Seiden: Revisiting Query Processing in Video Database Systems

Jaeho Bang[*]
Georgia Institute of Technology
jaehobang@gatech.edu

Gaurav Tarlok Kakkar[*]
Georgia Institute of Technology
gkakkar7@gatech.edu

Pramod Chunduri
Georgia Institute of Technology
pramodc@gatech.edu

Subrata Mitra
Adobe Research
subrata.mitra@adobe.com

Joy Arulraj
Georgia Institute of Technology
arulraj@gatech.edu

ABSTRACT

State-of-the-art video database management systems (VDBMSs) often use lightweight proxy models to accelerate object retrieval and aggregate queries. The key assumption underlying these systems is that the proxy model is an order of magnitude faster than the heavyweight oracle model. However, recent advances in computer vision have invalidated this assumption. Inference time of recently proposed oracle models is on par with or even lower than the proxy models used in state-of-the-art (SoTA) VDBMSs. This paper presents Seiden, a VDBMS that leverages this radical shift in the runtime gap between the oracle and proxy models. Instead of relying on a proxy model, Seiden directly applies the oracle model over a subset of frames to build a query-agnostic index, and samples additional frames to answer the query using an exploration-exploitation scheme during query processing. By leveraging the temporal continuity of the video and the output of the oracle model on the sampled frames, Seiden delivers faster query processing and better query accuracy than SoTA VDBMSs. Our empirical evaluation shows that Seiden is on average 6.6 × faster than SoTA VDBMSs across diverse queries and datasets.

PVLDB Reference Format:
Jaeho Bang, Gaurav Tarlok Kakkar, Pramod Chunduri, Subrata Mitra, and Joy Arulraj. Seiden: Revisiting Query Processing in Video Database Systems. PVLDB, 16(9): 2289 - 2301, 2023.
doi:10.14778/3598581.3598599

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/georgia-tech-db/seiden_submission.

1 INTRODUCTION

To extract information embedded in videos, researchers have recently proposed a wide range of video database management systems (VDBMSs) [3, 9–11, 16, 23, 24, 31, 42, 43]. These systems seek to reduce query processing time over videos with a tolerable drop in query accuracy. They primarily support two types of queries: (1) computing the aggregate distribution of objects in a video, and (2) retrieving a subset of frames that contain the target objects of interest from a video. Consider these illustrative queries:

```
--- Aggregate Query
SELECT AVG(COUNT(CAR)) FROM UA-DETRAC
WITH CONFIDENCE > 95% AND ERROR < 0.2;
--- Retrieval Query
SELECT frame_id FROM UA-DETRAC
WHERE COUNT(TRUCK) > 0 WITH PRECISION > 0.95;
```

The aggregate query computes the *frame-averaged* count of cars in the given UA-DETRAC video [41]. The user specifies that the difference between the actual count and the approximate count returned by the VDBMS must fall within an *error bound* of 0.2 with a *confidence score* greater than 95%. Such a query may be issued by a traffic analyst interested in identifying busy intersections for guiding future lane expansions. In contrast, the retrieval query seeks to find frames containing a truck with *precision* greater than 0.95 (*i.e.*, at least 95% of all the frames returned must contain a truck). Such a query may be issued by a researcher working at an autonomous car company looking for specific traffic scenarios.

Prior Work. To accelerate these two types of queries, VDBMSs often leverage a lightweight *proxy model* that is faster (but less accurate) than the *oracle model*. The output of the proxy model, known as a *proxy label*, approximates the output of the oracle model and is cheaper to generate compared to running the oracle model. The proxy model is typically a compressed deep learning model (*e.g.*, Resnet-18 [18]). The oracle model is typically a heavyweight, object detection model (*e.g.*, MASK R-CNN [30]). To answer retrieval queries, SoTA VDBMSs use the lightweight proxy model to quickly filter out irrelevant frames and only pass a smaller subset of frames to the heavyweight oracle model [3, 19, 23, 24, 31]. To answer aggregate queries, these systems use the proxy model to quickly approximate the aggregate [23].

Limitations of Prior Work. The key assumption underlying these systems is that the proxy model is significantly faster than the oracle model. For example, the inference time of the MASK R-CNN oracle model is an order of magnitude higher than the Resnet-18 proxy model. However, with recent advances in computer vision, this assumption is *not valid*. For example, Yolov5s [40] is a recently released object detection model that is 11.2 × faster than MASK R-CNN and delivers higher model accuracy (*i.e.*, mAP values) than MASK R-CNN on the COCO dataset [29]. This observation is important because Mask-RCNN is used as the oracle model in many SoTA VDBMSs [23, 24, 26].

Yolov5s achieves reduced inference time by improvements in the box proposal algorithm necessary for detecting objects. The

[*] Both authors contributed equally to the paper.

Proceedings of the VLDB Endowment, Vol. 16, No. 9 ISSN 2150-8097.
doi:10.14778/3598581.3598599

Table 1: Qualitative comparison of SoTA VDBMSs – TASTI and TASTI-PT use the farthest-point first (FPF) clustering algorithm to build an index of representative frames.

VDBMS	Proxy Model	Label Propagation
TASTI-PT [26]	RESNET-18	FPF Clustering
TASTI [26]	Custom RESNET-18	FPF Clustering
BLAZEIT [23]	Custom RESNET-18	None
SEIDEN	None	Temporal Interpolation

inference throughput of YOLOV5S and the RESNET-18 proxy model on 960×540 images is 140 and 170 fps, respectively. Due to this massive drop in inference time of oracle models, it is no longer beneficial to pass the frame through a proxy model before sending it to the oracle model.

PROXY-BASED VDBMSs. We next provide a brief overview of how two SoTA proxy-based VDBMSs answer these two types of queries.

❶ TASTI. TASTI [26] is a SoTA VDBMS tailored for answering aggregate and retrieval queries. It trains a custom proxy model that is based on a pre-trained RESNET-18 feature extractor for the given video dataset using triplet loss with respect to the MASK R-CNN [30] oracle model. It also supports another mode (TASTI-PT) where the VDBMS directly uses a pre-trained RESNET-18 model without fine-tuning. In both modes, the VDBMS builds an *index* of representative frames using a farthest-point-first (FPF) algorithm on the RESNET-18 features of all the frames in the video. During query processing, the VDBMS invokes the oracle model (MASK R-CNN) only on the representative frames and propagates the labels to other frames based on pairwise RESNET-18 feature distance.

❷ BLAZEIT. BLAZEIT [23] trains a custom TINY RESNET model to directly answer the aggregate query. It applies the trained proxy model on all the video frames. To generate labeled data for training the TINY RESNET model, it runs the MASK R-CNN oracle model on a small subset of frames to derive the answer for the aggregate query (*e.g.*, number of cars).

The key limitation of these systems is that the inference time of the RESNET-18 and TINY RESNET *proxy* models is now comparable to *oracle* models like YOLOV5S. So, running the oracle model (YOLOV5S) on all the frames delivers more accurate results in a comparable time. Further, replacing these proxy models with even more lightweight proxy models leads to a significant accuracy drop.

OUR APPROACH. In this paper, we present SEIDEN, a VDBMS tailored for lightweight oracle models like YOLOV5S. SEIDEN circumvents the limitations of SoTA VDBMSs by directly running the oracle model on a subset of frames to answer both retrieval and aggregate queries. Table 1 presents a qualitative comparison between SEIDEN and SoTA proxy-based VDBMSs.

❶ INDEX CONSTRUCTION PHASE. While ingesting a video dataset, SEIDEN constructs a query-agnostic index by running the oracle model over a subset of frames from the video. To lower the overhead of decoding frames, SEIDEN simply picks I-frames that are spaced out across the video during this phase. This sampling scheme leverages the property that the target videos tend to be a collection of frames with smooth motion of objects. This step is performed only once when the video is loaded into the VDBMS.

❷ QUERY EXECUTION PHASE. During query execution, SEIDEN utilizes an *exploration-exploitation* scheme to sample additional frames for answering the query. SEIDEN manages each video as a collection of *segments* (*i.e.*, a sequence of contiguous frames). To explore the video, SEIDEN samples frames from *segments* that were not previously sampled. Frames in these segments are likely to contain more information than frames that are temporally close to already sampled frames. SEIDEN combines this exploration strategy with an exploitation strategy. SEIDEN runs the oracle model on the sampled frames and identifies segments where a *label difference* exists between the first and the last frames of the segment. It then picks more frames in those segments to exploit the label difference.

Unlike SoTA VDBMSs, SEIDEN employs a simpler approach that leverages the *temporal continuity* of the video to assign proxy labels to the remaining unsampled frames. Once the *oracle labels* of the sampled frames are collected using the oracle model, SEIDEN interpolates the labels to the remaining frames in the video to derive their *proxy labels*. These proxy labels are then utilized to answer both aggregate and retrieval type of queries.

We note that SEIDEN's query processing technique is only possible with the advent of lightweight oracle models (like YOLOV5S [40]). Otherwise, the computational cost of running the oracle model on all the selected frames will dominate the query processing time.

