Scalable Complex Event Processing on Video Streams

CHENXIA HAN, The Chinese University of Hong Kong, Hong Kong
CHAOKUN CHANG, The Chinese University of Hong Kong, Hong Kong
SRIJAN SRIVASTAVA, Centre For Perceptual and Interactive Intelligence, Hong Kong
YAO LU, National University of Singapore, Singapore
ERIC LO, The Chinese University of Hong Kong, Hong Kong

The rapid expansion of video streaming content in our daily lives has rendered the real-time processing and analysis of these video streams a critical capability. However, existing deep video analytics systems primarily support only simple queries, such as selection and aggregation. Considering the inherent temporal nature of video streams, queries capable of matching patterns of events could enable a wider range of applications. In this paper, we present Bobsled, a novel video stream processing system designed to efficiently support complex event queries. Experimental results demonstrate that Bobsled can achieve a throughput improvement over state-of-the-art ranging from 2.4× to 11.6×, without any noticeable loss in accuracy.

CCS Concepts: • **Information systems** → **Stream management**; *Query optimization*; • **Computing methodologies** → *Activity recognition and understanding*.

Additional Key Words and Phrases: video stream analytics, complex event processing, action recognition, Markov chain models, sequential information maximization

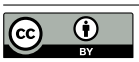
ACM Reference Format:
Chenxia Han, Chaokun Chang, Srijan Srivastava, Yao Lu, and Eric Lo. 2025. Scalable Complex Event Processing on Video Streams. *Proc. ACM Manag. Data* 3, 3 (SIGMOD), Article 214 (June 2025), 29 pages. https://doi.org/10.1145/3725419

1 Introduction

With the rapid expansion of video streaming content in our daily lives, the ability to process and analyze these video streams in real-time has become critically important, enabling rapid decision-making and intelligent automation. However, despite recent efforts, existing deep video analytics systems primarily support queries about simple events, such as selecting frames that contain a red car [5, 41, 44, 58] or counting the average number of cars in a video [5, 40, 44, 67, 73]. Considering the *temporal* nature of video streams, queries that can match *patterns of events* could enable a wider range of use cases and applications.

Transports Analytics In many places that permit a right turn on red light, it is a traffic offense if an individual fails to come to a complete stop before making the right turn [18, 55, 71]. Although such a 'right-turn-after-rolling-stop' behavior is not uncommon, it poses significant risks to pedestrians on the adjacent right sidewalk. Spotting such an offense requires identifying two events, 'rolling stop' and 'right-turn', and matching them against a temporal 'happen-after' pattern.

Authors' Contact Information: Chenxia Han, The Chinese University of Hong Kong, Sha Tin, Hong Kong, cxhan@cse.cuhk.edu.hk; Chaokun Chang, The Chinese University of Hong Kong, Sha Tin, Hong Kong, ckchang@cse.cuhk.edu.hk; Srijan Srivastava, Centre For Perceptual and Interactive Intelligence, Tai Po, Hong Kong, s.srivastava@cpii.hk; Yao Lu, National University of Singapore, Singapore, Singapore, luyao@comp.nus.edu.sg; Eric Lo, The Chinese University of Hong Kong, Sha Tin, Hong Kong, ericlo@cse.cuhk.edu.hk.



ACM 2836-6573/2025/6-ART214
https://doi.org/10.1145/3725419

Sports Analytics During sports competitions, coaches monitor the game and adjust tactics based on the on-field situations [66, 77, 91]. For instance, in basketball, 'pick-and-roll' is a commonly used offensive play to create scoring opportunities. It involves a player A setting a 'screen' by picking a defender, which creates an opportunity for either (1) a teammate B to take an open shot or (2) player A to move ('roll') towards the basket and receive a pass. To identify such plays, it is necessary to extract all events of 'screen', 'roll', 'pass', and 'shoot', and match them according to a specific pattern within a specific time window (e.g., 5 seconds).

Shopping Behavior Analysis The way goods are arranged in a store can greatly influence customers' buying decisions. Therefore, it is important to analyze shopping behaviors in order to understand customer preferences and develop strategies to boost sales [37, 64]. For instance, if it is observed that a significant number of customers head to the checkout area after picking up beers, separating the beer section from the checkout counters could potentially increase the customers' shopping time. Identifying such behaviors cannot be accomplished using traditional techniques that rely on mining transactional data (e.g., association rules [1, 2]), as many interesting events occur before customers check out. Instead, these behaviors can be identified by composing and evaluating various pattern matching queries against a large volume of videos.

Matching complex patterns across different events in a video requires a deep understanding of the video content. Despite the achievements of recent Multi-modal Large Language Model (MLLM) such as GPT-4 [65], Gemini [31], LLaMA-VID [53], and LLaVA [56], their pattern matching abilities remain rudimentary [39, 61, 80]. More importantly, these models process each query by treating each frame from the video-of-interest as token(s) in the prompt. In other words, every video frame must pass through the large model with billions of parameters, resulting in exceptionally high processing costs or processing times.

Alternatively, one can rely on single-modality models and fast selection techniques such as Zeus [17] and Probabilistic Predicate (PP) [58] to first localize all the individual interesting events (e.g., 'rolling stop', 'right-turn'). Afterwards, one can utilize a Complex Event Processing (CEP) engine [3, 11, 48, 76, 88, 93] as a post-processing step to match the query pattern. However, this "localize-all-post-CEP" approach would over-localize many unnecessary events that end up not matching the query pattern, thereby wasting valuable GPU resources. An alternate approach is to directly train or fine-tune a single-modality binary model for each specific query pattern. However, this approach is not scalable either, especially when there are numerous potential query patterns to consider.

When it comes to scalable video analytics, the database community has been actively developing solutions that offer improved processing cost and efficiency [5, 17, 40–42, 50, 51, 58]. However, existing works in this field have not addressed the challenge of handling pattern matching queries. Instead, the primary focus of these works revolves around simpler queries such as selection [17, 41, 50, 58], top-k [51, 72], and aggregation [40, 73]. Most of these works accelerate processing by a *cascading* approach, which involves leveraging a lightweight but less accurate "proxy" model to first quickly inference the data once and using those approximate inference results to guide the search [40, 41, 44, 51, 58] or the sampling process [42, 62]. The more accurate but expensive "oracle" model is involved only when necessary.

In this paper, we adopt the cascading approach and introduce Bobsled, a resource-efficient video analytics system designed for complex event processing. Bobsled uses an event pattern language [1, 89] as its query language, consisting of constructs that allow precise expression of various pattern queries such as windowing [32, 94] and Kleene closure [33].

Unlike many existing work [40, 41, 51, 58], Bobsled does not require training or fine-tuning an oracle/proxy model for each query. Initially, it employs a lightweight *pretrained* single-modality model as a proxy to obtain a distribution of events for each clip in the video stream. Using this

	c_1	c_2	c_3	c_4
RS	**0.99**	0.05	0	0
FS	0.01	**0.80**	0	0
RT	0	0.15	**0.70**	0.03
LT	0	0	0.30	**0.97**

(a) Case 1

	c_1	c_2	c_3	c_4
RS	**0.99**	0.25	0	0.10
FS	0.01	0.03	0.15	**0.90**
RT	0	**0.72**	0	0
LT	0	0	**0.85**	0

(b) Case 2

Fig. 1. Adaptive Event Materialization (FS stands for 'full-stop', an event not included in the query)

information, it derives the probability p of a window matching the query pattern and forms a cascade: If p falls below a certain threshold, a clip within a window is discarded without further processing by the oracle. Deriving the probability p in the context of pattern matching, however, is actually non-trivial. Specifically, matching a window of events against a pattern requires translating the query pattern into a *nondeterministic finite automaton* (NFA) [89] and consuming the window of events to determine if the final state can be reached. However, since a pattern query can be translated into an NFA with **non-mutually exclusive** outgoing transitions [1], it cannot be directly used to form a probabilistic model like Markov chain to estimate p (as non-mutually exclusive edges would sum up probabilities to more than 1). Bobsled utilizes Non-Homogeneous Markov Chains [8] and develops the concept of *Probabilistic Pattern* (PPAT) to address this challenge.

For windows that cannot be directly pruned by PPAT, Bobsled carries out further optimization in order to minimize the inference cost. For example, consider a query that searches for the following three-event pattern on windows of size four (i.e., four clips per window):

'rolling stop' (RS) → 'right turn' (RT) → 'left turn' (LT)

Consider two different cases for a window shown in Figure 1. The window covers four clips c_1 to c_4. After scanning by the proxy, the discrete distribution of potential events for each clip is obtained.

In Case 1, a smart Adaptive Event Materialization (AEM) strategy can choose to select clip c_2 and pass it to the oracle first to materialize its real label. By doing so, it can reject the window early once the oracle confirms that c_2 is not a 'right turn'. This is because a non-'right turn' event in the second position of the window immediately violates the query pattern. Such a smart approach enables early stopping, incurring the cost of passing only one clip to the oracle, and can then proceed to processing the next window.

In contrast, a basic strategy would be to sequentially pass the clips inside the window to the oracle, starting from c_1. However, if c_1 is identified as a 'rolling stop' according to the oracle, then unlike the smart strategy mentioned above, the window cannot be discarded until it also materializes c_2 and finds that c_2 is not a 'right turn'. This would incur a cost of materializing both c_1 and c_2 before the window can be rejected.

Designing a smart materialization strategy is non-trivial because the optimal choice not only depends on the early rejection probability, but also depends on the early acceptance probability if the window contains a positive match. Consider case 2 in Figure 1. In this case, after materializing c_1 and c_2 as 'rolling stop' and 'right turn', a smart strategy shall pick c_3 over c_4 because it can potentially early accept the window without processing c_4.

Given the complexity and probabilistic nature of the decision space, Bobsled formulates the problem as a Sequential Information Maximization (SIM) problem [13]. This formulation enables a greedy policy, allowing Bobsled to make near-optimal choices by maximizing the mutual information among different materialization choices.

To our knowledge, Bobsled is the first system that can effectively **handle pattern matching queries on video streams**. Bobsled accelerates the complex event processing while meeting the desired accuracy. Experiments indicate that Bobsled can achieve a throughput improvement of 2.4× to 11.6× without any noticeable loss in accuracy. This boost in throughput translates to **enhanced resource efficiency**, allowing the same hardware resource to handle a significantly higher number of concurrent queries/video streams.

The rest of the paper is organized as follows. Section 2 gives the background and related work of this paper. Sections 3 to 8 present the details of Bobsled. Section 9 presents the experimental results and Section 10 presents the conclusion.

2 Background and Related work

2.1 Video Event Detection

The transformer model [85] has literally transformed the world in recent years. The latest Multi-modal Language Models (MLLMs) such as GPT-4 [65], Sora [7], Gemini [31], LLaMA-VID [53], and LLaVA [56] have the ability to manage tasks like text-to-video generation [7] and video question answering [19, 53, 56]. However, current MLLMs lack sophisticated pattern matching capabilities [39, 61]. Even with future advancements, the inherent need to process every frame might render MLLMs computationally expensive for data-driven video analytics. Employing the broader capabilities of MLLMs for complex event video processing could be unnecessarily resource-intensive. Furthermore, event pattern languages can precisely capture query intents, whereas achieving the same level of precision through natural language prompts is almost an art. Therefore, the language modality in MLLMs is not particularly necessary for our applications but brings unnecessary overhead due to the significant proportion of model parameters allocated to natural language processing. For detecting individual events in a video, single-modality video recognition models are sufficient.

Bobsled supports complex video event processing using single-modality models. It is agnostic to the type of event, whether it is an action detected by a video recognition model, or an object occurrence detected by an image model. In the following, we focus on action-based events because they align better with our interested use cases.

Video recognition models take consecutive *clips* (sequences of frames) as input and leverage both the spatial information within each frame and the temporal information across frames for recognition. These models can be broadly categorized into two classes: convolution-based (e.g., C3D [82], R(2+1)D [83]) and transformer-based (e.g., VideoMAE [81], MViT [26]). Convolution-based models use 3D convolution instead of 2D convolution to capture temporal information in video clips. On the other hand, transformer-based models tokenize the input video and employ self-attention mechanisms to capture contextual information.

Inputs to these video recognition models are typically clips of 16, 64, or 128 consecutive frames. These models have the ability to identify multiple events occurring within a clip. For example, given a clip, they can output co-occurring events such as ‘player A is screening’ and ‘player B is dribbling’. Furthermore, since an action may span multiple clips, a model may output an ‘eating’ label for hundreds of consecutive clips.

2.2 Deep Video Analytics

The database community has developed a series of systems to optimize query processing on videos, supporting various query types including selection [5, 10, 14–17, 23, 41, 42, 49, 50, 58, 62, 90], limit [35, 40, 62], aggregation [40, 43, 73], and top-k [12, 51, 72]. Most of these systems are based on building *query-specific* proxy models. For example, they train a lightweight binary classifier for a

query predicate, say, `COUNT(CAR) > 5`, as a "proxy". Frames with a *proxy score* (i.e., the probability of a frame passing the query, according to the proxy model) below a certain *passing threshold* are discarded, thus avoiding the need for processing by the expensive oracle model.

