Finding Interesting Frames in Deep Video Analytics: a Top-K Approach

Ziliang Lai[1] Chenxia Han[1] Chris Liu[1] Pengfei Zhang[1] Eric Lo[1] Ben Kao[2]
[1]Chinese University of Hong Kong [2]University of Hong Kong
{zllai, cxhan, cyliu, pfzhang, ericlo}@cse.cuhk.edu.hk kao@cs.hku.hk

ABSTRACT

Recently, the impressive accuracy of deep neural networks (DNNs) has created great demands on practical analytics over video data. Although efficient and accurate, the latest video analytics systems have not supported analytics beyond selection and aggregation yet. In data analytics, Top-K is a very important analytical operation that enables analysts to focus on the most important entities. In this paper, we present EVEREST, the first system that supports efficient and accurate Top-K video analytics. EVEREST can help rank and identify the most interesting frames/moments from videos with theoretical accuracy guarantees. EVEREST is built on a careful synthesis of machine learning, computer vision, and Top-K query processing. Evaluations on six real-world video streams and the latest Visual Road benchmark show that EVEREST can achieve between 10.8× to 17.9× higher efficiency than baseline approaches with high accuracy.

PVLDB Reference Format:
. A Sample Proceedings of the VLDB Endowment Paper in LaTeX Format. *PVLDB*, 12(xxx): xxxx-yyyy, 2019.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

Cameras are ubiquitous. Billions of them are being deployed in public (e.g., at road junctions) and private (e.g., in retail stores) all over the world [38]. Recent advances in deep convolutional neural networks (CNNs) have led to incredible leaps in their accuracies in many machine learning tasks, notably image and video analysis. Such developments have created great demands on practical analytics over video data [31, 25, 36, 5, 51]. For instance, civil engineers are using traffic cameras in intelligent transportation systems for road planning, traffic incident detection, and vehicle re-routing in public transit. An example in which urban planners had to go through a year's worth of traffic video to retrieve all video frames that contain more than five buses is reported in [28]. Tools for fast and accurate video analytics are essential in smart city applications.

Proceedings of the VLDB Endowment, Vol. 12, No. xxx
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

Object recognition in image and video using CNNs are accurate [20, 19, 49]. However, CNNs are computationally expensive to train and to use. This poses significant challenge in using CNNs-based objection recognition techniques to process massive amounts of streaming videos. For example, a state-of-the-art object detector, Mask R-CNN [7], runs at 5 fps (frames per second) using GPUs. This frame-processing rate is 6 times slower than the frame rate of a typical 30-fps video. For the example where a year's worth of video is to be analyzed, a "scan-and-test" approach that invokes the object detector on every frame would take 6 years to complete. For video analytics in widespread use, the research community has started to build systems with innovative solutions to support fast analytics over video data [29, 4, 21, 28, 39, 27].

Video analytics is an emerging research area that intersects database and computer vision. It is, however, still in its infancy – even the latest system BlazeIt [28] supports only simple filter and aggregate queries. In data analytics, Top-K is a very important analytical operation that enables analysts to focus on the most important entities in the data [8, 24, 6, 42]. In this paper we present the very first set of solutions for efficient Top-K video analytics. Top-K can help rank and identify the most interesting frames/moments from videos. Example use cases include:

Property Valuation. The valuation/rent of a shop is strongly related to its peak foot traffic [10]. Instead of manual counting, one can use a camera to capture pedestrian flow and use a machine to identify, say, the top-10 frames (time of the day) with the highest pedestrian counts.

Transport Planning. Severe traffic congestion can occur when multiple bus routes pass through the same narrow expressway. This is because a bus stopping at a stop for passengers blocks the traffic (other buses) behind. The busiest bus creates a *convoy effect*, where a long line of other buses follow its lead. A Top-K query on the moments with the biggest convoys provide valuable information on the congestion caused and for better bus scheduling.