CONTRIBUTIONS. The key contributions of this paper are:

- We motivate the need for revisiting query processing in VDBMSs by showing that there is no longer a significant gap in inference time between the oracle model and the proxy model.
- We present SEIDEN, a VDBMS that simply leverages the lightweight oracle model and the temporal continuity of the video to answer queries faster than the SoTA VDBMSs.
- We propose a novel multi-arm bandit based sampling algorithm to accelerate retrieval and aggregate queries. It uses a query-dependent reward function to balance the trade-off between exploiting high-rewarding video regions versus exploring unexplored regions.
- We present a label propagation algorithm that leverages the temporal continuity of videos to inexpensively and effectively assign proxy scores to all unsampled frames.
- We empirically illustrate that SEIDEN works well across three queries on four real-world datasets. We conduct ablation studies to examine the efficacy of SEIDEN and explain why other features and proxy models are not effective in practice.

2 BACKGROUND & MOTIVATION

In this section, we provide a brief overview of video compression and the usage of lightweight proxy models in VDBMSs. We then describe recent advances in object detection to motivate the need for revisiting query processing in VDBMSs.

2.1 Video Compression

Video compression codecs leverage the temporal continuity between the nearby frames [37]. Specifically, they compress the video as a collection of GOPs (group of frames). Each GOP starts with a keyframe (also referred to as I-frame) which is independently decoded. The remaining frames in the GOP referred to as P- and B-frames store the deltas from neighboring frames and hence are not

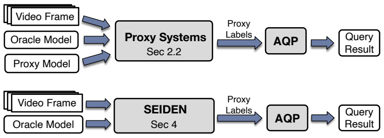


Figure 1: Video Analytics in Proxy-based VDBMSs. Proxy systems (*e.g.*, TASTI [26]) take video data, the oracle model, and a proxy model (*e.g.*, Resnet-18) as inputs, whereas Seiden does not require a proxy model. AQP signifies approximate query processing (*e.g.*, SUPG [25].)

independently decodable. Seiden exploits the temporal continuity in videos(§4) to accelerate queries.

2.2 Proxy Models in VDBMSs

In the last five years, researchers have presented several VDBMSs for accelerating queries over videos using lightweight proxy models [3, 9, 19, 23, 24, 26, 31]. These systems assume the proxy model is significantly faster than the oracle model and, as a result, execute the proxy model on all video frames.

Figure 1 provides an overview of proxy-based VDBMSs. They take the videos, query, and the oracle model as inputs. Then, using the proxy model, they assign a proxy score to every frame in the videos. For a retrieval query, the proxy score is a value between 0 and 1 (signifying the probability of the frame satisfying the query predicate). For an aggregate query, a proxy score is a number that approximates the aggregate for the given frame. These proxy scores are then used in an Approximate Query Processing (AQP) algorithm to ensure that the user's accuracy constraints are satisfied (*e.g.*, SUPG [25] for retrieval queries and BLAZEIT [23] for aggregate query). The output of the AQP algorithm is the final query result. As discussed earlier §1, the assumption that the proxy model is much quicker than the oracle model is no longer valid, and the architecture needs to be reconsidered.

2.3 Object Detection Models

Object detection models are a class of deep learning networks that localize and classify the objects present in a given frame. They include MASK R-CNN, SSD [30], EfficientDet [39], and Yolov5s [40].

Mask-RCNN. MASK R-CNN [17] is an object detection model proposed in 2017 that delivers highly accurate results. The model network consists of two components. The first component extracts features from the given frame using a series of convolutional layers (*e.g.*, Resnet-18, VGG, or Resnet-50). The second component uses the extracted features to propose boxes in the image that are likely to contain the objects of interest. The region proposal component is 14.5× slower than the feature extractor. So, while MASK R-CNN returns accurate results, it is significantly slower than Resnet-18.

Yolov5. Recently proposed models, like Yolov5, lower the computational complexity of the object detection task by combining the two phases of feature extraction and box proposal. Instead of proposing boxes based on the final feature extraction layer, the model proposes boxes even on intermediate feature extraction layers. To improve accuracy, instead of relying on a lot of expensive convolutional layers, it leverages upsampling layers to generate robust and accurate features.

Table 2: Accuracy and Inference latency of SoTA object detection models – Yolov5 and EfficientDet are significantly faster than MASK R-CNN and deliver comparable accuracy on the COCO dataset [39, 40]

Model Name	Image Size	mAP^{val}	Latency (ms)
EfficientDet-D1	640	40.2	13.5
Yolov5 m	640	45.4	8.2
EfficientDet-D2	768	43.5	17.7
MASK R-CNN	768	44.3	103
EfficientDet-D5	1280	51.3	72.5
MASK R-CNN	1280	50.2	234
Yolov5 m6	1280	51.3	11.1

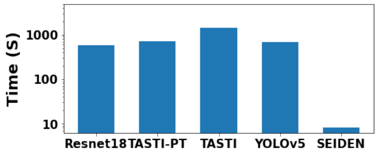


Figure 2: Motivating Experiment – Comparison of time taken to construct an index using different techniques. This experiment invalidates the assumption that the oracle model (*i.e.*, Yolov5) is significantly slower than the feature extractor (*i.e.*, Resnet-18).

Comparison of Accuracy and Inference Latency. We next compare the accuracy and inference speed of Yolov5, MASK R-CNN, and EfficientDet [39] family of object detectors on the COCO dataset [29]. The COCO dataset contains 120 K images and 880 K object instances. Given the size and diversity of objects in the dataset, it is widely used for comparing different object detectors. The results are shown in Table 2. We evaluate the models on a Titan V100 GPU. Across diverse image sizes, Yolov5 is faster than EfficientDet models that deliver comparable mean average precision (mAP) values. These EfficientDet models in turn are faster than the MASK R-CNN model that delivers a comparable mAP value.

Specifically, for 640×640 images, Yolov5 m has 12% higher mAP and 40% lower latency compared to EfficientDet D1. With an image size of 768, the mAP value of EfficientDet is 2% lower than that of MASK R-CNN, but the latency is improved by a factor of 5.8×. With increasing image size, the mAP scores are improved for all models. However, the mAP scores of EfficientDet and MASK R-CNN for an image size of 768 are still lower than that of Yolov5 m on an image size of 640. The gap in inference latency widens on larger images. For an image size of 1280, Yolov5 m6 is 21× faster than MASK R-CNN. Given these observations, we contend that future VDBMSs would use these faster oracle models.

2.4 Motivating Experiment

In this experiment, we compare the time taken to build an index over a video using five techniques: (1) a feature extractor (Resnet-18), (2,3) two variants of TASTI [26], (4) Yolov5, and (5) Seiden. While TASTI-PT uses a pre-trained Resnet-18 model for feature extraction, TASTI trains a custom Resnet-18 network. Both TASTI and TASTI-PT internally run the Resnet-18 feature extractor on all the video frames. We evaluate these indexing algorithms on a UA-DETRAC traffic surveillance video with 84 K 960×540 frames.

Query Processing Time. The results are shown in Figure 2. The most notable observation is that YoloV5 is either on par or faster than the tasti VDBMS that creates lightweight models. This invalidates the assumption that the oracle model is significantly slower than the feature extractor. The index construction time with YoloV5 is 678 s. It is 1.1× and 2.1× faster than tasti-pt and tasti, respectively. Resnet-18 is slightly faster than YoloV5. It takes 587 s to build the index. However, since the feature extractor is less accurate than the YoloV5, there is no justification for using Resnet-18 during index construction. This illustrates that the fundamental assumption in SoTA VDBMSs, like tasti, is not valid.

Another notable observation from this experiment is that Seiden is 84.8× faster than YoloV5. We defer a detailed description of how Seiden constructs an index to §3. The key reason why Seiden is better than tasti is that there is no longer a significant gap in inference time between the feature extractor (*e.g.*, Resnet-18) and the object detector (*e.g.*, YoloV5). So, it is not possible to justify running Resnet-18 on all the frames to derive proxy scores. Seiden leverages the *temporal continuity* property of videos to adopt a simpler sampling scheme that not only reduces query execution time but also delivers accurate results.

Accuracy. We next compare the accuracy and inference latency of mask r-cnn and YoloV5 models on the ua-detrac video. While it takes 1.7 h to run mask r-cnn over the video, YoloV5 only requires 0.15 h (11.2× faster). Resnet-18 takes 0.13 h to complete. To measure accuracy, we compare the maP scores with an intersection over union (IoU) threshold of 0.5. With the car object category, mask r-cnn and YoloV5 deliver 0.37 and 0.4 mAP scores, respectively. Thus, YoloV5 is both faster and more accurate than mask r-cnn which is used in SoTA VDBMSs.