The training data of the proxy is obtained by sampling frames from the queried video, passing them to the oracle, and regarding the oracle outputs as the true labels. The passing threshold is calibrated based on relating the empirical recall/precision on the proxy model with respect to the desired recall/precision targets, based on different levels of accuracy guarantees [23, 41, 42, 58].

Proxy models are not the sole method for providing proxy scores. TASTI [44] and Seiden [5] employ an alternative approach by initially processing a subset of frames using the oracle model. They then use the labels obtained from this processed subset to interpolate proxy scores for the remaining frames. These techniques can be applied to any proxy-model based system including Bobsled. ExSample [62] can answer LIMIT queries without relying on any proxy scores. It uses Thompson sampling to select frames from the queried video for processing by the oracle model and stops early once a sufficient number of results are found. However, for selection queries or pattern queries that aim to find all answers, determining when to stop the search early while still meeting a specific recall or precision target remains unclear, especially when no additional information, such as proxy scores, is available.

Some systems [10, 17, 49] are designed to adaptively select the most appropriate model or configuration (e.g. image size, sampling rate) from a collection for inference. Their goal is to process important frames using more expensive models or configurations, while using cheaper models or configurations for irrelevant frames. For example, Zeus [17], which is dedicated to action localization, leverages a reinforcement learning agent to predict the configuration of the next video clip to process. Other works, such as Rekall [29] and VOCAL [20], have primarily concentrated on the programming model and data labeling processing for complex event processing in videos. In contrast, our work pioneers the exploration of the efficiency aspect of pattern matching in videos.

2.3 Complex Event Processing

Complex Event Processing (CEP) systems [22, 25, 34, 52, 57, 59] are designed for pattern matching in conventional data streams. CEP queries comprise a sequence of events that occur in a specific order along with constraints on these events. Various query languages have been proposed to specify patterns, including SQL-TS [75], Cayuga [6], and SASE+ [21]. These languages support expressing event sequences, Kleene closure, complex predicates, and contiguity, thereby providing richer expressibility compared to languages for regular expression matching.

A typical CEP query contains three clauses 'SEQ', 'WHERE', and 'WITHIN'. The 'SEQ' clause defines a sequence of events, along with associated quantifiers, which defines the minimum and maximum occurrences of an event. One quantifier is *Kleene plus*, denoted by '+', enables definition of one or more occurrences of an event, such as 'A+'. Specific number of occurrences of an event can also be defined. For instance, 'A{3, 5}' indicates that the event 'A' should occur at least three times and at most five times. The 'WHERE' clause specifies conditions on events, whereas the 'WITHIN' clause specifies the window size.

```
SEQ (A, B+, C) //Example Query EQ1
WHERE A.condition = 'sunny'
AND B.condition = 'rainy'
AND C.condition = 'cloudy'
WITHIN 1 hour
```

Query EQ1 above is an example CEP query, capturing a sequence of weather conditions: within the past hour, the weather condition changes from 'sunny' to 'rainy' for a period of time, and finally

turns to 'cloudy'. The Kleene plus quantifier matches one or more 'rainy' events. By default, the *contiguity* between these events is *relaxed* [21]. *Strict contiguity* requires two events to be contiguous in the stream, which is commonly used in regular expression matching. *Relaxed contiguity* removes the contiguity requirements between events, where all irrelevant events are skipped until the next relevant event is found. This is particularly important in real-world scenarios where the streams contain noise events that should be ignored by the query.

During the execution, a CEP query is usually compiled into a *nondeterministic finite automaton* (NFA). Formally, the NFA can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q denotes a set of states, Σ represents a set of conditions, δ is a transition function, q_0 is the start state, and F is the final state. The transition function, δ, defines the transitions of an NFA. For instance, $\delta(q, a) = q'$ refers to the transition from state q to state q' under the condition of a. The condition is a boolean expression composed of basic operators ($\wedge$, $\vee$, $\neg$), comparison operators (e.g., $>$, $<$, $=$), and variables, which evaluates to either true or false. An *epsilon* transition models the Kleene plus operator under relaxed contiguity, allowing the matching process to move from one state to another state without consuming any event; its condition is always evaluated to true.

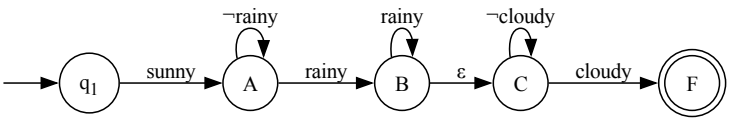


Fig. 2. NFA compiled from EQ1

Figure 2 depicts the NFA compiled from EQ1. In this example, the condition 'sunny' must be satisfied for the matching process to transition from state 'q_1' to 'A'. There is an epsilon transition from state 'B' to state 'C', which can take place without consuming any event. Hence, at state 'B', a 'rainy' event can either stay at state 'B', or first transits to state 'C' via the epsilon transition and then stays at state 'C' after consuming the 'rainy' event via the '¬cloudy' transition. This highlights the inherent non-deterministic exhibited by NFAs, where the conditions associated with two edges at a particular state can be non-mutually exclusive.

A match is found if there exists some choices of transitions that lead to the final state after consuming some events. If no match is found in the current window, the matching process *slides* the window to the next event. For instance, from X[ABDD]EE to XA[BDDE]E, where '[' and ']' denote the start and end of a window. When a match is found, the matching process has different options to move the window. For example, the *skip-past-last-event* strategy [27] moves the window to skip past the last event that matches, from X[ABCD]EEFF to XABC[DEEF]F.

The primary focus of traditional CEP systems is to enhance throughput, thereby facilitating the processing of more data streams at real-time. The key challenge lies in the complexity of evaluating the NFA on each window. Various work addresses this in terms of pattern matching cost and memory consumption [1, 4, 9, 60, 68, 98]. However, CEP on video events should have a different focus because the primary bottleneck stems from the model inference rather than the pattern matching process. For instance, a typical video recognition model like MViTv2-S [54] achieves 52 clips per second on an Nvidia GeForce RTX 3090 GPU when using 16 frames per clip. In contrast, CEP systems are capable of handling over 100,000 events per second when running on a single CPU core [9]. Therefore, the challenge shifts towards minimizing model inferences to maximize the system's overall throughput.

3 Bobsled

Problem Definition. The problem addressed in this paper can be formally defined as follows:

Given a set of video streams D, a set of pattern queries Q, an oracle model M, and an accuracy target T, the goal is to maximize the overall throughput in terms of frames per second (fps) for processing Q over D while ensuring that the F1-score of each result $R(q)$ of query $q \in Q$ is at least T, using the original query result $R_o(q)$ as the reference of true positive and true negative matchings.

Consistent with existing work on video analytics [5, 40–42, 51], Bobsled considers the labels from the oracle model as the "ground truth". The primary objective of Bobsled is to accelerate complex event processing on video streams while producing results that closely match those obtained using the oracle model.

Event Space and Vision Model. Bobsled aims to support scalable complex event processing on video streams. It supports events recognized by any recognition model including image classification, action recognition, object detection, and others. Currently, events that are not direct outputs of a recognition model are not supported. For instance, the event '*more than five cars*' is not a direct output of any common object detection model, but rather an event that is post-processed by an external module based on the model's direct output. However, if a model is fine-tuned and can directly return such a '*more than five cars*' event, Bobsled can also support it. The reason is that Bobsled avoids any query-specific proxy models, as that approach is not scalable with the number of queries. Instead, Bobsled utilizes proxy models that perform the same tasks as the oracle models. These models are often pre-trained and provided alongside the oracle model. For example, the latest version of YOLOv10 [87] object detector has 5 variants: `nano`, `small`, `medium`, `large`, and `extra-large`. One can use the `nano` version as the proxy and the `extra-large` version as the oracle. When lightweight versions of an oracle model are unavailable, users can utilize a model quantization tool [69] to prepare one.

Query Language. Bobsled implements a subset of the Flink CEP Pattern API [27] and adopts it for complex event processing on video streams:

```
query = <component>
    [.<contiguity><component>]*
    [.window(<window-size>)] //in # of clips
    [.f1(<f1-target>)]
    .model(<oracle-model>, <proxy-model>)
component = (<condition>)
    [.minDuration(<minimum-duration>)] //in # of clips
    [.maxDuration(<maximum-duration>)] //ditto
contiguity = followedBy | next
```

Compared with Flink's CEP API, one modification we made is introducing the 'model' keyword, which requires the user to specify the oracle and proxy models. Another modification is adding the 'f1' keyword, which enables acceleration by requiring users to specify the target F1 score, with a default value of 0.99. As an example:

```
('rolling stop').minDuration(1) //Example Query EQ2
.next('right turn').minDuration(1)
.followedBy('left turn').minDuration(1)
.window(5)
.f1(0.99)
.model(mvit-v2-s, c3d-3)
```

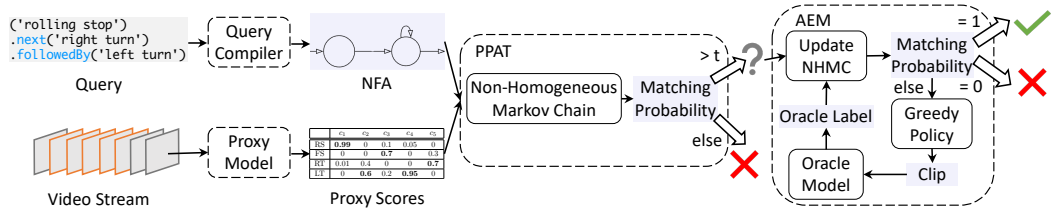


Fig. 3. Overview of Bobsled

Other keywords are all inherited from Flink CEP. The keywords 'minDuration' and 'maxDuration' specify the minimum and maximum duration of an event, respectively. For example, 'minDuration(1)' indicates matching at least one occurrence of the event, which is equivalent to a Kleene plus quantifier. As another example, ('right turn').minDuration(4) indicates that the 'right turn' event must consecutively appear in at least 4 clips in order to consider as a match. Regarding contiguity, strict contiguity is specified by the keyword 'next', while relaxed contiguity is specified by 'followedBy'. The window size is determined by the 'window' keyword, with a default value of 30 seconds in Bobsled if not specified.[1] In Bobsled, once a match is found, the *skip-past-last-event* strategy is employed to prevent the return of overlapping results, thereby improving result diversity.

4 System Overview

Figure 3 illustrates the system overview of Bobsled. The system consists of four components: query compiler, proxy model, probabilistic pattern (PPAT), and adaptive event materialization (AEM). Bobsled aims to support more queries and more video streams given the same hardware (GPU) resources. However, for clarity, we present the details from the perspective of a single query on a single stream.

Query Compiler. In addition to translating the query into NFA, the compiler also infers the minimum duration for an event in the query if the user has not specified it. This is necessary because user-perceived events in a video, such as actions, are often mapped from multiple raw model-predicted events that span across multiple clips. For instance, when using a clip size of 16 frames (equivalent to approximately 0.5 seconds), a user-perceived 'eating' action in a video is often represented by several model-predicted 'eating' labels spanning multiple consecutive clips. Determining the precise minimum duration for different types of events can be challenging for query users (e.g., what should be the minDuration for a 'drinking' event or an 'eating' event?). Making an uninformed decision could compromise either accuracy (by setting the minimum duration too large) or efficiency (by setting the minimum duration to 1, resulting in reduced selectivity and unnecessary overhead due to prodigal use of the Kleene plus operator). Therefore, in Bobsled, the compiler automatically infers the minimum duration for all events with unspecified minDuration in the query (section 5).

Proxy Model. The proxy model is a lightweight but less accurate model to first quickly inference the video and generate a discrete distribution of potential events for each clip (e.g., see Figure 1). There are no specific requirements for the proxy model in relation to the oracle model, other than that the proxy (1) shares the same event space as the oracle model and (2) operates faster than the oracle model. However, the degree of performance improvement depends on the divergence in distribution between the two – the smaller the divergence, the greater the improvement should be

[1]In Bobsled, number of clips, number of frames, and time duration (e.g., seconds, minutes) are interchangeable units.

expected. In practice, this divergence is often reasonable, enabling the prior works that leverage the proxy model for acceleration, including Bobsled.

PPAT. The NFA, together with the proxy scores predicted by the proxy model, is used to construct a Non-Homogeneous Markov Chain (NHMC) that serves as a Probabilistic Pattern (PPAT) (section 6) for estimating the probability of a window matching the query pattern. Windows with a high likelihood of not matching the query pattern are filtered early, without performing any additional processing on the clips within the window. The PPAT component determines the decision by deriving a *rejection threshold* from the target F1 accuracy specified in the query.