Data-Driven Agriculture. The global food demand is expected to increase by 70% from 2010 to 2050 [30]. In order to improve farm productivity, the FarmBeats project at Microsoft sends drones to collect and analyze farm videos [50]. Recent news also report that oil-palm farmers in Malaysia send drones to monitor the growth of oil-palm [41]. With oil-palm plantations spread across 86,100 square miles in Malaysia, farmers with limited resources can only inspect a small number of fields onsite each day. Finding Top-K fields (e.g., based on number of well-grown palm trees) over drone

videos can drastically help farmers prioritize field trips.

In video analytics, *CNN Specialization* and *Approximate Query Processing* (AQP) are two popular techniques for overcoming the inefficiency of the scan-and-test method [29, 4, 21, 28]. With CNN specialization, a video analytic system trains a lightweight model specifically designed for certain query type(s) (e.g., "*select* all frames with dogs") using frames sampled from the video-of-interest as training data. Since the lightweight model is highly specific to the query instance and the video-of-interest, it is both efficient to train and to use (but is generally less accurate). The lightweight model is then applied to all video frames to obtain a set of candidate frames that are likely to satisfy the given query. Finally, the candidate frames are passed to an accurate but less efficient object detector (e.g., YOLOv3 [45], Mask R-CNN [18]) to verify their validity. Approximate Query Processing (AQP) [28, 1] is an orthogonal technique for fast analytics when users are willing to tolerate some degree of error. The complex interplay between deep model inference and Top-K query processing, however, creates novel challenges for CNN specialization and AQP.

First, for CNN specialization, new lightweight models need to be designed for handling Top-K queries, which differ from traditional ones, such as selection. Second, traditional Top-K algorithms process objects with definite values. They are generally inapplicable to video analytics when lightweight models are used because these models' outputs are imprecise and probabilistic. Third, classic AQP techniques aim at determining confidence intervals for *statistical* aggregate queries (such as point estimates). The answers of Top-K queries, however, are set-based (e.g., sets of Top-K frames). Therefore, existing AQP algorithms are inapplicable. Fourth, video data has temporal locality; Top-K queries can be frame-based or *time-window-based*. For example, one could be interested in finding the Top-K 5-second clips with the highest number of vehicles. This adds to the complexity of answering Top-K queries in video analytics.

To address all these challenges, we present Everest, a system that empowers users to answer Top-K queries from video based on any given scoring (ranking) function. Everest is able to present results with *probabilistic guarantees*. Everest's probabilistic outputs play in unison with video analytics where data uncertainty is prevalent. This is in sharp contrast to existing systems where valuable information in uncertain data is not fully leveraged. To illustrate, Table 1a shows an example output of the lightweight model used in BlazeIt [28]. The *car count* in each traffic video frame is best given by a probability distribution, which captures the *uncertainty* information of the object detector. Existing systems, however, process queries based on a *simplified* view of the table where less likely cases are dropped and much of the uncertainty information is discarded (see Table 1b). Everest, in contrast, is designed to treat uncertain data as a first-class citizen. Query processing in Everest is probabilistic in nature. Furthermore, Everest is extensible to support windowing so that users can ask Top-K-windows queries as well.

Supporting Top-K analytics over videos enables rich analyses over the visual world, just as what traditional Top-K query processing has been doing over relational data. Everest draws heavily on *uncertain data management* techniques (e.g., [48, 3, 43]) but with its own novelty and uniqueness because of the *existence of ground-truth* at run-time.

timestamp /frame	count	prob.
f_1	0	0.78
	1	0.21
	2	0.01
f_2	0	0.49
	1	0.42
	2	0.09
f_3	0	0.16
	1	0.48
	2	0.36

(a) Output of a lightweight model

timestamp /frame	count
f_1	0
f_2	0
f_3	1

(b) Existing systems discard uncertainty information

Table 1: State-of-the-art

Specifically, uncertain query processing has been assuming no access to the ground-truth at run-time. By contrast, video analytics can access an accurate object detector to get the ground-truth and reduce the uncertainties at run-time. That essentially opens a new class of uncertain query processing problems in which the ground-truth is in the loop. We evaluate Everest on six real video streams and the latest Visual Road benchmark [17]. We show that Everest can achieve 10.8× to 17.9× speedup over the baseline approach. In summary, we make the following contributions:

1. We design and present a novel notion of Top-K analytics over videos. Our design is the first to treat uncertain data as a first-class citizen in video analytics.