3 SYSTEM OVERVIEW

The objective of Seiden is to minimize the overall time taken for query processing while meeting the user's accuracy constraints. The query processing time (t_{QP}) consists of index construction time (t_{IC}) and query execution time (t_{QE}) using the constructed index. The VDBMS takes the videos, query, and oracle model as inputs and seeks to quickly generate proxy labels with a tolerable drop in accuracy. Seiden seeks to minimize the query processing time t_{QP}:

$$\min \quad t_{QP} = \frac{t_{IC}}{N} + t_{QE} \tag{1}$$

Seiden constructs an index only once for a given video dataset and oracle model. So, the index construction time is amortized across all the N queries over the same video with the same oracle model. Query execution time consists of the time taken to obtain additional proxy labels and to confirm whether the result meets the accuracy constraint using an approximate query processing (AQP) technique. To ensure that the proxy labels meet the user's accuracy constraint, Seiden relies on the AQP techniques presented in blazeit [23] and supg [25]. The time taken for the AQP technique depends on the difference between the proxy labels (S_p) and the oracle labels (S_o). Thus, t_{QP} is given by:

$$t_{QP} \propto \alpha * |S_p - S_o|^2 + (1 - \alpha) * \frac{t_{IC}}{N} \tag{2}$$

Seiden seeks to meet two objectives: (1) minimize the difference between the proxy labels that it generates and the oracle labels, and (2) minimize the time taken to generate the proxy labels.

3.1 Architecture

As shown in Figure 3, to meet its objectives, Seiden strategically performs IndexConstruction and QueryExecution. During the IndexConstruction phase, Seiden samples a subset of frames from the video and directly runs the oracle model on them (*i.e.*, YoloV5s). Unlike prior VDBMSs that rely on a proxy model during this phase, Seiden directly uses the oracle model. The index consists of the oracle labels for these sampled frames.

Seiden later uses these labels in the index during the QueryExecution phase to answer a given query. During QueryExecution, Seiden samples additional frames based on the query's predicate and the contents of the video using MABSampling (§4.2). It again runs the oracle model on these additional frames. Lastly, it propagates the oracle labels of all the frames sampled during the IndexConstruction and QueryExecution phases to derive the proxy labels of all the other frames in the video. Seiden then passes on these proxy labels onto the SoTA AQP algorithms such as blazeit [23] and supg [25].

Approximate Query Processing (AQP) Algorithms. The purpose of the AQP algorithms is to derive statistical guarantees on meeting the query's accuracy constraint. These algorithms vary based on the query being asked.

▶ **Aggregate Query.** For aggregate queries, the user specifies the confidence score and error bound.

```
--- Aggregate Query
SELECT AVG(COUNT(CAR)) FROM UA-DETRAC
WITH CONFIDENCE > 95%  AND ERROR < 0.2
```

Here, the error bound represents the difference between the aggregate reported by the VDBMS and the value of the aggregate obtained by running the oracle model on all the frames. The objective is to guarantee that the error bound is lower than 0.2 with the desired confidence (*i.e.*, 95%). Seiden relies on the aggregate optimizer in blazeit [23]. blazeit requires the error bound and confidence level as input parameters along with proxy scores, and outputs the number of additional samples needed to generate the query result that satisfies the given accuracy constraint. It utilizes Empirical Bernstein Stopping (EBS) algorithm to satisfy the bounds.

▶ **Retrieval Query.** For retrieval queries, the user specifies the accuracy constraint in terms of either a precision *or* recall constraint.

```
--- Retrieval Query
SELECT frameID FROM UA-DETRAC
WHERE COUNT(TRUCK) > 2 WITH PRECISION > 0.95
```

Here, the query seeks to retrieve frames that satisfy an arbitrary predicate (*e.g.*, contains two trucks). The objective is to guarantee that either a subset of the retrieved frames are true positives (precision constraint) or a subset of all the true positive frames are retrieved (recall constraint). We consider the oracle model's label as ground truth and measure the accuracy of the proxy label with respect to that. To confirm that the accuracy constraint is met, Seiden relies on the importance sampling algorithm presented in supg [25]. This algorithm iteratively computes the appropriate confidence threshold using the oracle model to satisfy the given accuracy constraint. If the oracle labels of the sampled frames (*i.e.*,

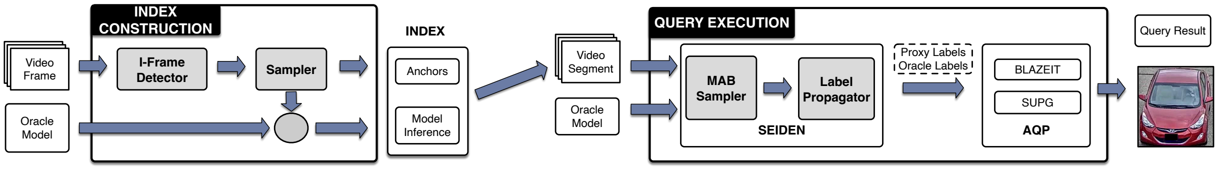


Figure 3: Overview of Seiden– During index construction, Seiden samples a subset of I-frames, referred to as anchors, and runs the oracle model over them to construct the index. During query execution, Seiden selects additional frames using MABSampling and runs the oracle model on the selected frames. Lastly, Seiden propagates the oracle labels of all the frames that were sampled during the two phases to assign proxy labels to all the other frames in the video. These proxy labels are then used by the AQP algorithms to generate query results.

Table 3: Notation

Symbol	Description
X_s	Input video
B	Number of video segments
$\hat{r}_{k,t}$	Expected reward for arm k up to time t
$N_{k,t}$	Number of draws from arm k up to time t
N	Total number of draws across all arms
$X_{k,i}$	i^{th} draw from arm k arm
f	aggregation expression
$\hat{\sigma}_{k,t}$	Empirical standard deviation for arm k arm up to time t

proxy scores in supg) are representative of those of the other frames in the video, then the sampling algorithm terminates early.

4 QUERY PROCESSING

We next present the algorithms that Seiden uses to process a query. The notation used in the paper is listed in Table 3. Query processing consists of two phases: IndexConstruction (§4.1) and QueryExecution (§4.2). The IndexConstruction step is query-agnostic. Seiden selects a random subset of I-frames from the video, known as *anchor frames*. It logically splits the video into a sequence of segments based on the selected anchors.

In the QueryExecution phase, Seiden uses adaptive sampling to select additional frames from the video. It prioritizes sampling from video segments that provide relevant information for the query. Seiden takes a sampling budget N as input. It uses a configurable parameter, *anchor sampling ratio* α, to distribute the sampling budget N between the two phases. Specifically, Seiden selects αN anchor frames (I-frames) in the IndexConstruction phase and remaining $(1 - \alpha)N$ samples using adaptive sampling in the QueryExecution phase. This approach allows Seiden to strike a balance between exploration and exploitation (§6.5). We next describe these two phases in more detail.

4.1 Index Construction

Seiden leverages the intrinsic *temporal continuity* property of videos to build the index. Algorithm 1 presents the IndexConstruction algorithm. It takes the source video X_s, the sampling budget N, the oracle model M, and the anchor sampling ratio α as inputs. Anchor sampling ratio, α, represents the fraction of the total sampling budget N spent during IndexConstruction. First, Seiden selects αN frames using random sampling of the video's I-frames (line 4). If αN exceeds the total number of the video's I-frames, then Seiden selects all I-frames and defers the remaining sampling budget to QueryExecution (line 6). It then runs the oracle model M on all of the sampled I-frames and computes the inference results (*e.g.*, objects detected) (line 8). These inference results are stored in a key-value index where the key represents the sampled frame id and the value represents the inference result obtained from the oracle model on that frame. Seiden leverages the index later during the QueryExecution phase.

```
Algorithm 1: Index Construction
  Input  :Video $X_s$, Sampling budget $N$, Oracle model $M$,
          Anchor sampling ratio $\alpha$
  Output :Anchor indices $I_A$, Anchor labels $L_A$
1 Function IndexConstruction ($X_s, N, M, \alpha$):
2     $I_A$ = getIFrames($X_s$)                    ▷ Get I-Frames from Video
3     if $\alpha N < len(I_A)$ then
4         $I_A$ = randomSample($I_A$, $\alpha N$)     ▷ Sample subset of I-Frames
5     else
6         $I_A = I_A$                               ▷ Select all I-Frames
7     end
8     $L_A$ = M($X_s, I_A$)          ▷ Get oracle labels of all selected anchors
9     return $I_A, L_A$
```

The reason behind using I-frames as anchor frames is twofold. First, by using I-frames, we save video decoding time as they are independently decodable. Secondly, utilizing I-frames leverages the optimizations of the video compression codec. The codec selects I-frames that are evenly spaced or when a drastic scene change has happened. With I-frames as the anchor frames, Seiden decreases the chance of missing out on drastic changes. Seiden constructs the index only once for a given video X_s and oracle model M. The cost of IndexConstruction is amortized across all queries over X_s using M, even for other predicates like $COUNT(VAN) > 0$ or $AVG(COUNT(TRUCK))$).

4.2 Query Execution

Algorithm 2 presents the QueryExecution algorithm, during which Seiden leverages the anchors selected during IndexConstruction. QueryExecution consists of two steps: (1) MABSampling and (2) LabelPropagation.