In NoScope [41], there is another threshold known as the *acceptance threshold*. Windows with probabilities higher than the acceptance threshold are immediately returned without any additional processing. Accepting based on the acceptance threshold carries the potential for false positives and can impact precision. Similarly, filtering based on the rejection threshold can potentially result in false negatives and impact recall. Given that Bobsled utilizes the F1 metric, which incorporates both precision and recall, as the target accuracy; and considering that complex event queries tend to be more selective than simple event queries, our empirical results indicate that it is not worthwhile to allocate materialization budget for early accepting a window. Hence, PPAT only establishes the rejection threshold at this stage, prioritizing the filtration of negative windows and deferring the remaining processing to the Adaptive Event Materialization (AEM) module.

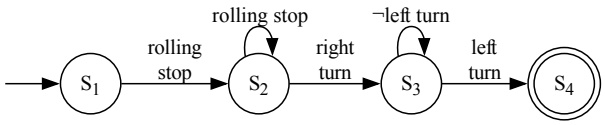
AEM. Windows that cannot be filtered by PPAT are passed to AEM for further inference (section 7). AEM aims to select the minimal number of clips to be processed by the oracle model that can potentially lead to early rejection or early acceptance of the window. For example, consider Query EQ2 in the previous section, which looks for windows of size five where the first event is a 'rolling stop' (RS), the second event is a 'right turn' (RT), followed by a 'left turn' (LT) event that occurs in any other position. Now, let us consider the window of events in Figure 4(a). We use '?' to denote a clip whose event that has not been materialized by the oracle model. Given the window in Figure 4(a), if c_2 is first selected for oracle inference and confirmed as a 'left turn' event, the entire window [? LT ? ? ?] can be *rejected* early without needing to materialize the other clips. Alternatively, if clips c_1, c_2, and c_4 are chosen for oracle inference with the following results [RS RT ? LT ?], we can *accept* the window early without needing to materialize clips c_3 and c_5.

In a nutshell, given a window, AEM aims to pick the next clip that maximize the early acceptance probability if the window eventually matches the query, and conversely, picks the one that maximize the early rejection probability if the window eventually does not match the query. However, it is unknown whether a window matches the query or not in advance. To address this, we employ information entropy to unify these two cases and solve it as a *Sequential Information Maximization* (SIM) problem [13] using a near-optimal solution (section 7).

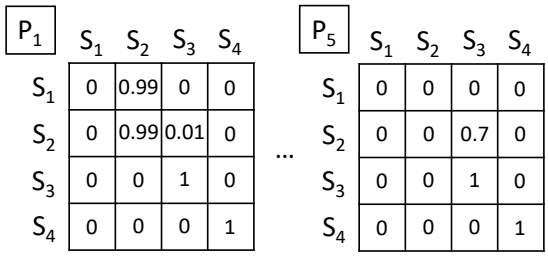
Optimization Focus. Bobsled's optimization focus is on the *entire query in terms of the number of accesses to the deep model*, rather than on any individual operator. Specifically, the effectiveness of Probabilistic Pattern (PPAT) is inherently tied to the **selectivity** of the query pattern – the more selective the query, the more powerful PPAT becomes. Similar to PPAT, the optimization power of Adaptive Event Materialization (AEM) also depends on the selectivity. However, unlike PPAT, AEM also has the ability to accept positive windows early, making it more general than PPAT and its effectiveness less sensitive to selectivity. Although the effectiveness of both PPAT and AEM depend on selectivity, they operate in fundamentally different dimensions: PPAT focuses on window filtering and aims to minimize the number of windows that need to be processed by AEM (i.e., window pruning). In contrast, AEM aims to utilize the difference between the query pattern length and the window size to minimize event materialization for the windows passed by PPAT (i.e., event pruning).

	c_1	c_2	c_3	c_4	c_5
RS	**0.99**	0	0.1	0.05	0
FS	0	0	**0.7**	0	0.3
RT	0.01	0.4	0	0	**0.7**
LT	0	**0.6**	0.2	**0.95**	0

(a)



(b)



(c)

Fig. 4. (a) An example window of five clips. (b) DFA converted from the NFA of Query EQ2. (c) Non-Homogeneous Markov Chain constructed for Query EQ2.

5 Query Compiler

The query compiler in Bobsled follows the implementation in SASE+ [21], which compiles each query into an NFA first. Notably, we do not implement the match buffer in SASE+, as model inference is the primary bottleneck in Bobsled and a translation to the basic NFA is sufficient for our purposes.

Since each user-perceived event is derived from multiple consecutive model-predicted events, the query compiler automatically annotates the minimum duration for each event if it is not specified in the query before translating the query to an NFA. Specifically, Bobsled follows [58] to first sample clips from the queried video and utilize the oracle model (or human labels, if available) to identify the events within these clips; after that, it derives the minimum duration accordingly. For instance, if the 'rolling stop' event appears five times in the samples with duration of 6, 7, 5, 7, and 10, then the minimum duration of 'rolling stop' is 5. All events whose corresponding minimum duration is unspecified in the query will have the inferred minimum duration annotated after the compilation.

6 Probabilistic Pattern

Given a window of *known* events, it is straightforward to determine whether a match exists by evaluating the query's nondeterministic finite automaton (NFA). The NFA returns a match if a series of transitions leads to the final state. However, since we want to reduce accesses to the oracle model, we employ a proxy model to obtain the discrete distribution of each event, and use those *uncertain* events to build a Probabilistic Pattern (PPAT) for calculating the pattern matching probability and early discard the improbable ones.

A naive approach to estimate the probability $\Pr(W \models q)$ that a window W matches a query pattern q is to enumerate all *possible worlds* [63] of the window based on the set of possible events and aggregate the probabilities of all worlds that match the query: $\Pr(W \models q) = \sum_{\hat{W} \models q} \Pr(\hat{W})$, where $\hat{W}$ is a possible world of window W that matches the query. However, this naive approach is computationally expensive as it involves $|\mathcal{Y}|^w$ worlds, where $|\mathcal{Y}|$ is the event space and w is the

window size. To address this challenge, we formulate the problem as a Non-homogeneous Markov Chain (NHMC) [8], which allows us to significantly reduce the computational complexity.

6.1 Non-Homogeneous Markov Chain as the Probabilistic Pattern

Given a query pattern q, we construct a Non-Homogeneous Markov Chain (NHMC) for each window W to estimate the probability $\Pr(W \models q)$ that W matches the pattern q.

Non-Homogeneous Markov Chain is a variant of Markov Chain. In Markov Chain, the sum of transition probabilities from any given state must equal 1. However, since the transitions from an NFA compiled from the query can be non-mutually exclusive, the NFA cannot directly serve as the state-space and transitions of any Markov Chain [74].

To build the state-space and transitions of an NHMC, it is necessary to use an automaton that is equivalent to the query pattern (NFA), but with mutually exclusive transitions. In this regard, a Deterministic Finite Automaton (DFA) fits the bill. In Bobsled, we adopt the powerset construction algorithm [70] to obtain a DFA from the initial NFA. The DFA can be represented by the 5-tuple $(Q', \Sigma', \delta', q_0', F')$, where Q' is a set of states, Σ' is a set of conditions, δ' is the transition function, q_0' is the start state, and F' is the final state. For any two states S_i and S_j, there is either exactly one transition from S_i to S_j with the condition σ such that $\delta'(S_i, \sigma) = S_j$, or no transition at all.

Figure 4(b) illustrates the DFA derived from the NFA compiled for example Query EQ2. Transitions in this DFA are mutually exclusive. The states and transitions of the DFA can serve as the basis for the states and the transitions of the NHMC, with the transition probabilities derived from the proxy scores. However, the probabilistic pattern matching cannot be modeled by a basic Markov Chain alone, as the transition probabilities are *non-homogeneous*. Specifically, the transition probability from one state to the other is not static, but instead depends on the current clip-of-interest. For example, consider the DFA in Figure 4(b) and the window in Figure 4(a). If clips c_1 and c_2 have been consumed and the current state has reached S_3, then the transition probability that it stays at S_3 is $\Pr(\neg\text{left turn}|c_3) = 0.1 + 0.7 + 0 = 0.8$. However, assuming the state remains at S_3 after consuming c_3, then the transition probability that it stays in S_3 again should be $\Pr(\neg\text{left turn}|c_4) = 0.05$ instead. This explains why we use Non-Homogeneous Markov Chain as PPAT.

In an NHMC, the transition probabilities are represented by a set of transition matrices, each specific to a given time. Here, we use the discrete-time NHMC. Let the set of states be $S = \{S_1, S_2, \ldots\}$. Each transition matrix P_t is a square matrix with dimensions $|S| \times |S|$, where $|S|$ is the number of states in the NHMC. The (i, j)-th entry of the t-th transition matrix indicates the probability of transitioning from state S_i to state S_j at time t. Each transition matrix P_t is constructed using the proxy scores of the clip c_t.

To build the transition matrix P_t, we define a random variable e_t to represent the possible events in clip c_t, with its distribution determined by the proxy scores. The entry $(P_t)_{i,j}$ is set to the probability of transitioning from state S_i to state S_j given the condition σ: $(P_t)_{i,j} = \Pr(\sigma|e_t)$. If there is no transition from state S_i to state S_j, then $(P_t)_{i,j}$ is set to 0. Here, $\Pr(\sigma|e_t)$ refers to the probability that the condition σ holds given the distribution of e_t. Consequently, a window of w clips forms an NHMC with w transition matrices.

One caveat is about the final state in the DFA, as Markov Chains inherently do not have terminal states where the process ends. To address this, we add a self-transition with a probability of 1 to the final state obtained from the DFA. Formally, this is achieved by setting $(P_t)_{i,i}$ to 1 if S_i is the final state. By using this method, a valid NHMC can be formed based on the query and the proxy scores.

Figure 4(c) illustrates the NHMC and some of its transition matrices for Query EQ2 based on the window in Figure 4(a). For example, consider the entry $(P_1)_{1,2}$ in the transition matrix P_1. This entry denotes the probability of transitioning from the initial state S_1 to state S_2 is 0.99 at the beginning of processing the window because the first clip c_1 has a probability of 0.99 to be a 'rolling-stop'.

Similarly, consider the entry $(P_5)_{2,3}$ in the transition matrix P_5. This entry denotes the probability of transitioning from state S_2 to state S_3 is 0.7 after consuming c_1 to c_4, where c_5 has a probability of 0.7 being a 'right turn'.

Given an NHMC, the probability that the window w matches the query q is equivalent to the probability of transitioning from the start state (q'_0) to the final states (F'). In NHMC, the transition probability after w steps is calculated by multiplying the w transition matrices together. To extract the specific transition probability from the start state to the final state, we employ two indicator vectors u and v. Here, u is an indicator vector corresponding to the start state q'_0. Specifically, $u_i = 1$ if $S_i = q'_0$ and all other entries in u are 0. Similarly, v is an indicator vector corresponding to the final state F'. Specifically, $v_i = 1$ if $S_i = F'$ and all other entries are 0. Thus, the probability that the window W, which consists of clips $c_t, c_{t+1}, \ldots, c_{t+w-1}$, matches the query q is given by:

$$\begin{aligned} \Pr(W \models q) &= \Pr(X_w = F' | X_0 = q'_0) \\ &= u^T (P_t P_{t+1} \cdots P_{t+w-1}) v \end{aligned} \tag{1}$$

The time complexity of calculating this pattern matching probability is $\Theta(w \cdot |S|^2)$, where w is the window size and $|S|$ is the number of states in the NHMC, which is the same as the number of states in the DFA. Note that $|S|$ is determined solely by the number of actions specified in the query. While long pattern queries would have a larger $|S|$, in practice $|S|$ would typically fall within single to double-digit. Furthermore, longer pattern queries also means they are more selective. This effectively empowers the PPAT with stronger filtering power, which can reduce the number of oracle model accesses.

6.2 The Rejection Threshold

Given the F1 target specified in the query, we translate it to a rejection threshold t for PPAT. Since the F1 score combines both recall and precision, and our priority in PPAT is to filter out negative windows, we set the precision target P_{target} to 1 and focus on the recall target R_{target}. With $P_{\text{target}} = 1$, the recall target can be derived from the F1 target: $R_{\text{target}} = \frac{F1}{2-F1}$.

To achieve the recall target, we follow [58] to determine the rejection threshold. PP seeks to identify the largest threshold that meets the desired recall target, using data sampled from the queried video and labeled by the oracle model (or human labels, if available). Specifically, with a given threshold t, recall is calculated as the ratio of windows that both probabilistically match the query (based on threshold t) and truly match the query, to the total number of windows that match the query according to the oracle model. The objective is to search from a threshold value of 1 down to the threshold value t that can result in a recall just meeting the target R_{target}. Doing so allows windows with a pattern matching probability below t to be pruned, without filtering out positive windows. In other words, it strikes a balance between efficiency (pruning non-matching windows) and accuracy (meeting the recall target). We leave the use of more sophisticated techniques, such as the ones proposed in [42, 51], to derive thresholds with tighter quality bound as future work.