2. We present Everest, the first video analytics system that supports Top-K queries with probabilistic guarantees. The system further includes extensions to find Top-K windows instead of Top-K frames.

3. We introduce efficient algorithms and implementations for each system module. Our algorithms overcome the combinatorial explosion in the number of possible worlds commonly found in uncertain query processing. Our implementation is highly optimized taking into account various practical efficiency issues such as numeric stability and GPU utilization.

The remainder of this paper is organized as follows. Section 2 provides background of our study. Section 3 defines Top-K queries supported by Everest. Sections 4 and 5 present Everest's algorithms and implementation. Section 6 gives evaluation results. Section 7 discusses related work. Finally, Section 8 concludes the paper.

2. BACKGROUND

In this section we provide brief background on object detection by CNN, video analytics systems, Top-K query processing, and uncertain query processing. Readers who are familiar with those topics may skip this section. We will further discuss their recent developments in Section 7.

CNN Object detection is an important problem in Computer Vision (CV). The problem is to identify object occurrences in images. Large volumes of training data have been made available nowadays, which enable modern techniques, especially *convolutional neural networks* (CNNs), to achieve near-human or even better-than-human accuracy [19]. However, these deep models are rather complex with millions

or even billions of parameters. Answering queries (or *inference*) with these models is thus computationally very expensive. For example, Mask R-CNN and YOLOv3, two state-of-the-art models, process video at respective rates of 5 fps and 30 fps on machines equipped with GPUs [7].

Timestamp (ts)	Class	Polygon	ObjectID	Content	Features
01-01-2019:23:05	Human	(10, 50), (30, 40), ...	16	...	...
01-01-2019:23:05	Bus	(45, 58), (66, 99), ...	58	...	...
01-01-2019:23:06	Human	(20, 80), (7, 55), ...	16	...	...
01-01-2019:23:06	Car	(6, 91), (10, 55), ...	59	...	...
01-01-2019:23:06	Car	(78, 91), (40, 55), ...	60	...	...
...	...	...	...	...	...

Table 2: A video relation fully populated by a ground-truth object detector

Video Analytics The huge volume of video data poses great challenges to video analytics research [29, 21, 40, 4, 55, 28]. For analytical processing, video data is often modeled as relations, each captures object occurrences in a specific video clip [28]. Specifically, each tuple in a video relation corresponds to a single object in a video frame. Since a frame may contain 0 or more objects (of interest) and an object may appear in multiple frames, a frame can be associated with 0 or more tuples in a relation and an object can be associated with multiple tuples. Typical attributes of a tuple include a frame timestamp (ts); a unique id of an identified object (objectID); the object's class label (class), bounding polygon (polygon), raw pixel content (content), and feature vector (features). A video relation can be *materialized* by invoking an object detector to process each frame to compute tuple values. Table 2 shows an example of a materialized video relation. The objectID attribute can be populated by invoking an object tracker (e.g, [56]), which takes as input two polygons from two consecutive frames and returns the same objectID if the two polygons enclose the same object. In video analytics, a video relation that is materialized by an accurate object detector such as YOLOv3 is regarded as the system's *ground-truth* [21]. However, fully materializing a ground-truth relation is computationally expensive. Therefore, the key challenge is how to answer analytic queries without a fully materialized ground-truth relation [29, 21, 40, 4, 55, 28]. This constraint distinguishes our work with previous video database studies [26, 9], which assume that video relations are *already given*, presumably via some external means (such as human annotations).