MABSampling. In this phase, Seiden is aware of the predicate (for retrieval queries) and aggregate expression for aggregate queries. Seiden utilizes it to adaptively sample additional frames. The objective is to sample more frames from the video segments that provide relevant information for the query to get a better estimate. Seiden models the sampling problem as *multi-armed bandit* problem.

multi-armed bandit background. A multi-armed bandit problem [6] is a sequential allocation problem. There are B competing arms, and each arm provides a random reward r_k. A player repeatedly selects one of the arms and collects the reward. The goal

```
Algorithm 2: Query Execution using Constructed Index
  Input  : Video $X_s$, Sampling budget $N$, Oracle model $M$,
           Anchor sampling ratio $\alpha$, Anchor indices $I_A$,
           Anchor labels $L_A$
  Output : Sample indices $I_S$, Sample labels $L_S$
1  Function QueryExecution ($X_s, L, M, I_A, L_A$):
2      $B$ = generateArms($I_A$)              ▹ MAB Prep – Generate Arms
3      $S_I, S_L$ = [], []
4      for $i \leftarrow 0$ to $N - len(I_A)$        ▹ Adaptive Sampling Until Budget
5       do
6          $b$ = selectOptimalArm($B, L_A$)        ▹ Select Optimal Arm
7          $s_i = sample(b)$                  ▹ Random Sample From Arm
8          $s_l$ = Inference($X_s, s_i, M$)        ▹ Get Model Inference Result
9          $S_I$.append($s_i$)
10         $S_L$.append($s_l$)
11     end
12     return $S_I, S_L$
```

is to maximize the accumulated reward after limited draws. It exemplifies the exploitation-exploration tradeoff dilemma. The player can either exploit the arm with the maximum expected reward or explore other arms to get better information about the expected rewards of other arms. The Upper Confidence Bound (UCB) algorithm is a formal way to deal with this problem. At each step t, the algorithm computes the $UCB_{k,t}$ Equation (3) for each arm k.

$$UCB_{k,t} = \hat{r}_{k,t-1} + c\sqrt{\frac{2\ln N}{N_{k,t-1}}} \tag{3}$$

$\hat{r}_{k,t-1}$ is the expected reward of arm k till step $t-1$, $N_{k,t-1}$ is the total number of draws from arm k up to step $t-1$, and N is the total number of draws across all arms. This represents a tradeoff between exploitation and exploration. The first term corresponds to exploiting the arm with the optimal expected reward. The second term corresponds to exploration as it prefers selecting the least explored arm. The algorithm prefers the arm that maximizes UCB. The c parameter controls the trade-off between exploitation and exploration. Higher values of c prefer exploration (§6.6).

Problem Formulation. Seiden models the sample selection problem as multi-armed bandit problem.

Arms: Using the anchors selected during IndexConstruction, Seiden logically splits the original video into segments. Each video segment is considered as an arm for the multi-armed bandit formulation. The objective is to balance the trade-off between exploring less sampled segments of the video and exploiting the segments that yield higher rewards. To achieve this, Seiden uses UCB algorithm to determine the segment to sample next.

Rewards: We explain how Seiden assigns the reward to each segment. The reward for each video segment is dependent on the query type.

$$\hat{r}_{k,t-1} = \hat{\sigma}_{X_{k,t-1}} \quad \text{where} \tag{4a}$$

$$\textbf{Retrieval queries:} \quad X_{k,t-1} = \begin{cases} 1, & \text{if sample satisfies predicate} \\ 0, & \text{otherwise} \end{cases} \tag{4b}$$

$$\textbf{Aggregate queries:} \quad X_{k,t-1} = f(x_{k,t-1}) \tag{4c}$$

$X_{k,i}$ is the $i-$th sample drawn from arm k. For retrieval queries with precision or recall constraints, the random variable $X_{k,t-1}$ is 1 if it satisfies the predicate and 0 otherwise (Equation (4b)). For aggregate queries, $X_{k,t-1} = f(x_{k,t-1}) \in \mathbb{R}$, where f is aggreagte expression in the query (Equation (4c)). After $t-1$ draws from video segment k, Seiden defines its reward as the empirical standard deviation $\hat{\sigma}_{k,t-1}$ of the random variable $X_{k,t-1}$ (Equation (4a)). The intuition behind the reward is that Seiden must sample more from the uncertain video segments. The variance of the predicate (for retrieval queries) and the aggregation expression (for aggregate queries) in a video segment indicates uncertainty.

As listed in Algorithm 2, Seiden first logically constructs video segments (arms) using the anchors selected during IndexConstruction (line 2). Next, Seiden selects the segment with the maximum UCB Equation 3 score (line 6) and randomly draws a frame from the segment (line 7). It then runs the oracle model over the sampled frame (line 8). This sampling process continues until QueryExecution exhausts the sampling budget.

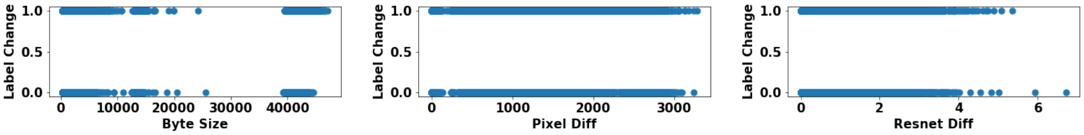
4.3 Label Propagation

After the MABSampling step, Seiden propagates the labels from the sampled frames to the remaining frames. To propagate the label to a frame q, Seiden need a distance metric d to identify the set of sampled frames closest to q with respect to d. Seiden uses the temporal distance between frames as the distance metric. The reasons for this design decision are twofold. First, it leverages the temporal contiguity property of the video. Second, it is inexpensive to compute the temporal distance between frames.

Example. Consider two sampled frames whose frame ids are 10 and 20. The aggregate query results using the oracle model for the two sampled frames are 5 and 10, respectively. Then, for a non-sampled frame whose frame-id is 12, the result of the aggregate query is linearly interpolated to 6. The same interpolation logic applies to retrieval queries. If sampled frames' retrieval query results are 0.3 and 0.4, respectively, then the non-sampled frame whose frame-id is 12 would have a probability of 0.32.

After the LabelPropagation step, every non-sampled frame in the video gets an approximate proxy label. For an aggregate query, Seiden derives an approximate aggregate for all the remaining frames, while for a retrieval query with a precision or recall constraint, Seiden derives an approximate probability. The proxy labels of non-sampled frames and oracle labels of sampled frames are then sent to the AQP modules for further processing.

Since MABSampling adjusts sample selection based on the contents of the video, it priorities selecting samples from regions with high uncertainty. As a result, the ground truth label of any unsampled frame is highly likely to be correlated with the labels of temporally-nearby, sampled frames. So, Seiden adopts a simple linear interpolation based on temporal distance to generate proxy labels. Besides temporal distance, we also consider four more schemes for label propagation: (1) pixel distance, (2) Resnet-18 feature distance, (3) pixel and temporal distance (hybrid), and (4) Resnet-18 feature and temporal distance (hybrid). However, as we show in §6.3, §6.4, and §6.8, temporal linear interpolation is inexpensive and effective for a wide range of queries and datasets.



(a) Frame Byte Size

(b) Frame Pixel Value

(c) Frame Resnet Feature

Figure 4: Correlation between Features and Oracle Model Label Changes – There is no statistically significant correlation between label changes and differences between features of frames using these three feature functions: (1) frame byte size, (2) frame pixel value, and (3) frame Resnet-18 value. The Pearson correlation coefficients for these features are 0.014, 0.134, and 0.274, respectively.

5 INEFFECTIVE FEATURES

SoTA VDBMSs [24, 26] leverage image features to quickly generate proxy labels. In this section, we examine the efficacy of these features: (1) frame byte size (examined in this paper), (2) frame pixel values [24], and (3) deep features of a frame using Resnet-18 [26]. The objective of this analysis is to determine if these features are *viable* in the presence of a lightweight oracle model. For any feature to be considered *viable*, it needs to meet two constraints. First, the feature extraction process must be significantly faster than running the lightweight oracle model on each frame. If the feature extraction process takes too long, it would be more accurate to simply run the oracle model on all frames. Second, the feature must be an accurate proxy signal for changes in the oracle label. This is crucial for the exploitation strategy used in Seiden, which is based on the difference between labels. We empirically found that none of these features meet both constraints, indicating a need to revisit the design decisions in SoTA VDBMSs.

▶ **Frame Byte Size.** Since Seiden aims to quickly generate accurate proxy scores, we explore whether frame byte size is suitable for picking additional samples during query processing. Frame byte size can be easily accessed without decoding the video data and is correlated with changes in frame content across nearby frames. B and P frames have smaller byte sizes than I frames since they only encode the difference with respect to a nearby frame. We examine the correlation between label changes and byte size difference on the Cherry traffic-surveillance dataset [33] that is obtained from a static camera (Table 4). The query focuses on the existence of a car (retrieval query). The result in Figure 4a shows a statistically insignificant correlation (0.014) between byte size and the oracle model label. The reasons are twofold. The byte size of I-frames does not capture changes in frame content as they are not encoded as a difference with respect to a nearby frame. B- and P-frames' byte size changes are not directly related to the objects of interest. For example, changes in environmental conditions lead to higher byte size even though they do not impact the query label.