7 Adaptive Event Materialization

In Bobsled, windows that pass PPAT are sent to AEM for further processing. The goal of AEM is to dynamically decide the next clip to be materialized by the oracle model that can potentially lead to early rejection or early acceptance of a window. In other words, it aims to make a sequence of choices online that can result in a clip materialization order that incurs minimal cost. AEM is an optimization technique designed for efficiency without impacting accuracy. In cases where a window has no early stop opportunities (e.g., consider a query ('A').next('B').next('C') and a window

[ABC]), the materialization order is immaterial and AEM would materialize all the clips to uphold correctness. However, since pattern matching queries are generally selective, AEM is an effective technique that can bring substantial speedup.

AEM works by picking the next clip that maximizes the acceptance probability if the window eventually matches the query, or pick the one that maximizes the rejection probability if the window eventually does not match. The challenge is that it is unknown in advance whether the current window will ultimately match the query or not. To address this uncertainty, AEM employs information entropy to unify the decision-making process across these two possible outcomes.

Let Z be a random variable, representing whether a window W matches the query q. Z follows a Bernoulli distribution: $Z = 1$ if W matches q and $Z = 0$ otherwise. The uncertainty of Z can be measured by its entropy $\mathbb{H}(Z)$, calculated from the probability that W matches q, denoted as $\Pr(W \models q)$. Initially, when it is uncertain whether W matches q, the entropy of Z is greater than 0. Let ψ be a random variable, representing the materialized events within window W. As more clips are materialized, the conditional entropy $\mathbb{H}(Z|\psi)$ decreases and eventually reaches 0. Thus, minimizing the conditional entropy is equivalent to maximizing the probability of acceptance or rejection.

Given this unified form, the objective of AEM is to minimize the conditional entropy to zero with the minimal cost. However, since the events are unknown in advance, AEM selects the clips **sequentially**, frugally materializing them one at a time. Furthermore, to minimize the number of materialized clips, AEM aims to minimize $\mathbb{H}(Z|\psi)$ as fast as possible. This is equivalent to **maximizing the mutual information** between Z and ψ, which measures the amount of information that Z contains about ψ. According to information theory, we have $\mathbb{I}(Z;\psi) = \mathbb{H}(Z) - \mathbb{H}(Z|\psi)$, where $\mathbb{I}(Z;\psi)$ denotes the mutual information between Z and ψ. Since the entropy of Z is a constant, maximizing $\mathbb{I}(Z;\psi)$ would lead to minimizing $\mathbb{H}(Z|\psi)$. Therefore, the problem is essentially a *Sequential Information Maximization* (SIM) problem [13].

In SIM, we have random variables Z, Y_1, Y_2, …, Y_m, where Z depends deterministically on Y_1, Y_2, …, Y_m via a function $\mathcal{D}$, i.e., $Z \coloneqq \mathcal{D}(Y_1, Y_2, \ldots, Y_m)$. The random variables Y_1, Y_2, …, Y_m are observable, while Z is not directly observable. To gain the information about Z, a test can be performed on each Y_i, which will reveal its outcome y_i. Each test incurs a unit of cost. Given a fixed budget of b tests, the goal is to sequentially choose tests that are maximally informative about Z. This sequence of choices is encoded as a policy $\pi \coloneqq \{\pi_1, \pi_2, \ldots, \pi_b\}$, where π_i specifies the test at the i-th step. For example, at the first step, a test is performed on Y_{π_1} and the outcome y_{π_1} is observed. The outcomes collected through π are denoted as $\psi_\pi \coloneqq \{(Y_{\pi_1}, y_{\pi_1}), (Y_{\pi_2}, y_{\pi_2}), \ldots, (Y_{\pi_k}, y_{\pi_k}))\}$.

The optimal policy π^* maximizes mutual information between Z and ψ_π: $\pi^* = \operatorname{argmax}_\pi \mathbb{I}(Z;\psi_\pi)$. Finding the optimal policy π^*, however, is an NP-hard problem [47]. Instead, a greedy policy is widely used, which chooses the test with the maximal expected conditional mutual information at each step: $\pi_{h+1} = \operatorname{argmax}_{i \in I} \mathbb{E}_{Y_i}[\mathbb{I}(Z;\psi_\pi|Y_i)]$, where $I = \{1, 2, \ldots, m\}$ and ψ_π refers to the materialized events in the previous h steps. This approach is simple and has been proved to be near-optimal [13, 45]. In fact, SIM bears resemblance to multi-arm bandit [62] and reinforcement learning (RL), where the reward can be defined as the reduction of uncertainty. However, in RL, the environment and the inherent uncertainties are unknown, necessitating that RL solutions explore and exploit the environment. In contrast, in SIM, the probability distribution of each Y_i is already known through the proxy scores. As a result, SIM solutions can focus on leveraging this known information to yield near-optimal results, without the need for extensive exploration.

Now, consider a query q and a window W of w clips $[c_t, c_{t+1}, \ldots, c_{t+w}]$. We define random variables Z, Y_t, Y_{t+1}, …, Y_{t+w} where Z represents whether W matches the query q, and Y_i represents the possible events at clip c_i. By adopting the greedy policy of the SIM problem (with a budget of w tests, since a window has a maximum of w clips to materialize), we materialize the clip with the

maximal expected conditional mutual information at each step. Note that the conditional mutual information is calculated by $\mathbb{I}(Z;\psi_\pi|Y_i) = \mathbb{H}(Z|\psi_\pi) - \mathbb{H}(Z|\psi_\pi, Y_i)$. Since $\mathbb{H}(Z|\psi_\pi)$ is constant for all clips, maximizing the expected conditional mutual information $\mathbb{I}(Z;\psi_\pi|Y_i)$ becomes a dual of minimizing the an expected conditional entropy $\mathbb{H}(Z|\psi_\pi, Y_i)$. Consequently, for the $(h+1)$-th step, we exclude the ones that have already been materialized and pick:

$$\pi_{h+1} = \underset{i \in W \setminus \psi_\pi}{\operatorname{argmin}} \mathbb{E}_{Y_i}[\mathbb{H}(Z|\psi_\pi, Y_i)] \tag{2}$$

7.1 Efficient Computation of Expected Conditional Entropy

Equation 2 requires computing the expected conditional entropy, i.e., $\mathbb{E}_{Y_i}[\mathbb{H}(Z|\psi_\pi, Y_i)]$, $O(w)$ times. Each expected conditional entropy is calculated by:

$$\mathbb{E}_{Y_i}[\mathbb{H}(Z|\psi_\pi, Y_i)] = \sum_{y_i} \Pr(Y_i = y_i)\mathbb{H}(Z|\psi_\pi, Y_i = y_i) \tag{3}$$

The conditional entropy $\mathbb{H}(Z|\psi_\pi, Y_i = y_i)$ is calculated by:

$$-\sum_{z \in \{0,1\}} \Pr(Z = z|\psi_\pi, Y_i = y_i) \log \Pr(Z = z|\psi_\pi, Y_i = y_i) \tag{4}$$

where Z follows the Bernoulli distribution:

$$\Pr(Z = 1|\psi_\pi, Y_i = y_i) = \Pr(W \models q|\psi_\pi, Y_i = y_i) \tag{5}$$

$$\Pr(Z = 0|\psi_\pi, Y_i = y_i) = 1 - \Pr(W \models q|\psi_\pi, Y_i = y_i) \tag{6}$$

According to Equation 1, the pattern matching probability is calculated by constructing w transition matrices from a window with w clips. Therefore, to compute the pattern matching probability conditioned on ψ_π and $Y_i = y_i$, we have:

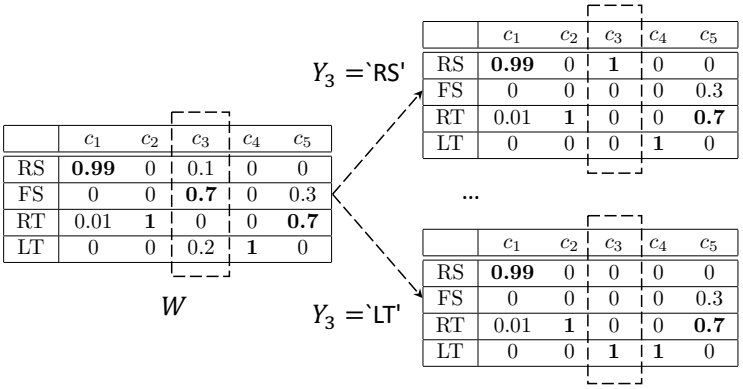
$$\Pr(W \models q|\psi_\pi, Y_i = y_i) = (u^T \prod_{j=t}^{i-1} P'_j) \cdot P'_i \cdot (\prod_{k=i+1}^{t+w-1} P'_k v) \tag{7}$$

where u and v are indicator vectors as defined in Equation 1. Here, P'_i is the transition matrix conditioned on $Y_i = y_i$, i.e., it is derived based on conditioning the current clip-of-interest c_i, assuming a label y_i. P'_j and P'_k are the transition matrices derived based on having some clips with certain labels (i.e., already after materialization) and some clips remain uncertain.

For example, consider the window W in Figure 5, $\pi = \{c_2, c_4\}$ has already been materialized. Among the uncertain ones, $W \setminus \psi_\pi = \{c_1, c_3, c_5\}$, Equation 2 aims to pick the one with the minimum expected conditional entropy. Computing the expected conditional entropy of, say, c_3, according to Equation 3, requires considering the entire event space of Y_3, i.e., {'RS', 'FS', 'RT', 'LT'}. P'_j in Equation 7 refers to the transition matrices constructed by c_1 and c_2. Similarly, P'_k in Equation 7 refers to the transition matrices constructed by c_4 and c_5. P'_i in Equation 7 refers to the transition matrix constructed by c_3, conditioning on Y_3 equals to 'RS', 'FS', 'RT', or 'LT'.

Computing Equation 3 requires enumerating all possible events for a specific clip. Equation 7 has a complexity of $\Theta(w|S|^2)$, where $|S|$ is dimension of square transition matrix. Denote the size of the event space as $|\mathcal{Y}|$, the complexity of calculating Equation 3 is $\Theta(|\mathcal{Y}|w|S|^2)$. Consequently, a naive implementation of Equation 2 requires $O(|\mathcal{Y}|w^2|S|^2)$.

While $|S|$ and $|\mathcal{Y}|$ are generally very small, w can be a large number. This suggests that, with a naive approach, the computation of expected conditional entropy could become the bottleneck of the system due to its high computational demands. To address this issue, we compute the products $u^T \prod_{j=t}^{i-1} P'_j$ and $\prod_{k=i+1}^{t+w-1} P'_k v$ only once per window in Equation 7 and share them among different values of Y_i.



	c_1	c_2	c_3	c_4	c_5
RS	**0.99**	0	0.1	0	0
FS	0	0	**0.7**	0	0.3
RT	0.01	**1**	0	0	**0.7**
LT	0	0	0.2	**1**	0

	c_1	c_2	c_3	c_4	c_5
RS	**0.99**	0	**1**	0	0
FS	0	0	0	0	0.3
RT	0.01	**1**	0	0	**0.7**
LT	0	0	0	**1**	0

	c_1	c_2	c_3	c_4	c_5
RS	**0.99**	0	0	0	0
FS	0	0	0	0	0.3
RT	0.01	**1**	0	0	**0.7**
LT	0	0	**1**	**1**	0

Fig. 5. Computing $\mathbb{E}_{Y_3}[\mathbb{H}(Z|\psi_\pi, Y_3)]$

Theorem 1. *By sharing the products $u^T \prod_{j=t}^{i-1} P'_j$ and $\prod_{k=i+1}^{t+w-1} P'_k v$ among different values of Y_i, the complexity of computing Equation 2 is reduced to $O(w|\mathcal{Y}||S|^2)$, which is linear to the window size.*

Proof. The products $u^T \prod_{j=t}^{i-1} P'_j$ and $\prod_{k=i+1}^{t+w-1} P'_k v$ in Eq 7 are independent of Y_i. Computing them once per window amortizes the complexity of Eq 7 to $\Theta(|S|^2)$. Since Eq 3 involves computing Eq 7 for each of the $|\mathcal{Y}|$ events, its total complexity is $\Theta(|\mathcal{Y}||S|^2)$. Finally, computing Equation 2 requires evaluating Eq 3 for $O(w)$ positions, resulting in a total complexity of $O(w|\mathcal{Y}||S|^2)$. □

7.2 Multi-Window Optimization

The greedy policy materializes clips within a window in a near-optimal order that can maximize the chance of early accept/reject it. However, the near-optimality is only specific to the current window-of-interest. In this section, we discuss a heuristic that aims to maximize the chance of early accept/reject of both the current window W **and** the windows that overlap with it. For example, consider the following query:

```
('rolling stop').minDuration(2) //Example Query EQ3
.next('right turn').minDuration(2)
.window(5)
.f1(0.99)
.model(mvit-v2-s, c3d-3)
```

In this query, the window size is 5 clips; the first event in the pattern requires a 'rolling stop' (RS) and the second event requires a 'right turn' (RT), both events required to appear in at least two clips to form a match.