CNN specialization is a key technique used to speed up video analytics [47, 29, 28, 21]. Inspired by the concept of "cascade" [52] in computer vision, the idea is to use frames sampled from the video-of-interest to train a lightweight CNN (e.g., with fewer neurons and layers) as a proxy to an expensive ground-truth object detector (e.g., Mask R-CNN). Training specialized CNNs is significantly faster because there are fewer neural layers and a specific context. For example, training of a CNN with only frames from a traffic video converges much faster than in a general setting because there are far fewer object classes to consider (e.g., cars, bicycles, pedestrians), as opposed to a general object detector that has to distinguish thousands of classes. Since specialized CNNs are lightweight, their models are much faster to execute. For example, it has been reported that specialized lightweight CNNs can infer at 10,000 fps with various degree of accuracy loss [28]. However, a specialized CNN is data specific and thus not generalizable to other video. Also, lightweight CNNs may produce false positives or false negatives. As a remedy, systems like BlazeIt [28] use statistical techniques to bound result errors.

Top-K Query Processing In many application domains (e.g,. information retrieval [6], multimedia databases [15], and spatial databases [46]), users are more interested in the K most important tuples ordered by a scoring(ranking) function. The essence of efficient Top-K query processing is to wisely schedule data accesses such that most of the irrelevant (non-top-K) items are skipped. This minimizes expensive data accesses (e.g., from disk or via the web) and provides early stopping criteria to avoid unnecessary computation.

Uncertain Query Processing One common uncertain data representation is "*x-tuples*" [2]. An *uncertain relation* is a collection of x-tuples, each consists of a number of alternative outcomes that are associated with their corresponding probabilities. Together, the alternatives form a discrete probability distribution of the true outcome. Different x-tuples are assumed to be independent of each other. We discuss later how EVEREST uses a difference detector to approximate that so that the outputs of a lightweight CNN can be represented by the x-tuple model (i.e., the inference results of a video frame is captured by one x-tuple). Hence, in the following discussion, we use the terms *x-tuple*, *frame*, and *timestamp* interchangeably.

3. TOP-K IN EVEREST

EVEREST allows users to set a per-query threshold, `thres`, to ensure that the returned Top-K result has a minimum of `thres` probability to be the exact answer. Given an uncertain relation D (Section 4.1 discusses how to obtain that) and a scoring function S (e.g., count the number of cars in a frame f), EVEREST returns a Top-K result $\hat{R}$ with confidence $\hat{p} = \Pr(\hat{R} = R) \geq$ `thres`, where R is the exact result and `thres` is the probability threshold specified by the user. The probability $\hat{p}$ is defined over an uncertain relation D (e.g., Table 1a) using the *possible world semantic* (PWS) [37]. The possible world semantic is widely used in uncertain databases, where an uncertain relation is instantiated to multiple possible worlds, each of which is associated with a probability. Table 3 shows 2 possible worlds (out of 3^3) of Table 1a. Given the probability of each possible world, the confidence $\hat{p}$ of a Top-K answer $\hat{R}$ is simply the sum of probabilities of all possible worlds in which $\hat{R}$ is Top-K:

$$\hat{p} = \sum_{W \in \mathcal{W}(D) \wedge \hat{R} = \text{Top-K}(W)} \Pr(W). \quad (1)$$

Here, $\mathcal{W}(D)$ denotes the set of all possible worlds of an uncertain relation D. In addition, the answer $\hat{R}$ has to satisfy the following condition:

DEFINITION: *The* **Certain Result (CR) Condition**. *The Top-K result $\hat{R}$* has to be chosen from D^c*, where frames in D^c are all* **certain***, i.e., they have already been verified by a ground-truth object detector (GTOD) and with no uncertainty.*

The certain-result condition is important in video analytics. For instance, the Top-1 result $\hat{R} = \{f_3\}$ of the uncertain relation D in Table 1a has a confidence of $\hat{p} = 0.85$ (based

timestamp	num of cars
f_1	0
f_2	0
f_3	0

(a) $Pr(W_1) = 0.78 \times 0.49 \times 0.16$

timestamp	num of cars
f_1	1
f_2	0
f_3	0

(b) $Pr(W_2) = 0.21 \times 0.49 \times 0.16$

Table 3: Two possible worlds W_1 and W_2

timestamp	num of cars	conf.
f_1	0	0.78
	1	0.21
	2	0.01
f_2	0	0.49
	1	0.42
	2	0.09
f_3	0	1.0