▶ **Frame Pixel Difference.** Comparing the difference between pixels in two frames is more expensive than retrieving their byte size, as it involves decoding the frame. However, computing pixel difference is still faster than running the oracle model. Further, it should intuitively capture changes in the frame content, as the oracle model also operates on these pixels [24]. To validate this hypothesis, we conduct the same experiment using the frame pixel difference feature. The result in Figure 4b indicates a correlation of 0.134 between label change and pixel difference. Although this correlation is stronger than that of frame byte size, it is still too low for Seiden to efficiently use this feature to sample additional frames. The reason for the weak correlation is that the oracle model performs non-linear transformations on pixel space to derive the label space. For example, a background change may result in a high pixel value difference, but it may not affect the label.

▶ **Deep Features.** To cope with changing environmental conditions, we consider a deep feature extractor (pre-trained Resnet-18). The rationale for using the feature extractor is that it runs faster than the oracle models used in prior VDBMSs (like mask r-cnn). The generated features should be more robust compared to pixel differences. To test this hypothesis, we conduct the same experiment using the deep features. From Figure 4c, we observe that the correlation between label change and pixel difference is 0.274. While it is higher than the pixel difference (0.134), it is still too low for Seiden to effectively use these features for sampling additional frames. The reason why the feature space is *not* strongly correlated with the oracle label space is that elements of the feature vector obtained from the pre-trained Resnet-18 model capture information about all the objects in the frame (*e.g.*, cars, trucks, vans, *e.t.c.*). These elements are not fully relevant for a query focusing on cars.

Faster + Accurate Proxy Model. In our experiments, none of the features we examined are able to provide both cost-effective and accurate proxy labels. However, in case a faster and more precise proxy model is developed in the future, VDBMSs must make an informed decision on when to use such a proxy model, by comparing the execution time and accuracy of running the proxy model on all the frames to that of running the oracle model on the sampled frames as done in Seiden.

6 EVALUATION

We seek to answer the following questions in our evaluation:

- **RQ1.** How does Seiden's index construction time compare to SoTA VDBMSs?(§6.2)
- **RQ2.** How does the execution time of Seiden compare against that of other VDBMSs on aggregate queries?(§6.3)
- **RQ3.** How does the F1-score of Seiden compare against that of other VDBMSs on retrieval queries?(§6.4)
- **RQ4.** How does varying the anchor sampling ratio (r) affect Seiden's query accuracy? (§6.5)
- **RQ5.** How does varying the c parameter in MAB affect Seiden's query accuracy? (§6.6)
- **RQ6.** How does reusing Seiden's anchors and samples affect query accuracy? (§6.7)

Table 4: Datasets – The key properties of the datasets used in our evaluation. We list the number of evaluated frames and the percentage of frames containing target object (*i.e.,* number of TRUE frames).

Video	Object Name	Frames Per Second	# of Frames	Object Selectivity
Cherry	CAR	24	100k	36%
Cherry	BUS	24	100k	2.5%
UA-DETRAC	CAR	10	84k	94%
Dashcam	CAR	24	75k	60%
Dashcam	BUS	24	75k	5.6%
Jackson	CAR	30	300k	4%

6.1 Experimental Setup

▶ **Baselines.** We compare Seiden against the following baselines:

svm. We implement a svm-based model inspired by [31] using the scikit-learn framework [8]. We train the svm using the oracle labels on a subset of frames randomly selected from the video.

Resnet-18. We implement a Resnet-18-based model inspired by BlazeIt [23]. We append a custom fully connected layer onto a pre-trained Pytorch model based on the type of query. For instance, the model returns an aggregate in the case of an aggregate query. We train this model on the oracle labels of a subset of video frames.

tasti-pt. While tasti uses triplet loss to train a custom feature extractor on labeled video data to generate features, we replace the custom feature extractor with a pre-trained Resnet-18 model. We refer to this baseline as tasti-pt as done in the tasti paper [26]. To determine the set of sampled frames, tasti applies furthest-point-first (FPF) algorithm [14] and a variant of K nearest neighbors [2] for propagating the labels to the rest of the video frames. We use the implementation of tasti-pt by the original author.

▶ **Datasets.** We evaluate the baselines on four datasets: Cherry [33], UA-DETRAC [41], Dashcam, and Jackson [26]. The key properties of these traffic-surveillance datasets are summarized in Table 4.

Cherry. Cherry dataset is a one-hour video with 100 K frames obtained at an intersection in Seattle during morning time [33]. The percentage of frames containing CAR in this dataset is 36% and that of BUS is 2.5%. The resolution of this dataset is 960x540.

UA-DETRAC. UA-DETRAC consists of numerous short traffic camera videos (each of one-minute duration). We concatenated these shorter videos to create a longer video. The resolution of this dataset is 960x540. Many of these shorter videos display dense traffic settings. So, this dataset has a high object occupancy of 94%.

Dashcam. Dashcam is a traffic video dataset recorded from a dashcam footage that is 52 minutes long. Hence, in this video dataset, the background in addition to the foreground is moving. This video's selectivity for CAR is 60% while that of BUS is 5.6%. The resolution of this dataset is 960x540.

Jackson. Jackson is a dataset used for evaluating SoTA VDBMSs [26]. The duration of this video is 2 hours 46 minutes with 300 K frames at a resolution of 300×300. Similar to Cherry and UA-DETRAC, Jackson is obtained from a static camera. However, unlike other datasets, it is collected at nighttime. The selectivity of CAR is 4%.

▶ **Queries.** Table 5 presents the queries that we use to evaluate the baselines. We consider three types of queries: (1) aggregate query, (2) retrieval query with precision constraint, and (3) retrieval query with recall constraint.

Table 5: Query Templates

Type	Query Template

Q1 - Aggregate	`SELECT COUNT(CAR) FROM $DATASET`
	`WITH CONFIDENCE > $1 AND ERROR < $2;`

Q2 - Precision	`SELECT frameID  FROM $DATASET`
	`WHERE COUNT($OBJ) > 0 WITH PRECISION > $3;`

Q3 - Recall	`SELECT frameID FROM $DATASET`
	`WHERE COUNT(CAR) > 0 WITH RECALL > $4;`

The first query counts the total number of objects in a video. The aggregate must meet the confidence and error bound constraints of the user. The default parameter for $1 is 0.95 and that of $2 is 0.1. The second query retrieves the set of frames that satisfy the given predicate (*i.e.*, existence of a car). The user specifies a precision constraint (*i.e.*, a significant subset of frames that contain a car must be returned). The default parameter for $OBJ is CAR and that of $3 is 0.95. The third query is a retrieval query with a recall constraint. In this case, a significant subset of frames that are returned by the VDBMS must contain a CAR (*i.e.*, they must be precise). The default parameter for $4 is 0.95.

We evaluate the VDBMSs on a diverse set of queries and datasets to examine their broader applicability. $DATASET is Cherry, UA-DETRAC, Dashcam, or Jackson.

▶ **Oracle Model.** We use Yolov5s [36] as the reference inference model. We rank all the systems based on how well their labels compare against those assigned by Yolov5s.

▶ **Hardware Environment.** We perform the experiments on a server with these specifications:

- CPU: 16 Intel(R) Xeon(R) Gold 6134 @ 3.20GHz
- GPU: 1 Geforce RTX 2080 Ti
- RAM: 385 GB

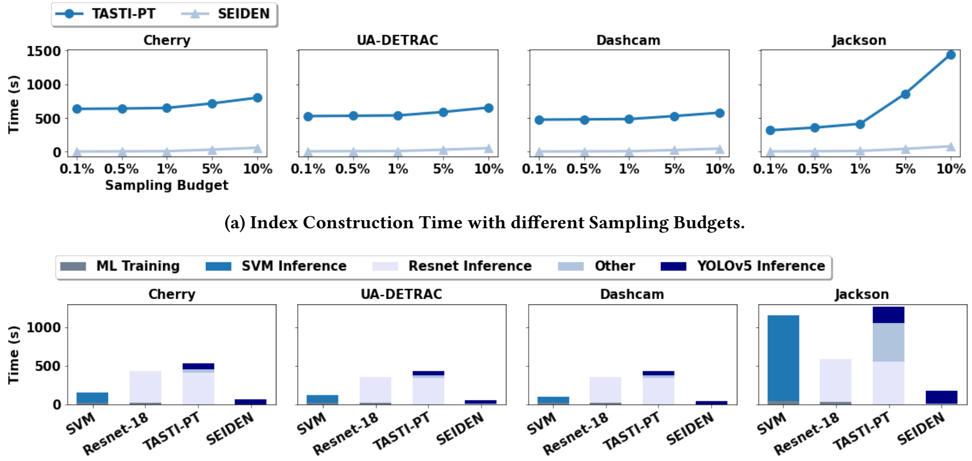
For a 960x540 image, YOLOv5 runs at 140 fps on one RTX 2080 Ti GPU in this server. For a 300x300 image, YOLOv5 runs at 1000 fps.