Now consider the video stream in Figure 6, which has 10 clips (c_1 to c_{10}) and their oracle labels shown. None of the first five overlapping windows W_1 to W_5 can find a match on the query pattern (e.g., the first clip in W_4 is, c_4, a 'left turn'; hence, W_4 can be rejected once knowing the label of c_4). When considering window W_1 alone, materializing clip c_2 first or c_4 first can also early reject W_1 at the same cost of inferencing one clip. However, materializing c_4 can actually conveniently reject windows W_2, W_3, and W_4 as well. In contrast, materializing c_2 can only early reject W_2 in addition to W_1. In fact, the materialization choice of the current window W can influence all future windows, including the ones that do not overlap with W. This is because a window that overlaps with W would in turn overlap with some other windows that are even more further apart (e.g., W_1 overlaps W_5, W_5 overlaps W_9).

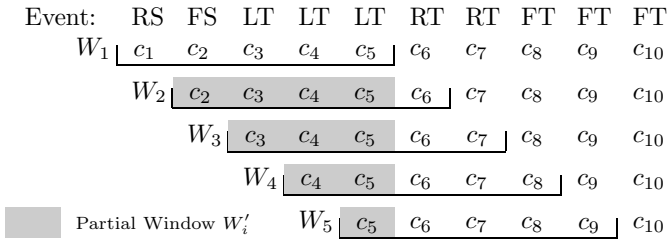


Fig. 6. Multi-Window Optimization

More specifically, regard W_1 is the current window of interest and AEM is considering which clip in it (c_1 to c_5) should be materialized next. The near-optimal choice can be identified by computing Equation 2. That choice is near-optimal with respect to the current window W_1 only. The next window of interest, W_2, indeed contains four clips that overlap with W_1 (they are c_2 to c_5). The near-optimal choice with respect to W_1, say, c_3, may not necessarily be the near-optimal choice with respect to W_2. In light of that, when deciding which clip to materialize when processing W_1, we hope to consider the benefit to not only W_1, but also to all future windows that could potentially benefit from our current decision, i.e., a near-optimal choice across the whole video stream.

To account for this, we enhance the original greedy policy by also considering the benefits to future windows when selecting the next clip to materialize. However, in a streaming context, future events (e.g., events of clips $c_6, c_7, \ldots$) are not visible at the time of decision-making. Therefore, we only consider the best-effort and account for the potential impact on "partial windows", which are sub-windows that overlap with the current window but include only the events that are currently visible. For example, in Figure 6, when processing window W_1, the partial windows of interest are W_2', W_3', W_4', and W_5'. The decision of which clip to materialize next would then consider the potential benefits to not only W_1, but also to W_2' through W_5'.

To consider the partial windows in our decision-making process, we can derive their corresponding entropy Z_i', where each Z_i' indicates whether the partial window W_i' matches the query q. However, we do not need to compute each Z_i' individually. Instead, we can rewrite the query, which allows the pattern in q to start matching at *any* position within W_i. For example, for Query EQ3, we can rewrite it as:

```
('true').minDuration(0) //Example Query EQ3*
.next('rolling stop').minDuration(2)
.next('right turn').minDuration(2)
.window(5)
.f1(0.99)
.model(mvit-v2-s, c3d-3)
```

The rewritten query essentially captures the semantic of matching against the original window and all its corresponding partial windows.

8 Discussion, Implementation, and Optimizations

Bobsled is implemented in both C++ and Python. We use Decord [24] for video decoding, PyTorch for model training and inference, and NumPy for matrix computation.

Cross-window Transition Matrix Sharing. The computation of pattern matching probability is achieved by constructing a Non-Homogeneous Markov Chain (NHMC). For a window of size w, W transition matrices are constructed to form an NHMC. The construction of these transition

Dataset	#action labels	#frames	Oracle's mAP	Proxy's mAP
BDD100K [17]	5	232k	91.2	22.8
MERL [78]	5	439k	78.6	51.4
ModelCar [2]	10	477k	89.2	53.6
Cooking [3]	10	296k	96.1	80.6
DailyLife [4]	10	332k	98.9	86.6
Baseball [5]	15	340k	82.2	63.5
Mixed	30	1105k	92.4	66.5

Table 1. Dataset and Model Accuracy.

matrices becomes computationally expensive when the window size is large. However, since the sliding windows overlap, Bobsled caches and reuses a transition matrix whenever possible. The idea here is similar to the one in subsection 7.1, but applying it across windows. This approach avoids the duplicate computation of transition matrices for different windows, thereby reducing overhead. Existing systems like Scotty [84] provide efficient sharing implementation for data streams. Unfortunately, their solutions are known to be not beneficial to count-based window with a slide step of one [84], which is our case.

Batch Inference. In AEM, the greedy policy deliberately selects only one clip to process at each step, which avoids over-materializing any unnecessary ones. However, that would in turn under-utilize the GPU. In Bobsled, we used batch inference. At each inference, we batch the clips selected by AEM from concurrent streams video and process them together by the GPU. Clips required by multiple queries on the same video streams would be shared.

Cold Start and Data Drift. Bobsled follows [58] and begins with PPAT initially disabled. It gathers proxy scores and oracle labels from the video streams being processed and then enables PPAT after a sufficient number of labels are acquired. During the cold start period, the system operates using AEM only. If there is any stringent latency constraint, we can allocate additional computational resources if needed [28, 86]. Whenever a significant shift in data distribution [30] is detected, Bobsled will recompute the threshold to adapt to the new distribution.

Limitations. As with most streaming systems [1, 89, 97], Bobsled requires more resources and computations when the size of the query windows increases. Nonetheless, in video stream analytics, we have observed that interesting patterns (e.g., 'right-turn-after-rolling-stop') often span short windows in the order of seconds to minutes, instead of long ones in the order of hours. Additionally, action models are still under active research and sometimes may not reach satisfactory quality. However, better models are emerging quickly, and Bobsled can consistently yield considerable speedup regardless of model development.

9 Evaluation

We conduct evaluations of Bobsled on seven real datasets. The experiments were conducted on an Intel Xeon W-2123 server with 80 GB RAM and one NVIDIA GeForce RTX 3090 GPU.

Dataset. Table 1(left) shows the details of the seven real datasets. The BDD100K dataset [96] is a collection of dash-cam footage. In [17], the authors have manually labeled 200 videos with five

[2]https://www.youtube.com/@A4Garage
[3]https://www.youtube.com/@MrT2022
[4]https://www.youtube.com/@LouCslife
[5]https://www.youtube.com/@MLB

ID	Data	Query	Description	Sel %
Q1	BDD100K	`('stop').minDuration(5)` `.followedBy('right')`	The car stops for at least 5 clips and then turns right after a while.	10.3%
Q2	BDD100K	`('left').followedBy('right')`	The car turns left and then turns right after a while.	3.1%

Q3	MERL	`('reach to shelf')`
		`.next('hand in shelf').minDuration(10)`
		`.next('retract from shelf')`
			The customer reaches to the shelf, places a hand in the shelf for at least 10 clips, and then retracts the hand.	4.7%

Q4	MERL	`('inspect shelf')`
		`.next('reach to shelf')`
		`.next('hand in shelf').minDuration(5)`
		`.next('retract from shelf')`
		`.followedBy('inspect product')`
			The customer inspects the shelf, reaches to the shelf, places a hand in the shelf for minimum of 5 clips, retracts the hand, and then follows by inspecting the product.	5.2%

Q5	ModelCar	`('unboxing')`
		`.next('inspecting car body')`
		`.next('removing chrome coating')`
			The model car builder unboxes the car parts, inspects the body of the car, and then removes chrome coating from a part.	0.3%

Q6	ModelCar	`('engraving').minDuration(5)`
		`.maxDuration(15)`
		`.next('test fitting')`
			The model car artist spends between 10 to 15 clips engraving a part, then proceeds to test its fitting with the model.	0.5%

Q7	Cooking	`('food plating').minDuration(15)` `.next('serving')`	The chef puts food on the plate for at least 15 clips and then serves it to the customer.	7.2%

Q8	Cooking	`('sauteing')`
		`.followedBy('cutting vegetables')`
		`.minDuration(10)`
		`.next('food plating')`
			The chef sautes the ingredients, followed by cutting more vegetables for at least 10 clips, and finally plating the food on plate.	8.1%

Q9	DailyLife	`('cooking')` `.next('cutting ingredients')`	The person is cooking and then cuts ingredients.	1.2%
Q10	DailyLife	`('assembling equipments')` `.followedBy('boxing or unboxing')`	The person is assembling some equipment, followed by opening or closing some boxes.	3.2%

Q11	Baseball	`('player walking')`
		`.next('in-stand reactions')`
		`.next('throwing')`
			The player walks, reacts in the stand, and then throws the ball.	1.8%

Q12	Baseball	`('showcasing player or umpire')`
		`.minDuration(5)`
		`.next('throwing')`
		`.next('getting ready for a swing')`
			The video showcases a player or umpire for at least 5 clips, then the player throws the ball, and then gets ready for a swing.	1.4%

Q13	Mixed	`('cutting vegetables')`
		`.next('food plating').next('serving')`
		`.next('sauteing')`
			The chef cuts vegetables, plates the food, serves it, and then sautes more ingredients.	1.5%

Q14	Mixed	`('engraving').next('cutting plastic')`
		`.next('test fitting')`
		`.followedBy('applying glue')`
			The model car artist engraves a part, cuts plastic, tests the fitting, followed by applying glue to assemble the parts.	0.7%

Table 2. Benchmark Queries

distinct action labels for that dataset, and we have concatenated those 200 videos into a single video stream. The MERL [78] dataset is a collection of 106 videos recorded using a fixed overhead camera, capturing people shopping in a grocery store setting. Each video is approximately 2 minutes long, and we have concatenated them into a single video stream. This dataset also has built-in action labels. Additionally, we have collected four more video streams from YouTube: ModelCar, Cooking, DailyLife, and Baseball. We manually labeled 10 actions for each of the first three datasets and 15 actions for the last dataset. Furthermore, we follow common practices in the computer vision field [36, 38, 46, 79, 95] to create a Mixed dataset by concatenating videos from the ModelCar, Cooking, and DailyLife datasets so as to obtain a video with 30 different actions.

Model. In the experiment, we used the pre-trained MViTv2-S model [54] as the oracle and the pre-trained C3D-3 [82] as the proxy. The MViTv2-S model consists of 16 attention layers, whereas the C3D-3 model is composed of 3 convolutional layers. Table 1(right) lists the models' accuracy, measured by mean Average Precision (mAP), on the videos based on the ground-truth human labels. mAP is a standard metric for evaluating the accuracy of action recognition models [54, 92]. The oracle model runs at 0.8k fps, while the proxy model operates at 10.9k fps.
Query. Table 2 lists the queries used in the evaluation. The queries were constructed to ensure meaningful results could be returned. Each dataset has two benchmark queries, and the pattern length of these queries varies from two to five events. The table also includes the selectivity of the queries.
Baseline. To the best of our knowledge, Bobsled is the first system capable of supporting efficient complex event processing on videos and it does not have any direct competitors. We have constructed four baselines to evaluate our system against:

- `Localize-all-by-Oracle` (All-Oracle). In this baseline, we use the oracle model to localize all events and then apply Flink's CEP engine to match the pattern.
- `Localize-all-by-Proxy` (All-Proxy). Same as the above, except using the proxy in place of the oracle model.
- `Localize-interested-by-Zeus` (Zeus). Here, we use the state-of-the-art action localization system, Zeus [17], to first localize only the events specified by the query, and then use Flink's CEP engine to match the query pattern. Zeus can also accept as input a target F1-score to balance between the speedup and accuracy.
- `Query-specific Model-based Filtering` (QMF). Lastly, we consider an AI-native approach that trains a query-specific model for each query. The model utilizes a frozen backbone based on the pre-trained C3D-3, with two additional layers added for fine-tuning. The training data for fine-tuning consists of human labels indicating whether a specific window matches the query pattern. During inference, this QMF method feeds a window of frames into the model to obtain the probability that the window matches the specific query pattern. Windows with probabilities below a rejection threshold are discarded. The rejection threshold is calibrated using the same methodology as in PPAT (Section 6.2). All frames in the remaining windows are further processed by the oracle model, and matches are identified using Flink's CEP engine.

Both Zeus and QMF require offline training for their query-specific models. We follow the ratio suggested in [78, 96] to split the data for training, validation, and testing for the BDD100K and MERL datasets. For the YouTube data, we evenly split the data into training, validation, and test sets.
Evaluation Metric. We report the system's performance in terms of throughput (frames processed per second) and accuracy (F1-score) using the test set from each dataset. To minimize interference, we assess throughput based on running a single query over a single video stream. The F1-score is measured using the approach for action localization [17]. For each match m_i of query pattern reported by Bobsled in the form of a time interval $[s_i, e_i]$, we calculate the Intersection over Union (IoU) score between it and every human-labeled match from the ground truth. If the pair with the maximum IoU exceeds 0.5, we consider m_i a true positive; otherwise, it is classified as a false positive. False negative is measured similarly.
Default configuration. By default, we regard 16 frames as one clip, which is approximately half a second. We set the batch size to 8 to fully utilize the GPU. The default window size and F1 target for our queries are 60 clips and 0.99, respectively.