Table 4: After $GTOD(f_3)$ on Table 1a

on Equation 1). Assuming `thres` $= 0.8$, the confidence of $\hat{R}$ being the correct answer is above the user threshold. However, this result is computed from the uncertain frames in D. Like other cases of uncertain query processing, the ground-truth is unknown during the computation. In video analytics, this is however undesirable because a user knows the ground-truth by visually inspecting returned frames in an answer to the query. In the example above, if the user sees no cars in the frame f_3, the returned Top-1 answer $\hat{R} = \{f_3\}$ is unacceptable. The certain result condition avoids such awkward answers and is essential in ensuring that all the tuples in $\hat{R}$ have been confirmed by a GTOD before an answer is returned. With that, $\{f_3\}$ would not be returned as the Top-1 result because its probability of being Top-1 is only 0.38 based on the *updated* uncertain relation D' in Table 4 and Equation 1, where f_3 is confirmed using a GTOD (we denote that operation as GTOD(f_3)). Finally, we remark that $\hat{p} = \Pr(\hat{R} = R) \geq$ `thres` guarantees not only the whole result set $\hat{R}$ has at least `thres` probability of being the true answer, but every frame $\hat{f}$ in $\hat{R}$ also has at least `thres` probability of being in the exact result set R because:

$$\Pr(\hat{f} \in R) \geq \Pr(\hat{R} = R) \geq \texttt{thres}$$

$\Pr(\hat{f} \in R)$ reflects the *precision* of Top-K answer, i.e., the fraction of results in $\hat{R}$ that belongs to R. Therefore, EVEREST effectively provides guarantees on the precision of the query answers.

4. EVEREST SYSTEM

Following recent works [4, 21, 29], EVEREST also focuses on a batch setting. In this setting, large quantities of video are collected for post-analysis. Online analytics on live video stream is a different setting and is beyond the scope of this paper. In addition, we focus on a bounded set of labels because state-of-the-art object detectors do so. For example, while the pre-trained YOLOv3 can detect cars, it cannot distinguish between sedan and hatchback. However, users can supply UDFs to do further classification if necessary. Supporting unbounded vocabulary is not impossible if our lightweight network also outputs embeddings like in [57]. But we leave it as our next step and we focus on Top-K query processing in this paper. To our knowledge, tracking model drift in visual data has not been well addressed in the literature. We will tackle that upon robust techniques for that are developed.

Figure 1 shows the system overview. EVEREST leverages CNN specialization and uncertain query processing to accelerate Top-K analytics with probabilistic guarantee. Processing a query involves two phases. The first phase aims to train a lightweight *deep convolutional mixture density network* (DCMDN) [14] that outputs a distribution about the scoring function (e.g., the number of cars in a frame), followed by sampling the DCMDN to build an initial uncertain relation D_0. The first phase can be done offline (i.e., ingestion time) for standard scoring functions (e.g., counting). The second phase takes as input the resulting uncertain relation D_0 from Phase 1 and finds a Top-K result $\hat{R}$ that has a confidence $\hat{p} \geq$ `thres`. Initially, given D_0 only, it is unlikely the initial Top-K result $\hat{R}_0$ from D_0 has confidence over the threshold. Furthermore, we see from the previous section that given a potential Top-K result $\hat{R}$, we have to confirm its frames for the CR condition using a GTOD — but that may conversely give the same $\hat{R}$ a lower confidence based on the updated uncertain relation D' (e.g., drops from 0.85 to 0.38). Of course, if the uncertain relation D' contains no more uncertainty (i.e., all tuples are certain), the Top-K result from that D' has a confidence of 1. Consequently, Phase 2 can be viewed as *Top-K processing via uncertain data cleaning*, in which we hope to selectively "clean" the uncertain tuples in the uncertain relation using a GTOD until the Top-K result from the latest uncertain relation satisfies the probabilistic guarantee. For high efficiency, Phase 2 aims to clean as few uncertain tuples as possible because each cleaning operation invokes the computational expensive GTOD. Furthermore, the algorithms in Phase 2 have to be carefully designed because uncertain query processing often introduces an exponential number of possible worlds.