6.2 RQ1. Index Construction Time

In this experiment, we examine the time taken to build an index across all the systems on all the video datasets. Index construction is a one-time cost for a given video that is amortized across all subsequent queries over that video. We next discuss how each system constructs the index:

Seiden. During index construction, Seiden selects anchors from the I-frames set. For each selected frame, Seiden runs the oracle model on that frame and stores the oracle label in the index. Seiden spends nearly all the index construction time on inference.

tasti-pt. It first extracts the features of *all* the frames using the Resnet-18 model. tasti-pt then selects the anchor frames using the FPF clustering algorithm over all the extracted features. The complexity of the FPF algorithm is $O(n^2)$. tasti-pt uses multi-threading to accelerate the clustering step. Lastly, it caches the oracle labels for the chosen anchor frames.



(a) Index Construction Time with different Sampling Budgets.

(b) Index Construction Time Breakdown.

Figure 5: Index Construction Time for Seiden and baselines.

SVM. It is trained using random subset of frames (1%) in the video. The oracle model (*i.e.*, Yolov5s) is applied on these frames to get labels which are then used to train the SVM (*e.g.*, number of cars in a given frame). After the model training step, the SVM is applied on all the frames and results are cached. SVM utilizes a quadratic kernel (radial basis function [7]) for optimization that is slow on longer videos.

Resnet-18. It is trained in a similar manner to SVM. After the model training step, the Resnet-18 feature extractor is applied on all the frames and the features are later used during query execution. Ratio α is set to 0.8 for this experiment.

The results are shown in Figure 5. Figure 5a illustrates the change in index construction time of Seiden and tasti-pt as we vary the number of anchor frames (*i.e.*, percentage of all the video frames selected as anchors). Figure 5b presents the index construction time of all the systems when we configure the sampling budget (N) to be 10% of the video frames.

Analysis. The most notable observation from the Figure 5a is that the index construction in Seiden is on average 87X faster than tasti-pt. This is because tasti-pt applies the feature extractor on all the frames of the video to derive the anchor frames. As we increase the percentage of anchors, the increase in index construction time is linear in Seiden. This is because Seiden only needs to run the oracle model on the additional anchor frames. However, with tasti-pt, the increase in index construction time is super-linear as it runs the Resnet model on all the frames regardless of the sampling budget. For example, on Cherry, Seiden is 422× faster than tasti-pt with 0.1% sampling budget, whereas it is only 13× faster with 10% budget.

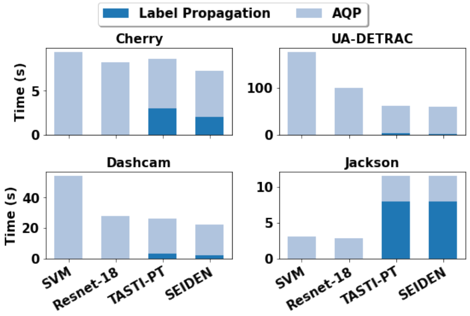


Figure 6: Query Execution Time – Aggregate Query.

In Figure 5b, we present a breakdown of time spent on different tasks during index construction. As Yolov5s is comparable in latency to Resnet-18 (Figure 2), the time spent on running the Resnet-18 model on all the frames is 5.7x higher than running the Yolov5s oracle model on the anchor frames (10% of the frames) Unlike Seiden and tasti-pt, SVM and Resnet-18 models are not query-agnostic. So, they have a non-trivial model training overhead (20 seconds on 1 K frames).

SVM is more lightweight than Resnet-18. So, the time taken to run the ML model on all the frames is 2.7x smaller than that for the Resnet-18 model. tasti-pt takes more time than Resnet-18 (1.23x) because, in addition to extracting the feature vectors using Resnet-18, it performs other operations like clustering to build a query-agnostic index. Seiden is 4× faster than the SVM baseline on the index construction task.

6.3 RQ2. Aggregate Query Execution

We next examine the time taken by different systems while executing an aggregate query on different datasets (Table 5). The results are shown in Figure 6. We configure the confidence score ($1) and error bound ($2) to 0.95 and 0.1, respectively.

For Seiden and tasti-pt, we breakdown the execution time into two components: (1) label propagation, and (2) AQP (Approximate Query Processing) time using the blazeit aggregate optimizer. Query execution time implicitly captures the accuracy of the proxy model. This is because the blazeit optimizer converges quickly when the proxy scores are similar to the oracle scores. Specifically, it invokes the oracle model on a random subset of frames to determine if the propagated label meets the user's accuracy constraints. So, proxy functions that result in fewer oracle model invocations lead to faster query execution.

With SVM and Resnet-18 baselines, there is no label propagation step. The output of both proxy models is an array of proxy scores (*i.e.*, aggregates) for all the frames in the video. The blazeit optimizer confirms whether the approximate aggregate meets the confidence score and error bound. It keeps invoking the model until it meets the accuracy constraints. For these two baselines, we use the model trained during the index construction phase. Since the proxy scores are already obtained for all the frames during index construction, the query execution time only captures the time spent on oracle model invocations with the blazeit optimizer.

Analysis. The most notable observation is that on Cherry, UA-DETRAC, and Dashcam datasets, Seiden is the fastest system. On these datasets, Seiden is on average 2.4×, 1.4×, 1.08× faster than SVM, Resnet-18, and tasti-pt, respectively. The performance gap between Seiden and SVM is highest on the UA-DETRAC dataset. This is because correctly computing the aggregate count of cars is more difficult in this dataset due to its high object selectivity (94%). Many frames in the video contain more than five cars. So, with the SVM, the blazeit optimizer makes more oracle invocations.

On all the datasets, Resnet-18 is faster than SVM. Both of these systems do not require label propagation. We attribute this gap to fewer oracle invocations by the blazeit optimizer with Resnet-18. This is because Resnet-18 provides more accurate proxy scores that better capture the contents of the frame.

Resnet-18 outperforms tasti-pt on Cherry and Jackson datasets. This is because the feature extractor is tailored for this particular query (*i.e.*, aggregate count of cars). In contrast, tasti-pt's index is query agnostic and does not require a training step for each query. Even though Seiden does not utilize compressed networks for feature extraction or inference, it is either similar to or slightly faster than tasti-pt. This shows the efficacy of leveraging temporal continuity for generating proxy scores.

On the Jackson dataset, SVM and Resnet-18 baselines are faster than tasti-pt and Seiden. We attribute this to the overhead of the label propagation step. All of the systems have similar AQP time with the blazeit optimizer (5× smaller than that on the UA-DETRAC aggregate query). The label propagation time is more significant on this dataset due to the size of the video, as it linearly increases with the total number of frames in the video. On more difficult datasets, like UA-DETRAC, the label propagation time is not as significant as the post-processing time.

6.4 RQ3. Retrieval Query Execution

We next examine the F1-score of all the systems on retrieval queries (*i.e.*, precision and recall queries in Table 5). The results for these queries are shown in Figure 7a and Figure 7b, respectively. We configure the precision ($3) and recall bounds ($4) to 0.95.

To ensure that the query result meets the user's accuracy constraints, we use the supg optimizer. With the supg optimizer, the user must configure the maximum oracle invocation budget. We configure this budget to 10% of the number of video frames. Within this oracle invocation constraint, the supg optimizer utilizes the generated proxy scores to determine the best set of frames to return to the user. We measure the accuracy of the results relative to the oracle model. On these queries, all the systems return an array of proxy scores. For each frame, they return a proxy score that lies between 0 and 1. For instance, 0.9 signifies that the frame is very likely to satisfy the query's predicate (*i.e.*, COUNT(CAR) > 0).

SVM and Resnet-18. We utilize the same model used for aggregate queries. Instead of generating an aggregate label, it returns a confidence score that is used to derive the proxy score. For example, if the confidence score for a given frame containing 0 cars is 0.7, then for the query retrieving frames with at least one car, the proxy score is 0.3.

tasti-pt and Seiden. Query execution consists of label propagation and using the supg optimizer for AQP. The proxy system used for aggregate queries is used for retrieval queries as well.

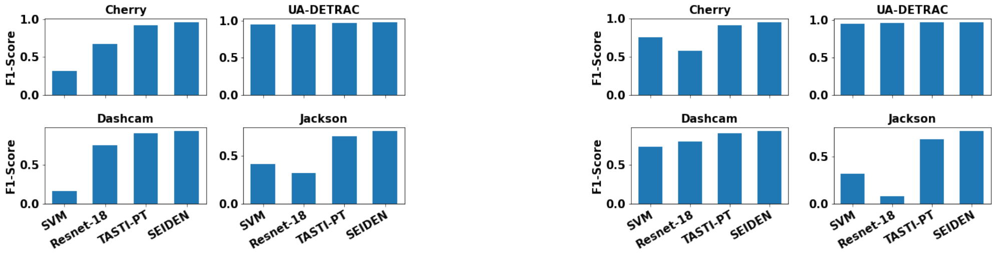
Analysis. As shown in Figure 7a, on the precision query across all the datasets, Seiden delivers a higher F1-score compared to SVM and Resnet-18, and slightly higher F1-scores to tasti-pt. The F1-score of Seiden is on average 95%, 34%, and 3% higher than SVM, Resnet-18, and tasti-pt, respectively.