9.1 End-to-end Comparison

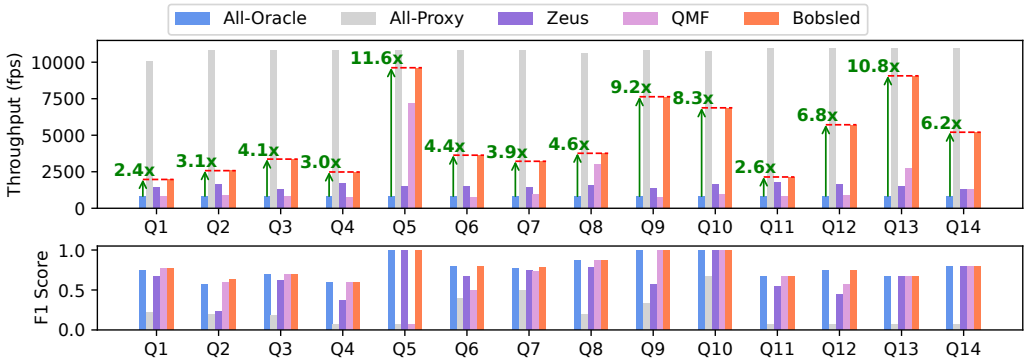


Fig. 7. End-to-end comparison

Figure 7 presents the performance of Bobsled and the others in terms of throughput and accuracy.

All-Oracle generally achieves the best accuracy, but its throughput is the lowest. In contrast, All-Proxy consistently has the highest throughput but the worst accuracy. This aligns with major findings [40, 41, 51, 58], which indicate that any solution relying solely on the proxy model exhibits unacceptable accuracy. Since All-Proxy has an unacceptable accuracy, we therefore exclude it from further discussion from now on.

QMF as a query-specific solution has good accuracy for many queries. However, it also has a noticeable accuracy loss in others (Q6, Q7, Q12). Furthermore, its accuracy is simply unacceptable on query Q5. Despite all these, QMF generally shows no significant performance improvement over All-Oracle except on queries like Q5 which has large accuracy loss. The reason for this is that the query-specific model struggles to differentiate between the window that matches the query and the one that does not because of its lightweight architecture and the limited number of positive samples (matches) in the training set. Therefore, building a query-specific model to serve as an early filter does not appear to be an effective approach.

Zeus has certain throughput improvement over All-Oracle, but it also has a noticeable accuracy loss in many cases. This arises due to a fundamental difference between this paper's target and Zeus's target. Specifically, Zeus targets high F1-score for individual action events, whereas we target high F1-score for pattern matching in this paper.

Bobsled consistently maintains high accuracy, close to All-Oracle across all queries. In fact, there are instances (e.g., Q1, Q2) where Bobsled, or some other methods, may achieve slightly better accuracy than All-Oracle, as the oracle model itself is not perfect (Table 1) and can produce false positives. In such cases, Bobsled is able to filter these out early via PPAT, thereby achieving higher accuracy than All-Oracle. In terms of performance, Bobsled achieves 2.4×–11.6× higher throughput than All-Oracle, and a 1.2×–10× increase in throughput compared to QMF. When compared to Zeus, Bobsled achieves 1.2×–6.4× higher throughput.

9.2 Ablation Study

We next carry out an ablation study to examine the effectiveness of the individual techniques within Bobsled. Figure 8 reports the average throughput and average F1-score across all queries. As a reference, we include (i) the result of All-Oracle in the figure. We then report the result of Bobsled (ii) with only PPAT enabled (PPAT), i.e., with AEM disabled; (iii) with only AEM enabled,

but AEM is done through the naive approach discussed in Sec 7.1 (AEM-O); (iv) with only AEM enabled, but AEM is done through the efficient method discussed in Sec 7.1 (AEM); and (v) with both PPAT and AEM with efficient computation enabled (PPAT+AEM).

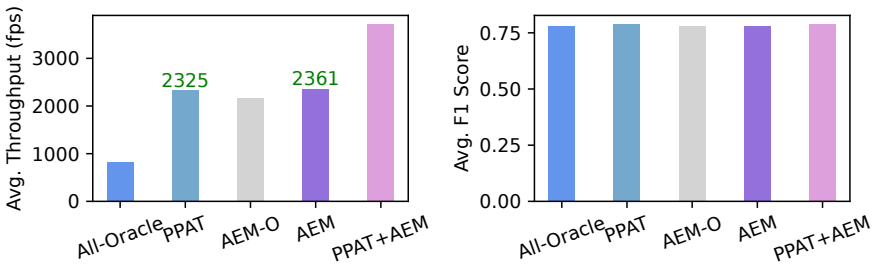


Fig. 8. Ablation study

From the figure, we can observe that both PPAT and AEM demonstrate their individual effectiveness, each contributing considerable throughput improvements without compromising result quality. Furthermore, comparing AEM-O with AEM, the efficient computation method discussed in Section 7.1 demonstrates noticeable effectiveness in optimizing the calculations of Equations 2 and 3.

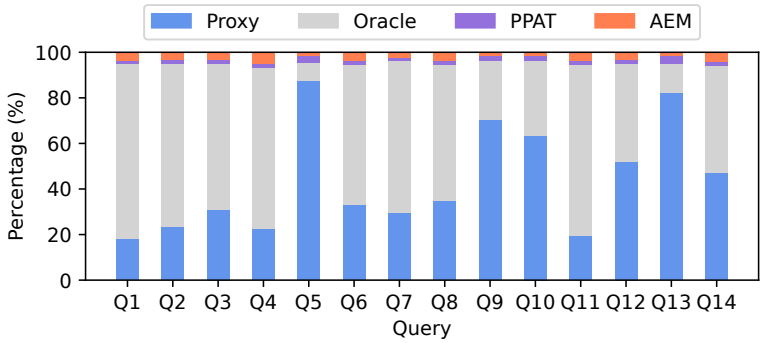


Fig. 9. Time breakdown

The throughput improvement brought by PPAT and AEM do not come for free. Figure 9 shows the time breakdown of each query within Bobsled, detailing the time spent on the proxy model, oracle model, PPAT and AEM. It reveals that the computation overheads of PPAT and AEM are noticeable. However, their presence is very worthwhile, as they together consume only a minor fraction of the overall processing time but significantly increase throughput as shown in Figure 8.

9.3 Impact of F1 Target

Figure 10 shows the average throughput and average accuracy of Bobsled across all queries with varying F1 accuracy targets. As expected, Bobsled's accuracy increases with more stringent requirements, at the cost of lower throughput. Most importantly, when the F1 target is above 0.9 (the preference of most users), Bobsled maintains high and stable accuracy and performance gains.

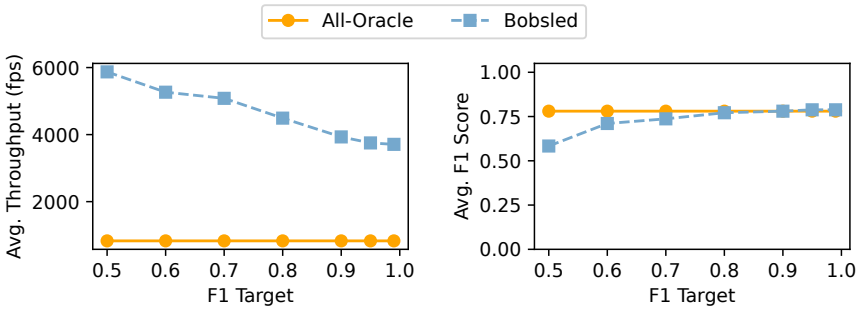


Fig. 10. Impact of F1 target

9.4 Impact of Pattern Length

Figure 11 shows the throughput and accuracy of Bobsled when we vary the pattern length of Q4. We chose Q4 because it is the query with the longest pattern of length five in our query set such that we can control the pattern length by removing the queried event(s) in it.

From the figure, we can see that the throughput of Bobsled increases with the pattern length, while maintaining an accuracy almost identical to All-Oracle. As the pattern length of a query increases, it becomes more selective, giving Bobsled's PPAT and AEM more optimization opportunities, thus improving the throughput.

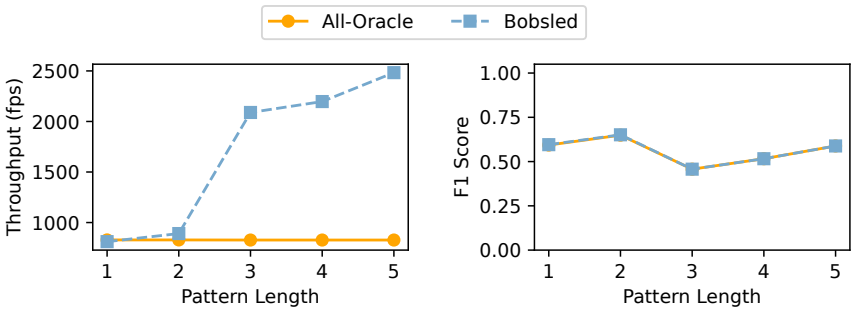


Fig. 11. Impact of pattern length

9.5 Impact of Window Size

Figure 12 shows the average throughput and average accuracy of Bobsled when we vary the queried window size of all the queries from 30 clips to 300 clips. As expected, the throughput decreases when we use a larger window size because a larger window has more clips needed to be processed. Nonetheless, we can see that Bobsled remains effective as its throughput drops less than 55% when the window size is increased 10× from 30 clips to 300 clips.

In addition, we present the results of Bobsled without the efficient computation described in Sec 7.1 (Bobsled-AEM-O), which allows for a clear comparison to observe the significant difference as the window size increases. The efficient computation in AEM offers a notable advantage by reducing the time complexity by a factor of w, where w is the window size. Consequently, the

benefits of this optimization become more pronounced with larger window sizes, as illustrated in the figure. Specifically, when the window size reaches 300, Bobsled-AEM-O exhibits almost no speedup due to the substantial overhead introduced by AEM. In contrast, Bobsled continues to deliver an average speedup of 2.3×.

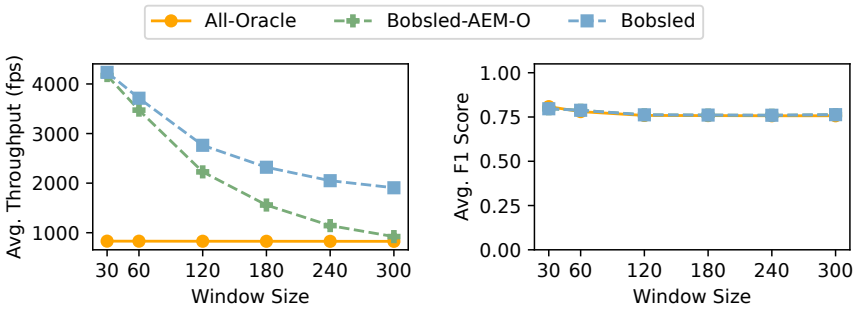


Fig. 12. Impact of window size

9.6 Impact of Filtering Method

Figure 13 illustrates the average throughput and average accuracy of Bobsled across all queries, evaluated under different filtering methods. In addition to the `All-Oracle` baseline, we include three filtering methods in the experiments. All methods discard windows with probabilities below a rejection threshold, calibrated using the same methodology as in PPAT (Section 6.2). However, they differ in how they derive the pattern matching probability: (i) `QMF`, a learned method from the baseline, uses a query-specific model to directly predict the pattern matching probability of a window; (ii) `Heuristic`, which approximates the pattern matching probability by assuming independence and computing the probability of all events in query q occurring: $\Pr(W \models q) \approx \Pr(\bigcap_{e \in E(q)} \{e \in W\})$. Here, $E(q)$ is the set of unique events in query q, and $e \in W$ indicates the presence of event e in window W. This heuristic approach, however, ignores almost all CEP semantics, such as event ordering; (iii) `PPAT`, the filtering method used by Bobsled.

The results demonstrate that PPAT achieves a greater speedup than the other filtering methods while maintaining accuracy. In contrast, the speedups provided by QMF and Heuristic come at the cost of noticeable accuracy loss.