4.1 Phase 1: Building the initial uncertain relation D_0

The crux of CNN specialization is the design of a lightweight model that is suitable for the specific query type. Prior works [29, 28] that focus on selection and aggregate queries build classifiers, where each distinct integer is treated as a standalone class label in the `softmax` layer. The `softmax` layer effectively generates a discrete distribution of number of cars for frames as shown in Table 1a. One drawback of building a classifier is that the maximum number of classes is restricted to be the maximum value found in the sample frames. This is undesirable because it will fail when the Top-1 frame is missed in the training frames. We therefore discard that approach. Instead, we train a lightweight deep convolutional mixture density network (DCMDN). Figure 2 shows the architecture of our lightweight DCMDN. It uses YOLOv3-tiny [45] (excluding the output layer) as the backbone to extract features from the input frame and uses a mixture density network (MDN) to output parameters of g Gaussians (means μ and variances σ) and their weights (π) in the mixture. Compared with typical regression models, the MDN layer can provide valuable uncertainty information (e.g., variance) in the output. In order to generate a high quality uncertain relation (i.e., high log-likelihood to the ground-truth), we follow a recent work[53] to build the MDN layer, where the neural network first generates a set of h hypotheses as possible predictions. By regarding the hypotheses as samples drawn from a mixed Gaussian

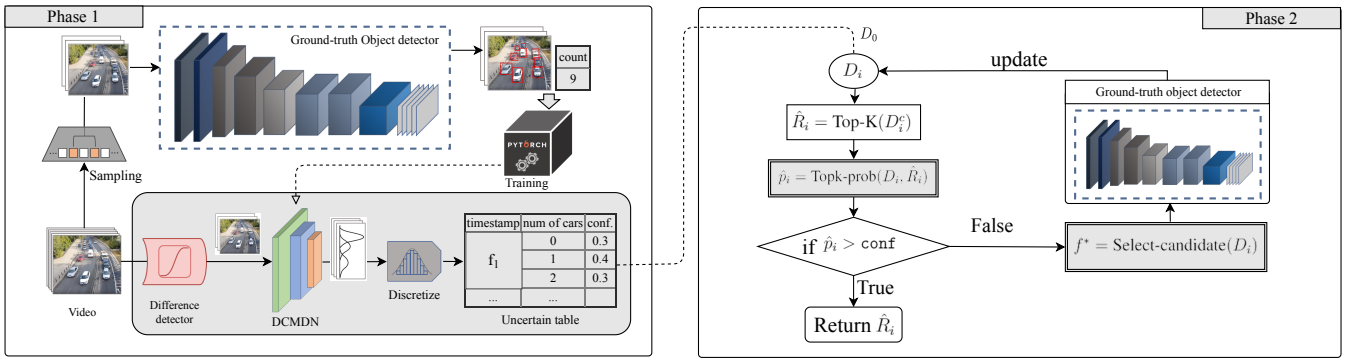


Figure 1: EVEREST System overview

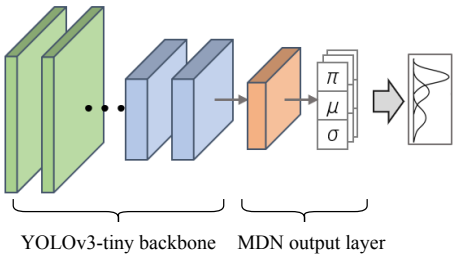


Figure 2: EVEREST's Neural Network

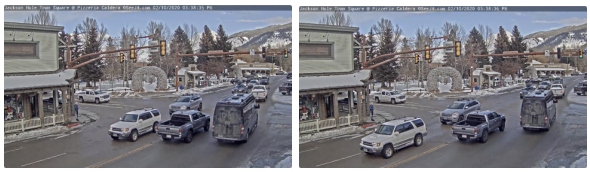


Figure 3: Similar frames are not informative

distribution, the parameters of the distribution is then estimated. This method is reported to overcome the limitations of traditional MDN including degenerated predictions and overfitting.

