus on objects and relationships between them. Recent works, [6, 49] present declarative query processing on video streams involving objects and their interactions, inspired for object interaction frameworks [20, 34]. Our work follows this research thread proposing a framework to incorporate object detection and action detection in a declarative framework both for streaming and offline videos.

Efficient processing of top-k queries [11, 13, 17] is a well-studied task, demonstrating large performance benefits in multimedia search. The basic idea of this family of algorithms is to combine information from multiple sources utilizing a monotone aggregation function enabling early stopping when only a few highly ranked results are desired in the output. We utilize such algorithms in our proposal and demonstrate that applying them out of the box yield unsatisfactory performance. We demonstrate however that suitably adapting and augmenting them such approaches can also be utilized in a video query processing framework.

7 Conclusions

We considered the problem of online and offline queries on videos incorporating objects and actions present in the content of the videos as first class citizens for query processing. As off the self deep learning models for action recognition focus on the action itself and do not incorporate object constraints, we proposed a framework based on scan statistics to identify sequences of frames that contain all query specified objects and actions in a statistically sound manner.

We presented experiments utilizing real videos demonstrating the accuracy and performance benefits of our approaches. This work raises a few avenues for follow up work. First incorporating other types of action detection models in our framework (such as group action detection) should be investigated. In addition, queries incorporating interactions among objects along with actions present in the video feed are also interesting to explore further.

References

[1] Christopher M Bishop et al. 1995. *Neural networks for pattern recognition.* Oxford university press.

[2] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 961–970.

[3] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. 2018. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340* (2018).

[4] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. 2019. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987* (2019).

[5] Joao Carreira and Andrew Zisserman. 2017. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[6] Daren Chao, Nick Koudas, and Ioannis Xarchakos. 2020. SVQ++: Querying for Object Interactions in Video Streams. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data.* 2769–2772.

[7] Yueting Chen, Xiaohui Yu, and Nick Koudas. 2020. Evaluating Temporal Queries Over Video Feeds. *arXiv preprint arXiv:2003.00953* (2020).

[8] Yueting Chen, Xiaohui Yu, and Nick Koudas. 2020. TQVS: Temporal Queries over Video Streams in Action. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data.* 2737–2740.

[9] VNI Cisco. 2018. Cisco visual networking index: Forecast and trends, 2017–2022. *White Paper* 1 (2018), 1.

[10] Peter Diggle. 1985. A kernel method for smoothing point process data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 34, 2 (1985), 138–147.

[11] Ronald Fagin. 1999. Combining fuzzy information from multiple systems. *Journal of computer and system sciences* 58, 1 (1999), 83–99.

[12] Ronald Fagin. 2002. Combining fuzzy information: an overview. *ACM SIGMOD Record* 31, 2 (2002), 109–118.

[13] Ronald Fagin, Amnon Lotem, and Moni Naor. 2003. Optimal aggregation algorithms for middleware. *Journal of computer and system sciences* 66, 4 (2003), 614–656.

[14] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision.* 6202–6211.

[15] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. 2016. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[16] Daniel Y Fu, Will Crichton, James Hong, Xinwei Yao, Haotian Zhang, Anh Truong, Avanika Narayan, Maneesh Agrawala, Christopher Ré, and Kayvon Fatahalian. 2019. Rekall: Specifying video events using compositions of spatiotemporal labels. *arXiv preprint arXiv:1910.02993* (2019).

[17] Norbert Fuhr. 1990. *A probabilistic framework for vague queries and imprecise information in databases.* Technische Hochsch., Fachgebiet Datenverwaltungssysteme II.

[18] Jiyang Gao, Zhenheng Yang, Kan Chen, Chen Sun, and Ram Nevatia. 2017. Turn tap: Temporal unit regression network for temporal action proposals. In *Proceedings of the IEEE International Conference on Computer Vision.* 3628–3636.

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

[20] 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.

[21] Joseph Glaz, Joseph Naus, and Sylvan Wallenstein. 2001. *Scan Statistics* (1st ed.). Springer.

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision.* 2961–2969.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.

[24] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. 2018. Focus: Querying large video datasets with low latency and low cost. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18).* 269–286.

[25] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. Blazeit: Fast exploratory video queries using neural networks. *arXiv preprint arXiv:1805.01046* (2018).

[26] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. Challenges and Opportunities in DNN-Based Video Analytics: A Demonstration of the BlazeIt Video Query Engine.. In *CIDR.*

[27] 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.

[28] Nick Koudas, Raymond Li, and Ioannis Xarchakos. 2020. Video Monitoring Queries. In *2020 IEEE 36th International Conference on Data Engineering (ICDE).* IEEE, 1285–1296.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

[30] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. 2020. Deep learning for generic object detection: A survey. *International journal of computer vision* 128, 2 (2020), 261–318.

[31] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision.* Springer, 21–37.

[32] Joseph I Naus. 1982. Approximations for distributions of scan statistics. *J. Amer. Statist. Assoc.* 77, 377 (1982), 177–183.

[33] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. 2018. Scanner: Efficient video analysis at scale. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

[34] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. 2018. Learning human-object interactions by graph parsing neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV).* 401–417.

[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 779–788.

[36] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems.* 91–99.

[38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence* 39, 6 (2016), 1137–1149.

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

[40] Zheng Shou, Dongang Wang, and Shih-Fu Chang. 2016. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 1049–1058.

[41] Karen Simonyan and Andrew Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems.* 568–576.

[42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 1–9.

[43] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition.* 6450–6459.

[44] Ryan Turner, Zoubin Ghahramani, and Steven Bottone. 2010. Fast online anomaly detection using scan statistics. In *2010 IEEE International Workshop on Machine Learning for Signal Processing.* IEEE, 385–390.

[45] MJ Van Kreveld. 1996. *Variations on sweep algorithms: efficient computation of extended viewsheds and class intervals.* Vol. 1996. Utrecht University: Information and Computing Sciences.

[46] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision.* Springer, 20–36.

[47] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. 2019. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 1328–1338.

[48] Ioannis Xarchakos and Nick Koudas. 2019. Svq: Streaming video queries. In *Proceedings of the 2019 International Conference on Management of Data.* 2013–2016.

[49] Ioannis Xarchakos and Nick Koudas. 2021. Quering for Interactoins. In *2021 IEEE 37th International Conference on Data Engineering (ICDE).*

[50] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. 2019. A comprehensive survey of vision-based human action recognition methods. *Sensors* 19, 5 (2019), 1005.

[51] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. 2017. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision.* 2914–2923.

[52] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2020. Tracking Objects as Points. *CoRR* abs/2004.01177 (2020). arXiv:2004.01177 https://arxiv.org/abs/2004.01177