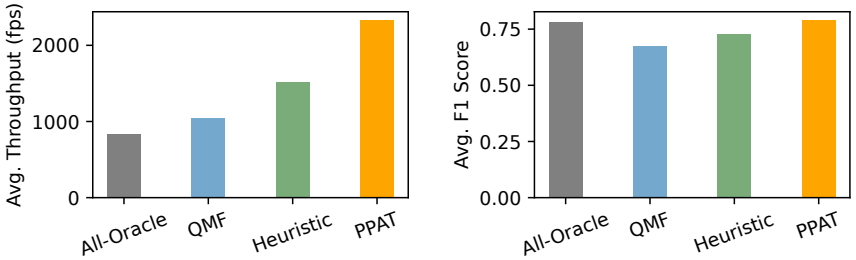


Fig. 13. Impact of filtering method

9.7 Impact of Materialization Policy

Figure 14 shows the average throughput and average accuracy of Bobsled across all queries with respect to different adaptive event materialization (AEM) policies. In the experiment, we have tested: (i) `Sequential`, which processes each clip within a window sequentially from the beginning; (ii) `Random`, which randomly picks a clip within a window every time; (iii) `Interval`, which selects the ik-th clip at each step i, where k is the pattern length. This ensures that if the materialization of the ik-th clip does not match any event specified in the query pattern, it can potentially reject the window-of-interest with minimal cost and advance the maximum number of windows. (iv) `Greedy`, which is Bobsled's greedy policy without multi-window optimization (MWO; Section 7.2); (v) `Greedy+MWO`, which is Bobsled's greedy policy with MWO.

From the figure, we can see that the greedy policy is more effective than the other basic policies, and the multi-window optimization can further improve the pruning power. Note that the materialization policy does not influence accuracy. It is because AEM only alters the order in which events are materialized within a window, as discussed in section 7. If a window does not match a pattern, Bobsled will discard it only after confirming it as a definite negative by materializing all necessary events. Similarly, if a window matches a pattern, Bobsled will return it only after confirming it as a definite positive. In other words, AEM will not yield any false positives or false negatives.

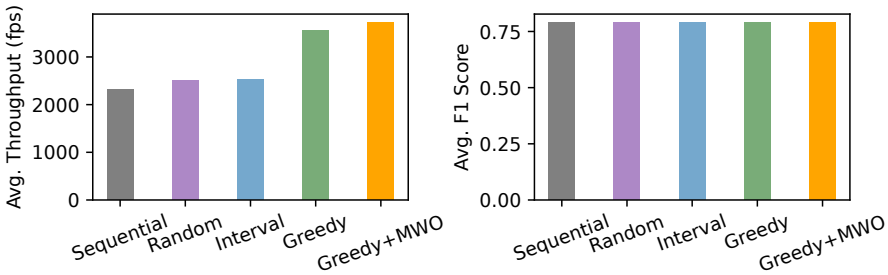


Fig. 14. Impact of materialization policy

10 Conclusions

In this paper, we present Bobsled, a novel video stream processing system designed to efficiently support complex event queries. Bobsled employs probabilistic pattern matching and adaptive event materialization to avoid processing unnecessary video clips. Experimental results demonstrate that Bobsled can achieve a significant improvement in processing rate without any noticeable loss in accuracy. Our future work includes the use of more sophisticated techniques, such as the one in [42], to derive rejection thresholds with tighter quality bound, and the use of the alternate approach like [5, 44] to obtain proxy scores.

Acknowledgments

This work is partially supported by Hong Kong General Research Fund (14208023), Hong Kong AoE/P-404/18, and the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under InnoHK supported by the Innovation and Technology Commission.

References

[1] Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman. 2008. Efficient pattern matching over event streams. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, Vancouver Canada, 147–160. doi:10.1145/1376616.1376634

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215. Santiago, 487–499. https://www.it.uu.se/edu/course/homepage/infoutv/ht08/vldb94_rj.pdf

[3] M. H. Ali, C. Gerea, B. S. Raman, B. Sezgin, T. Tarnavski, T. Verona, P. Wang, P. Zabback, A. Ananthanarayan, A. Kirilov, M. Lu, A. Raizman, R. Krishnan, R. Schindlauer, T. Grabs, S. Bjeletich, B. Chandramouli, J. Goldstein, S. Bhat, Ying Li, V. Di Nicola, X. Wang, David Maier, S. Grell, O. Nano, and I. Santos. 2009. Microsoft CEP server and online behavioral targeting. *Proceedings of the VLDB Endowment* 2, 2 (Aug. 2009), 1558–1561. doi:10.14778/1687553.1687590

[4] Adar Amir, Ilya Kolchinsky, and Assaf Schuster. 2022. DLACEP: A Deep-Learning Based Framework for Approximate Complex Event Processing. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 340–354. doi:10.1145/3514221.3526136

[5] Jaeho Bang, Gaurav Tarlok Kakkar, Pramod Chunduri, Subrata Mitra, and Joy Arulraj. 2023. Seiden: Revisiting Query Processing in Video Database Systems. *Proceedings of the VLDB Endowment* 16, 9 (May 2023), 2289–2301. doi:10.14778/3598581.3598599

[6] Lars Brenna, Alan Demers, Johannes Gehrke, Mingsheng Hong, Joel Ossher, Biswanath Panda, Mirek Riedewald, Mohit Thatte, and Walker White. 2007. Cayuga: a high-performance event processing engine. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data (SIGMOD '07)*. Association for Computing Machinery, New York, NY, USA, 1100–1102. doi:10.1145/1247480.1247620

[7] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. 2024. Video generation models as world simulators. (2024). https://openai.com/research/video-generation-models-as-world-simulators

[8] Pierre Brémaud. 2020. Non-homogeneous Markov Chains. In *Markov Chains*. Vol. 31. Springer International Publishing, Cham, 399–422. doi:10.1007/978-3-030-45982-6_12 Series Title: Texts in Applied Mathematics.

[9] Marco Bucchi, Alejandro Grez, Andrés Quintana, Cristian Riveros, and Stijn Vansummeren. 2022. CORE: a complex event recognition engine. *Proceedings of the VLDB Endowment* 15, 9 (May 2022), 1951–1964. doi:10.14778/3538598.3538615

[10] Jiashen Cao, Karan Sarkar, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. 2022. FiGO: Fine-Grained Query Optimization in Video Analytics. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 559–572. doi:10.1145/3514221.3517857

[11] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2011. Apache Flink™: Stream and Batch Processing in a Single Engine. (May 2011).

[12] Daren Chao and Nick Koudas. 2024. Querying For Actions Over Videos. doi:10.48786/EDBT.2024.15

[13] Yuxin Chen, S. Hamed Hassani, Amin Karbasi, and Andreas Krause. 2015. Sequential Information Maximization: When is Greedy Near-optimal?. In *Proceedings of The 28th Conference on Learning Theory*. PMLR, 338–363. https://proceedings.mlr.press/v40/Chen15b.html ISSN: 1938-7228.

[14] Yueting Chen, Nick Koudas, Xiaohui Yu, and Ziqiang Yu. 2022. Spatial and temporal constrained ranked retrieval over videos. *Proceedings of the VLDB Endowment* 15, 11 (July 2022), 3226–3239. doi:10.14778/3551793.3551865

[15] Yueting Chen, Xiaohui Yu, and Nick Koudas. 2022. Ranked Window Query Retrieval over Video Repositories. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 2776–2791. doi:10.1109/ICDE53745.2022.00253 ISSN: 2375-026X.

[16] Yueting Chen, Xiaohui Yu, Nick Koudas, and Ziqiang Yu. 2021. Evaluating Temporal Queries Over Video Feeds. In *Proceedings of the 2021 International Conference on Management of Data*. ACM, Virtual Event China, 287–299. doi:10.1145/3448016.3452803

[17] Pramod Chunduri, Jaeho Bang, Yao Lu, and Joy Arulraj. 2022. Zeus: Efficiently Localizing Actions in Videos using Reinforcement Learning. In *Proceedings of the 2022 International Conference on Management of Data*. ACM, Philadelphia PA USA, 545–558. doi:10.1145/3514221.3526181

[18] Ticket Crushers. 2022. What is a Rolling Stop ("California Roll") or a Rolling Right Turn Red Light Camera Violation? https://www.ticketcrusherslaw.com/rolling-stop/

[19] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning. doi:10.48550/arXiv.2305.06500 arXiv:2305.06500 [cs].

[20] Maureen Daum, Enhao Zhang, Dong He, Magdalena Balazinska, Brandon Haynes, Ranjay Krishna, Apryle Craig, and Aaron Wirsing. 2022. VOCAL: Video Organization and Interactive Compositional AnaLytics. *12th Annual Conference on Innovative Data Systems Research (CIDR '22)* (Jan. 2022). https://par.nsf.gov/biblio/10378499-vocal-video-organization-interactive-compositional-analytics

[21] Yanlei Diao, Neil Immerman, and Daniel Gyllstrom. 2007. Sase+: An agile language for kleene closure over event streams. *UMass Technical Report* (2007). https://google-code-archive-downloads.storage.googleapis.com/v2/code.google.com/sase-umass/sase-07-tech-report.pdf

[22] Nihal Dindar, Peter M. Fischer, and Nesime Tatbul. 2011. DejaVu: a complex event processing system for pattern matching over live and historical data streams. In *Proceedings of the 5th ACM international conference on Distributed event-based system (DEBS '11)*. Association for Computing Machinery, New York, NY, USA, 399–400. doi:10.1145/2002259.2002330

[23] Dujian Ding, Sihem Amer-Yahia, and Laks Lakshmanan. 2022. On Efficient Approximate Queries over Machine Learning Models. *Proceedings of the VLDB Endowment* 16, 4 (Dec. 2022), 918–931. doi:10.14778/3574245.3574273

[24] DMLC. 2022. Decord: An efficient video loader for deep learning with smart shuffling that's super easy to digest. https://github.com/dmlc/decord

[25] Sebastian Ehmes, Lars Fritsche, and Konrad Altenhofen. 2020. GrapeL: Combining Graph Pattern Matching and Complex Event Processing. In *Systems Modelling and Management*, Önder Babur, Joachim Denil, and Birgit Vogel-Heuser (Eds.). Springer International Publishing, Cham, 180–196. doi:10.1007/978-3-030-58167-1_13

[26] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. 2021. Multiscale Vision Transformers. http://arxiv.org/abs/2104.11227 arXiv:2104.11227.

[27] Apache Flink. 2024. FlinkCEP - Complex event processing for Flink. https://nightlies.apache.org/flink/flink-docs-master/docs/libs/cep/

[28] Avrilia Floratou, Ashvin Agrawal, Bill Graham, Sriram Rao, and Karthik Ramasamy. 2017. Dhalion: self-regulating stream processing in heron. *Proceedings of the VLDB Endowment* 10, 12 (Aug. 2017), 1825–1836. doi:10.14778/3137765.3137786

[29] 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. doi:10.48550/arXiv.1910.02993 arXiv:1910.02993 [cs].

[30] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *Comput. Surveys* 46, 4 (April 2014), 1–37. doi:10.1145/2523813

[31] Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. (2024). doi:10.48550/ARXIV.2403.05530

[32] Lukasz Golab. 2006. Sliding Window Query Processing over Data Streams. (2006).

[33] Daniel Gyllstrom, Jagrati Agrawal, Yanlei Diao, and Neil Immerman. 2008. On Supporting Kleene Closure over Event Streams. In *2008 IEEE 24th International Conference on Data Engineering*. IEEE, Cancun, Mexico, 1391–1393. doi:10.1109/ICDE.2008.4497566

[34] Daniel Gyllstrom, Eugene Wu, Hee-Jin Chae, Yanlei Diao, Patrick Stahlberg, and Gordon Anderson. 2006. SASE: Complex Event Processing over Streams. doi:10.48550/arXiv.cs/0612128 arXiv:cs/0612128.

[35] Wenjia He, Ibrahim Sabek, Yuze Lou, and Michael Cafarella. 2024. Optimizing Video Selection LIMIT Queries with Commonsense Knowledge. *Proceedings of the VLDB Endowment* 17, 7 (March 2024), 1751–1764. doi:10.14778/3654621.3654639

[36] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. ActivityNet: A large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 961–970. doi:10.1109/CVPR.2015.7298698

[37] Hong Huang, Bo Zhao, Hao Zhao, Zhou Zhuang, Zhenxuan Wang, Xiaoming Yao, Xinggang Wang, Hai Jin, and Xiaoming Fu. 2018. A Cross-Platform Consumer Behavior Analysis of Large-Scale Mobile Shopping Data. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. ACM Press, Lyon, France, 1785–1794. doi:10.1145/3178876.3186169

[38] Haroon Idrees, Amir R. Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. 2016. The THUMOS Challenge on Action Recognition for Videos "in the Wild". *CoRR* abs/1604.06182 (2016). arXiv:1604.06182 http://arxiv.org/abs/1604.06182

[39] Raghav Jain, Daivik Sojitra, Arkadeep Acharya, Sriparna Saha, Adam Jatowt, and Sandipan Dandapat. 2023. Do Language Models Have a Common Sense regarding Time? Revisiting Temporal Commonsense Reasoning in the Era of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 6750–6774. doi:10.18653/v1/2023.emnlp-main.418

[40] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics. (2018). doi:10.48550/ARXIV.1805.01046

[41] 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 (Aug. 2017), 1586–1597. doi:10.14778/3137628.3137664

[42] Daniel Kang, Edward Gan, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Approximate selection with guarantees using proxies. *Proceedings of the VLDB Endowment* 13, 12 (Aug. 2020), 1990–2003. doi:10.14778/3407790.3407804

[43] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, Yi Sun, and Matei Zaharia. 2021. Accelerating approximate aggregation queries with expensive predicates. *Proceedings of the VLDB Endowment* 14, 11 (July 2021), 2341–2354. doi:10.14778/3476249.3476285

[44] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2022. Semantic Indexes for Machine Learning-based Queries over Unstructured Data. http://arxiv.org/abs/2009.04540 arXiv:2009.04540 [cs].