The training data of the DCMDN are frames randomly sampled from the video-of-interest with real scores (e.g, number of cars) obtained from a ground-truth object detector (GTOD). Since both the network and training data are small, the training quickly converges in several minutes. After the DCMDN is trained, EVEREST uses a difference detector to discard similar frames from the video. This step serves two purposes. First, frames with little differences are not informative (e.g., see Figure 3) in Top-K context. Second, it approximates independence among frames so as to enable the use of "x-tuples" to model the data. After that, EVEREST feeds the unique frames to the DCMDN to obtain the score distribution for each frame. Finally, EVEREST draws samples from the DCMDN to build the uncertain relation. An x-tuple captures a discrete distribution but the Gaussian mixture is continuous and with infinitely long tails on both ends. In order to get a finite uncertain relation, we follow [13] to replace the Gaussians with truncated Gaussians so that the probabilities beyond 3σ are set to zero and evenly distributed to the rest. After that, we populate the uncertain relation D_0 by discretizing the truncated Gaussian mixture. For counting based scoring function, the score distribution is discretized to non-negative integers. For others, users have to provide the discretization step size when defining the scoring function. For frames that are already labeled by the GTOD when training the DCMDN, their scores are known and **certain**. They are inserted into the uncertain relation straight with no uncertainty.[1]

[1]Nevertheless, we still call that as an "uncertain relation".

4.2 Phase 2: Top-K processing via uncertain data cleaning

Figure 1 (right) outlines the flow of Phase 2. Starting from an uncertain relation D_0 given by Phase 1, it iteratively selects the best frame f^* (by the Select-candidate function) to clean using the GTOD until a Top-K result $\hat{R}_i$ based on the latest table D_i has a confidence $\hat{p}_i$ (computed by the Topk-prob function) exceeds the user threshold `thres`.

Finding a Top-K result $\hat{R}_i$ from the latest updated table D_i (by the Top-K(D_i^c) function) is straightforward because it simply extracts all the **certain** tuples D_i^c (because of the CR condition) from D_i and applies an existing Top-K algorithm (e.g., [8]) to find the Top-K as $\hat{R}_i$. The function Topk-prob that computes the probability $\hat{p}_i$ for $\hat{R}_i$ being the exact result and the function Select-candidate that selects the most promising frame are more challenging because they involve an exponential number of possible worlds. Note that the techniques used in traditional uncertain data cleaning [35, 12] are not applicable here because of the differences in the problem setting. In those cleaning problems, they have a constraint on the "budget", which is the number of x-tuples that could be cleaned; and their objective is to minimize the entropy of all possible Top-K results from the updated table. Since their setting is to return a batch of x-tuples for a person to manually carry out a one-off cleaning operation, using entropy allows them to measure the answer quality solely based on the uncertain data as the results of the cleaning operation are not available at run-time. By contrast, our constraint is to pass the probability threshold `thres` and our objective is to minimize the cost of using the GTOD (and the algorithm overhead). For us, the GTOD provides online feedback, allowing us to put that into the equation to come up with a probabilistic guarantee (between 0 and 1), which is way more intuitive than using entropy (which could be 0 to ∞).