Similar to the results on aggregate queries, Resnet-18 delivers more accurate proxy scores than SVM. The F1-score of all the systems on UA-DETRAC is high because of the high object selectivity of the dataset. So, it is possible to easily pick a set of frames that contain at least one car.

On the Dashcam dataset, SVM has a low F1-score (0.17). This is because of the complexity of videos collected from a moving camera. Due to the moving background, the SVM model is unable to effectively identify the car objects in the foreground. Without relying on convolutional layers, it is challenging to generate accurate proxy scores on this dataset. On the recall queries, as shown in Figure 7b, tasti-pt and Seiden deliver the most accurate results. F1-score of Seiden on recall queries is on average 32%, 52%, and 4% higher than SVM, Resnet-18, and tasti-pt, respectively.

6.5 RQ4. Impact of Anchor Sampling Ratio (α)

In this experiment, we examine the impact of the anchor sampling ratio (α) on the F1-score. This hyper-parameter determines how Seiden utilizes a given anchor budget. Specifically, α determines how many anchors are selected during index construction and query execution, respectively. We do this analysis with queries Q1 (aggregate query) and Q2 (precision query) on Cherry and Dashcam datasets, that are representative of other query-dataset pairs. The results are shown in Figure 8. For each query-dataset pair, we vary the sampling budget from 1% to 10% of the frames. If the dataset contains 100 K frames, this maps to a budget from 1 K to 10 K frames. For each sampling budget, we vary α from 20% to 100%. For example, if the dataset contains 100 K frames, the sampling budget is 1%, and α is 20%, then 200 frames will be selected as anchors during index construction. The remaining 800 frames will be picked during query execution. If α is 100%, then all the frames within the budget are picked during the index construction step itself. As the index construction step primarily picks I-frames spread across the entire video, it favors exploration. So, a higher value of α favors exploration. Lower values of α favor exploitation based on label difference. In each row in the heat map, the values are normalized for the given sampling budget. For aggregate query Q1, we measure query execution time. For precision query Q2, we measure the



(a) Precision Queries.

(b) Recall Queries.

Figure 7: Accuracy of VDBMSs on Precision and Recall Queries.

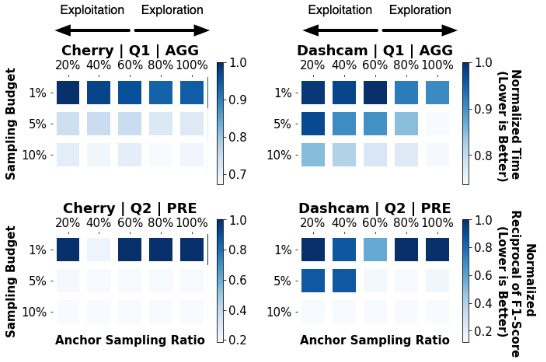


Figure 8: Impact of Anchor Sampling Ratio on Execution Time (Q1) and F1-score (Q2). Lower values (*i.e.*, lighter colors) are better.

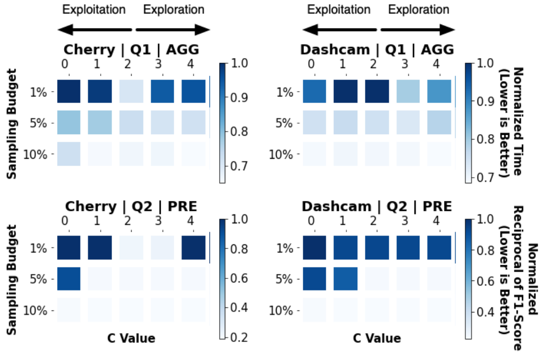


Figure 9: Impact of MAB Sampling Parameter on Query Execution Time (Q1) and F1-Score (Q2). Lower values are better.

reciprocal of the F1-score. So, lower values (*i.e.*, lighter colors) in the heat map are better.

In Q1, for both datasets, the optimal value of α is greater than 80% across all sampling budgets. This indicates that exploration is more important than the exploitation of aggregate queries. On Q2, for a given time constraint, the SUPG optimizer tries to maximize the F1-score. Here, the optimal values of α are between 40% and 60% which indicates the need for balancing between exploration and exploitation. This behavior is observed on Cherry | Q2 | 1% budget and Dashcam | Q2 | 1% and 5%.

On certain dataset-budget combinations, α does not have a noticeable impact on accuracy. This behavior is observed on Cherry | Q2 | 5% budget and Dashcam | Q2 | 10%. We attribute this to the increased sampling budget. With too many selected samples, regardless of whether the sample was picked during index construction or query execution, the SUPG algorithm generates accurate results.

Thus, for aggregate queries (Q1), SEIDEN tunes α to favor exploration. It allocates a significant sampling budget to the INDEXCONSTRUCTION phase (*i.e.*, α = 0.8). For precision queries (Q2), it strikes a balance between exploration and exploitation by splitting the budget evenly across the INDEXCONSTRUCTION and the QUERYEXECUTION phases (*i.e.*, α = 0.5).

6.6 RQ5. Impact of MAB Sampling Parameter (c)

The MAB sampling algorithm used in SEIDEN relies on c hyperparameter to determine the balance between exploitation and exploration Equation 3. Specifically, when c = 0, the sampling algorithm only performs exploitation. With higher values of c, it prefers exploration over exploitation. We consider the same query-dataset pairs used in §6.4. Besides varying the budget, we vary c from 0 to 4. The results are shown in Figure 9. The optimal configuration of c varies based on the query. With Q1, the optimal value of c ranges from 3 to 4. So, the aggregate query benefits from more exploration (similar to the trend in Figure 8). With Q2, the optimal value of c ranges from 2 to 3. So, the precision query benefits from a balance between exploration and exploitation.

The impact of c is not prominent on Cherry | Q2 | 10% and Dashcam | Q2 | 10%. This is because, with a high sampling budget (10%), the SUPG optimizer is able to return highly accurate results – 0.95 for Cherry | Q2 | 10% and 0.93 for Dashcam | Q2 | 10%, irrespective of whether the samples were picked using an exploration or exploitation scheme. This behavior is also observed on Dashcam | Q2 | 1%. In this case, even with a smaller sampling budget, F1-scores across all values of c are low (0.22).

Thus, SEIDEN picks a high c value to favor exploration on aggregate queries (Q1) (*i.e.*, c = 4). For precision queries, it balances exploration and exploitation by setting c = 2.

6.7 RQ6. Oracle Label Reuse Across Queries

In this experiment, we examine the ability of SEIDEN to reuse the results of the oracle model across multiple queries, similar to TASTI-PT. Consider a sequence of two precision queries based on CAR and BUS, respectively. To answer the second query, SEIDEN reuses all the available oracle labels from the first query (collected during both INDEXCONSTRUCTION and QUERYEXECUTION phases) and samples additional frames based on the second query's predicate

Table 6: Oracle Label Reuse Across Queries – F1-Scores of TASTI-PT and SEIDEN with reuse.

Dataset | Query	Cherry | Q2		Dashcam | Q2	
Object	CAR	BUS	CAR	BUS
TASTI-PT	0.92	0.63	0.91	0.48
SEIDEN	**0.95**	**0.68**	**0.93**	**0.62**

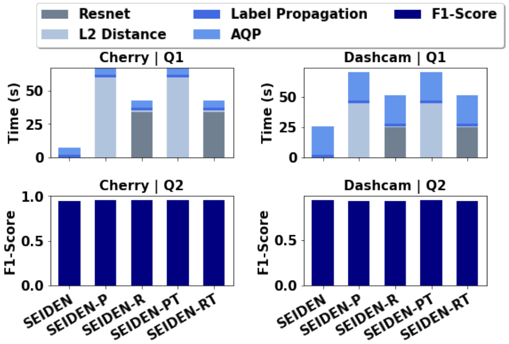


Figure 10: Impact of Label Interpolation Scheme on Execution Time (Q1) and F1-Score (Q2). Lower values are better.

(*i.e.*, BUS). Note that there is no INDEXCONSTRUCTION phase while answering the second query on the same video. SEIDEN samples additional frames based on label difference with respect to the BUS predicate and dedicates the entire frame budget to the QUERYEXECUTION phase. Table 6 presents the F1-scores. On the first query (Q2 | CAR), the F1-score of SEIDEN is slightly higher than that of TASTI-PT as it picks a better set of samples. On the second query (Q2 | BUS), SEIDEN 's F1-score is 8% and 29% higher than TASTI-PT on the Cherry and Dashcam datasets, respectively. The accuracy gap is more prominent because SEIDEN reuses all the oracle labels from the first query to guide the sampling for second query.