[45] Bojan Karlaš, Peng Li, Renzhi Wu, Nezihe Merve Gürel, Xu Chu, Wentao Wu, and Ce Zhang. 2020. Nearest neighbor classifiers over incomplete information: from certain answers to certain predictions. *Proceedings of the VLDB Endowment* 14, 3 (Nov. 2020), 255–267. doi:10.14778/3430915.3430917

[46] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. 2017. The Kinetics Human Action Video Dataset. *CoRR* abs/1705.06950 (2017). arXiv:1705.06950 http://arxiv.org/abs/1705.06950

[47] Chun-Wa Ko, Jon Lee, and Maurice Queyranne. 1995. An Exact Algorithm for Maximum Entropy Sampling. *Operations Research* 43, 4 (1995), 684–691. https://www.jstor.org/stable/171694 Publisher: INFORMS.

[48] Ilya Kolchinsky and Assaf Schuster. 2018. Join query optimization techniques for complex event processing applications. *Proceedings of the VLDB Endowment* 11, 11 (July 2018), 1332–1345. doi:10.14778/3236187.3236189

[49] Ferdi Kossmann, Ziniu Wu, Eugenie Lai, Nesime Tatbul, Lei Cao, Tim Kraska, and Sam Madden. 2023. Extract-Transform-Load for Video Streams. *Proceedings of the VLDB Endowment* 16, 9 (May 2023), 2302–2315. doi:10.14778/3598581.3598600

[50] Nick Koudas, Raymond Li, and Ioannis Xarchakos. 2020. Video Monitoring Queries. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 1285–1296. doi:10.1109/ICDE48307.2020.00115 ISSN: 2375-026X.

[51] Ziliang Lai, Chenxia Han, Chris Liu, Pengfei Zhang, Eric Lo, and Ben Kao. 2020. Top-K Deep Video Analytics: A Probabilistic Approach. (2020). doi:10.48550/ARXIV.2003.00773

[52] Wangchao Le, Anastasios Kementsietsidis, Songyun Duan, and Feifei Li. 2012. Scalable Multi-query Optimization for SPARQL. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, Arlington, VA, USA, 666–677. doi:10.1109/ICDE.2012.37

[53] Yanwei Li, Chengyao Wang, and Jiaya Jia. 2023. LLaMA-VID: An Image is Worth 2 Tokens in Large Language Models. (2023). doi:10.48550/ARXIV.2311.17043

[54] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. 2022. MViTv2: Improved Multiscale Vision Transformers for Classification and Detection. http://arxiv.org/abs/2112.01526 arXiv:2112.01526.

[55] Pei-Sung Lin, Achilleas Kourtellis, Zhenyu Wang, and Rui Guo. 2015. *Understanding interactions between drivers and pedestrian features at signalized intersections.* Technical Report. Florida. Dept. of Transportation. https://rosap.ntl.bts.gov/view/dot/29582

[56] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. (2023). doi:10.48550/ARXIV.2304.08485

[57] Eric Lo, Ben Kao, Wai-Shing Ho, Sau Dan Lee, Chun Kit Chui, and David W. Cheung. 2008. OLAP on sequence data. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. 649–660.

[58] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating Machine Learning Inference with Probabilistic Predicates. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, Houston TX USA, 1493–1508. doi:10.1145/3183713.3183751

[59] David C Luckham and Brian Frasca. 1998. Complex Event Processing in Distributed Systems. (Aug. 1998).

[60] Yuan Mei and Samuel Madden. 2009. ZStream: a cost-based query processor for adaptively detecting composite events. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (SIGMOD '09)*. Association for Computing Machinery, New York, NY, USA, 193–206. doi:10.1145/1559845.1559867

[61] Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 2023. Large Language Models as General Pattern Machines. doi:10.48550/arXiv.2307.04721 arXiv:2307.04721 [cs].

[62] Oscar Moll, Favyen Bastani, Sam Madden, Mike Stonebraker, Vijay Gadepally, and Tim Kraska. 2022. ExSample: Efficient Searches on Video Repositories through Adaptive Sampling. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 2956–2968. doi:10.1109/ICDE53745.2022.00266 arXiv:2005.09141 [cs].

[63] R. Moore. 1987. *Possible-world semantics for autoepistemic logic.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 137–142.

[64] Mohammad Hossein Moshrefjavadi, Hossein Rezaie Dolatabadi, Mojtaba Nourbakhsh, Amir Poursaeedi, and Ahmadreza Asadollahi. 2012. An Analysis of Factors Affecting on Online Shopping Behavior of Consumers. *International Journal of Marketing Studies* 4, 5 (Sept. 2012), p81. doi:10.5539/ijms.v4n5p81

[65] OpenAI. 2023. GPT-4 Technical Report. (2023). doi:10.48550/ARXIV.2303.08774

[66] Stats Perform. 2024. SportVU - Football Performance Analytics. https://www.statsperform.com/team-performance/football-performance/sportvu/

[67] V. Poosala and V. Ganti. 1999. Fast approximate answers to aggregate queries on a data cube. In *Proceedings. Eleventh International Conference on Scientific and Statistical Database Management*. 24–33. doi:10.1109/SSDM.1999.787618

[68] Olga Poppe, Chuan Lei, Salah Ahmed, and Elke A. Rundensteiner. 2017. Complete Event Trend Detection in High-Rate Event Streams. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, Chicago Illinois USA, 109–124. doi:10.1145/3035918.3035947

[69] PyTorch. [n. d.]. Quantization — PyTorch 2.3 documentation. https://pytorch.org/docs/stable/quantization.html#frequently-asked-questions

[70] M. O. Rabin and D. Scott. 1959. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development* 3, 2 (April 1959), 114–125. doi:10.1147/rd.32.0114

[71] Christian M. Richard, John L. Campbell, James L. Brown, and Battelle Human Factors Transportation Center. 2006. *Task analysis of intersection driving scenarios : information processing bottlenecks*. Technical Report FHWA-HRT-06-033. https://rosap.ntl.bts.gov/view/dot/825

[72] Francisco Romero, Caleb Winston, Johann Hauswald, Matei Zaharia, and Christos Kozyrakis. 2023. Zelda: Video Analytics using Vision-Language Models. (2023). doi:10.48550/ARXIV.2305.03785

[73] Matthew Russo, Tatsunori Hashimoto, Daniel Kang, Yi Sun, and Matei Zaharia. 2023. Accelerating Aggregation Queries on Unstructured Streams of Data. *Proceedings of the VLDB Endowment* 16, 11 (July 2023), 2897–2910. doi:10.14778/3611479.3611496 arXiv:2308.09157 [cs].

[74] Christopher Ré, Julie Letchner, Magdalena Balazinksa, and Dan Suciu. 2008. Event queries on correlated probabilistic streams. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA, 715–728. doi:10.1145/1376616.1376688

[75] Reza Sadri, Carlo Zaniolo, Amir Zarkesh, and Jafar Adibi. 2001. Optimization of sequence queries in database systems. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '01)*. Association for Computing Machinery, New York, NY, USA, 71–81. doi:10.1145/375551.375563

[76] Nicholas Poul Schultz-Møller, Matteo Migliavacca, and Peter Pietzuch. 2009. Distributed complex event processing with query rewriting. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*. ACM, Nashville Tennessee, 1–12. doi:10.1145/1619258.1619264

[77] Long Sha, Patrick Lucey, Yisong Yue, Xinyu Wei, Jennifer Hobbs, Charlie Rohlf, and Sridha Sridharan. 2018. Interactive Sports Analytics: An Intelligent Interface for Utilizing Trajectories for Interactive Sports Play Retrieval and Analytics. *ACM Transactions on Computer-Human Interaction* 25, 2 (April 2018), 1–32. doi:10.1145/3185596

[78] Bharat Singh, Tim K. Marks, Michael Jones, Oncel Tuzel, and Ming Shao. 2016. A Multi-stream Bi-directional Recurrent Neural Network for Fine-Grained Action Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Las Vegas, NV, USA, 1961–1970. doi:10.1109/CVPR.2016.216

[79] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *CoRR* abs/1212.0402 (2012). arXiv:1212.0402 http://arxiv.org/abs/1212.0402

[80] Zhaochen Su, Juntao Li, Jun Zhang, Tong Zhu, Xiaoye Qu, Pan Zhou, Yan Bowen, Yu Cheng, and Min zhang. 2024. Living in the Moment: Can Large Language Models Grasp Co-Temporal Reasoning? doi:10.48550/arXiv.2406.09072 arXiv:2406.09072 [cs].

[81] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. 2022. VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. doi:10.48550/arXiv.2203.12602 arXiv:2203.12602 [cs].

[82] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. http://arxiv.org/abs/1412.0767 arXiv:1412.0767 [cs].

[83] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. http://arxiv.org/abs/1711.11248 arXiv:1711.11248 [cs].

[84] Jonas Traub, Philipp Marian Grulich, Alejandro Rodríguez Cuéllar, Sebastian Breß, Asterios Katsifodimos, Tilmann Rabl, and Volker Markl. 2021. Scotty: General and Efficient Open-source Window Aggregation for Stream Processing Systems. *TODS* 46, 1, Article 1 (March 2021), 46 pages.

[85] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. http://arxiv.org/abs/1706.03762 arXiv:1706.03762 [cs].

[86] Shivaram Venkataraman, Aurojit Panda, Kay Ousterhout, Michael Armbrust, Ali Ghodsi, Michael J. Franklin, Benjamin Recht, and Ion Stoica. 2017. Drizzle: Fast and Adaptable Stream Processing at Scale. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, Shanghai China, 374–389. doi:10.1145/3132747.3132750

[87] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. 2024. YOLOv10: Real-Time End-to-End Object Detection. doi:10.48550/arXiv.2405.14458 arXiv:2405.14458 [cs].

[88] Di Wang, Elke A. Rundensteiner, and Richard T. Ellison. 2011. Active complex event processing over event streams. *Proceedings of the VLDB Endowment* 4, 10 (July 2011), 634–645. doi:10.14778/2021017.2021021

[89] Eugene Wu, Yanlei Diao, and Shariq Rizvi. 2006. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. ACM, Chicago IL USA, 407–418.

doi:10.1145/1142473.1142520

[90] Renzhi Wu, Pramod Chunduri, Dristi J. Shah, Ashmitha Julius Aravind, Ali Payani, Xu Chu, Joy Arulraj, and Kexin Rong. 2024. SketchQL Demonstration: Zero-shot Video Moment Querying with Sketches. doi:10.48550/arXiv.2405.18334 arXiv:2405.18334 [cs].

[91] Steven Wu and Luke Bornn. 2018. Modeling Offensive Player Movement in Professional Basketball. *The American Statistician* 72, 1 (Jan. 2018), 72–79. doi:10.1080/00031305.2017.1395365

[92] Wenhao Wu, Xiaohan Wang, Haipeng Luo, Jingdong Wang, Yi Yang, and Wanli Ouyang. 2023. Bidirectional Cross-Modal Knowledge Exploration for Video Recognition with Pre-trained Vision-Language Models. http://arxiv.org/abs/2301.00182 arXiv:2301.00182 version: 2.

[93] Piyush Yadav and Edward Curry. 2019. VidCEP: Complex Event Processing Framework to Detect Spatiotemporal Patterns in Video Streams. In *2019 IEEE International Conference on Big Data (Big Data)*. 2513–2522. doi:10.1109/BigData47090.2019.9006018

[94] Piyush Yadav, Dhaval Salwala, and Edward Curry. 2021. VID-WIN: Fast Video Event Matching With Query-Aware Windowing at the Edge for the Internet of Multimedia Things. *IEEE Internet of Things Journal* 8, 13 (July 2021), 10367–10389. doi:10.1109/JIOT.2021.3075336

[95] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. 2015. Every Moment Counts: Dense Detailed Labeling of Actions in Complex Videos. *CoRR* abs/1507.05738 (2015). arXiv:1507.05738 http://arxiv.org/abs/1507.05738

[96] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. 2020. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. doi:10.48550/arXiv.1805.04687 arXiv:1805.04687 [cs].

[97] Haopeng Zhang, Yanlei Diao, and Neil Immerman. 2014. On complexity and optimization of expensive queries in complex event processing. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. Association for Computing Machinery, New York, NY, USA, 217–228. doi:10.1145/2588555.2593671

[98] Erkang Zhu, Silu Huang, and Surajit Chaudhuri. 2023. High-Performance Row Pattern Recognition Using Joins. *Proceedings of the VLDB Endowment* 16, 5 (Jan. 2023), 1181–1195. doi:10.14778/3579075.3579090

Received October 2024; revised January 2025; accepted February 2025