6.8 RQ7. Impact of Label Interpolation Scheme

We next investigate the impact of other label interpolation schemes on query execution time and accuracy. We consider four schemes: (1) pixel distance, (2) RESNET-18 feature distance, (3) pixel and temporal distance (hybrid), and (4) RESNET-18 feature and temporal distance (hybrid). Pixel distance metric (SEIDEN-P) computes the L2 norm between the pixel values of the unlabeled frame and the two temporally closest sampled frames. It then interpolates the unlabeled frame's label using the labels of two sampled frames weighted by the L2 norm. RESNET-18 metric (SEIDEN-R) is based on L2 norm between the RESNET-18 features. Pixel and temporal distance (SEIDEN-PT) normalizes the pixel and temporal distances respectively, and equally weighs and combines these distance metrics. Similarly, SEIDEN-RT is based on RESNET-18 feature and temporal distance. For SEIDEN-P and -PT, we downsample the frames to 100×100 resolution to limit computation time. For SEIDEN-R and -RT, we downsample the frames to 100×100 before using RESNET-18 to extract features. We consider the query-dataset pairs used in §6.4.

The results are shown in Figure 10. For label propagation and AQP, all interpolation techniques require comparable times on Cherry | Q1 and Dashcam | Q1. However, SEIDEN-P and -PT require substantial time (60 s for Cherry and 45 s for Dashcam) to compute the L2 distance, while SEIDEN-R and SEIDEN-RT require significant time (34 s for Cherry and 25 s for Dashcam) to extract RESNET-18 features. Unlike these schemes, interpolation based on temporal distance in SEIDEN does not require expensive computation for label propagation. Another notable observation is that the impact of the label interpolation scheme is not significant on the F1-score, with the maximum difference in F1-score being less than 0.02. Thus, temporal linear interpolation is both fast and accurate for a broad range of queries and datasets.

7 RELATED WORK

VIDEO DBMSs. Researchers have presented several techniques for accelerating retrieval queries over videos based on: lightweight proxy models like SVM [31], model cascades [3, 24], and specialized models [21]. SEIDEN challenges the assumption made in these systems that the oracle model is significantly slower than the proxy model, and presents a simpler and efficient architecture for leveraging the temporal continuity of videos.

INDEXES IN VISUAL DBMSs. Systems for multimedia content retrieval [4, 12, 28] have long studied the problem of indexing visual data. They construct indexes on the high-dimensional feature space to accelerate nearest-neighbor and similarity-based retrieval queries. Among VDBMSs, Voodoo [18], Tasti [22], and Panorama [44] use general-purpose feature embeddings to index similar video frames and leverage the index to answer queries. SEIDEN shows that for videos, general-purpose feature embeddings do not offer a significant advantage over leveraging temporal continuity.

AQP. AQP techniques fall under two categories: offline and online methods [27]. Offline methods use synopses like pre-computed samples [1], histograms [34], sketches [5], wavelets [15] or recent ML-driven methods [13, 32, 38] to approximately answer queries. Online methods [20, 35] do not assume apriori knowledge of the workload. They compute the query statistics on the fly using sampling techniques. In VDBMSs, AQP cannot precompute synopses because, unlike structural data, which aggregates available records, VDBMSs aggregate the output of UDFs, which are expensive to compute. So, VDBMSs primarily rely on techniques similar to online sampling methods.

8 CONCLUSION

In this paper, we revisited query processing in VDBMSs given the radical shift in the performance gap between oracle and proxy models over the last few years. We presented SEIDEN, a VDBMS that simply leverages the lightweight oracle model and the temporal continuity of the video to support faster query processing than SoTA VDBMSs. Our empirical evaluation shows that SEIDEN works well across three types of queries on five real-world datasets, and is on average 6.6× faster than SoTA VDBMSs while delivering comparable query accuracy.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation (IIS-1908984, IIS-2238431), Cisco, Adobe, Alibaba Innovative Research (AIR) Program, and Intel.

REFERENCES

[1] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*. 29–42.

[2] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.

[3] Michael R Anderson, Michael Cafarella, German Ros, and Thomas F Wenisch. 2019. Physical representation-based predicate optimization for a visual analytics database. In *ICDE*. IEEE, 1466–1477.

[4] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. 2014. Neural codes for image retrieval. In *European conference on computer vision*. Springer, 584–599.

[5] Vladimir Braverman and Rafail Ostrovsky. 2013. Generalizing the layering method of indyk and woodruff: Recursive sketches for frequency-based vectors on streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 58–70.

[6] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* 5, 1 (2012), 1–122.

[7] Martin D Buhmann. 2000. Radial basis functions. *Acta numerica* 9 (2000), 1–38.

[8] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[9] 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*. 559–572.

[10] Pramod Chunduri, Jaeho Bang, Yao Lu, and Joy Arulraj. 2022. Zeus: Efficiently Localizing Actions in Videos Using Reinforcement Learning. In *SIGMOD*. 545–558.

[11] Maureen Daum, Brandon Haynes, Dong He, Amrita Mazumdar, Magdalena Balazinska, and Alvin Cheung. 2020. TASM: A Tile-Based Storage Manager for Video Analytics. *arXiv preprint arXiv:2006.02958* (2020).

[12] Jia Deng, Alexander C Berg, and Li Fei-Fei. 2011. Hierarchical semantic indexing for large scale image retrieval. In *CVPR 2011*. IEEE, 785–792.

[13] Shaddy Garg, Subrata Mitra, Tong Yu, Yash Gadhia, and Arjun Kashettiwar. 2023. Reinforced Approximate Exploratory Data Analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[14] Teofilo F Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *Theoretical computer science* 38 (1985), 293–306.

[15] Sudipto Guha and Boulos Harb. 2005. Wavelet synopsis for data streams: minimizing non-euclidean error. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 88–97.

[16] Brandon Haynes, Amrita Mazumdar, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. 2018. Lightdb: A dbms for virtual reality video. *VLDB* 11, 10 (2018).

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

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

[19] Wenjia He, Michael R Anderson, Maxwell Strome, and Michael Cafarella. 2020. A method for optimizing opaque filter queries. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1257–1272.

[20] Joseph M Hellerstein, Peter J Haas, and Helen J Wang. 1997. Online aggregation. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*. 171–182.

[21] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. 2018. Focus: Querying large video datasets with low latency and low cost. In *USENIX*. 269–286.

[22] Daniel Kang. 2023. TASTI. https://github.com/ddkang/tasti

[23] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv preprint arXiv:1805.01046* (2018).

[24] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. Noscope: optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529* (2017).

[25] Daniel Kang, Edward Gan, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Approximate selection with guarantees using proxies. *arXiv preprint arXiv:2004.00827* (2020).

[26] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Task-agnostic Indexes for Deep Learning-based Queries over Unstructured Data. *arXiv preprint arXiv:2009.04540* (2020).

[27] Kaiyu Li and Guoliang Li. 2018. Approximate query processing: What is new and where to go? *Data Science and Engineering* 3, 4 (2018), 379–397.

[28] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. 2015. Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 27–35.

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

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

[31] 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*. 1493–1508.

[32] Qingzhi Ma and Peter Triantafillou. 2019. Dbest: Revisiting approximate query processing engines with machine learning models. In *Proceedings of the 2019 International Conference on Management of Data*. 1553–1570.

[33] Seattle Department of Transportation. 2023. Travelers Home Page. https://web6.seattle.gov/travelers/

[34] Gregory Piatetsky-Shapiro and Charles Connell. 1984. Accurate estimation of the number of tuples satisfying a condition. *ACM Sigmod Record* 14, 2 (1984), 256–276.

[35] Chengjie Qin and Florin Rusu. 2014. PF-OLA: a high-performance framework for parallel online aggregation. *Distributed and Parallel Databases* 32, 3 (2014), 337–375.

[36] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *CVPR*. 779–788.

[37] Iain E Richardson. 2011. *The H. 264 advanced video compression standard*. John Wiley & Sons.

[38] Nikhil Sheoran, Subrata Mitra, Vibhor Porwal, Siddharth Ghetia, Jatin Varshney, Tung Mai, Anup Rao, and Vikas Maddukuri. 2022. Conditional Generative Model based Predicate-Aware Query Approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 8259–8266.

[39] Mingxing Tan, Ruoming Pang, and Quoc V Le. 2020. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10781–10790.

[40] Ultralytics. 2023. YOLOv5. https://github.com/ultralytics/yolov5

[41] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. 2020. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *CVIU* 193 (2020), 102907.

[42] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. 2019. Vstore: A data store for analytics on large videos. In *EuroSys*. 1–17.

[43] Zhuangdi Xu, Gaurav Tarlok Kakkar, Joy Arulraj, and Umakishore Ramachandran. 2022. EVA: A Symbolic Approach to Accelerating Exploratory Video Analytics with Materialized Views. In *SIGMOD*. 602–616.

[44] Yuhao Zhang and Arun Kumar. 2019. Panorama: a data system for unbounded vocabulary querying over video. *Proceedings of the VLDB Endowment* 13, 4 (2019), 477–491.