D	Uncertain relation
D_i	Uncertain relation at iteration i
D_i^c	Subset of D_i whose x-tuples are all certain
D_i^u	Subset of D_i whose x-tuples are all uncertain
$\hat{R}$	Approximate Top-K result
$\hat{p}$	Confidence/probability of $\hat{R}$
$\hat{R}_i, \hat{p}_i$	$\hat{R}$, $\hat{p}$ obtained in i-th iteration
R	Actual result
f	Frame / x-tuple
GTOD	Ground-truth object detector
S_f	Score of f
$\Bbbk_i$	The frame ranked K-th in $\hat{R}_i$
$\mathbb{p}_i$	The frame ranked penultimately in $\hat{R}_i$

Table 5: Major Notations

In the following, we discuss efficient algorithms to implement the Topk-prob and the Select-candidate functions. Table 5 gives a summary of the major notations we used.

4.2.1 *Implementing* Topk-prob$(D_i, \hat{R}_i)$

Given a potential result $\hat{R}_i$ extracted from D_i^c, computing its confidence $\hat{p}_i$ via Equation 1 has to expand all $O(m^n)$ possible worlds, where n is the number of frames in D and assume each of them has m possible scores in the uncertain relation. However, given the Certain Result (CR) condition, we can simplify the calculation of $\hat{p}_i$ as:

$$\hat{p}_i = \prod_{f \in D_i^u} \Pr(S_f \leq S_{\Bbbk_i}) \tag{2}$$

where (a) S stands for the score of a frame, (b) $\Bbbk_i$ is the "threshold" frame that ranks K-th in $\hat{R}_i$, and its score $S_{\Bbbk_i}$ is known and certain (because $\Bbbk_i$ is from $\hat{R}_i \subseteq D_i^c$), and (c) D_i^u are frames in D_i with uncertainty, i.e., $D_i^u = D_i \setminus D_i^c$.

Computing Equation 2 requires only time linear to $|D_i^u|$. Equations 1 and 2 are equivalent because the probability of $\hat{R}_i$ being Top-K is equal to the probability that no frames in D_i^u having scores larger than the frames in $\hat{R}_i$.[2]

A further optimization is to compute two functions before Phase 2 begins: (a) the CDF F for the score of each frame f, i.e., $F_f(t) = \Pr(S_f \leq t) = \sum_{j=0}^{t} \Pr(S_f = t)$ and (b) a function $H(t) = \prod_{f \in D_0^u} F_f(t)$, which is the joint CDF of all uncertain frames in D_0. With them, in the i-th iteration, we can compute $\hat{p}_i$ as follows:

$$\hat{p}_i = \frac{H(S_{\Bbbk_i})}{\prod_{f \in D_i^c} F_f(S_{\Bbbk_i})} \tag{3}$$

Equations 2 and 3 are equivalent because by definition $D_i^u = D_0^u \setminus D_i^c$. $F_f(t)$ and $H(t)$ for all f and t can be easily pre-computed one-off at the cost of $O(|D_0^u|)$. With Equation 3, Topk-prob$(D_i, \hat{R}_i)$ in the i-th iteration can compute $\hat{p}$ using $O(|D_i^c|)$ time instead, where $|D_i^c| << |D_i|$.

As a remark, $\hat{p}_i$ actually improves exponentially with the number of frames cleaned according to Equation 2. Therefore, we expect Phase 2 would spend more iterations to reach a small probability threshold, say 0.5. But after that, it would take fewer iterations to reach any probability threshold beyond.

[2]We allow frames in D_i^u to have scores tie with the threshold frame $\Bbbk_i$.

4.2.2 *Implementing* Select-candidate(D_i^u)

Select-candidate(D_i^u) is the function to select a frame f^* from the set of uncertain frames D_i^u in the i-th iteration to apply the GTOD such that cleaning f^* can maximize $\hat{p}_{i+1}$ of the next iteration; and hopefully $\hat{p}_{i+1} \geq$ `thres` after that and thus Phase 2 can stop early.

Of course, $\hat{p}_{i+1}$ is unknown before we apply GTOD(f^*). Therefore, we use a random variable X_f to denote the value of $\hat{p}_{i+1}$ after a frame f is cleaned. To maximize $\hat{p}_{i+1}$, we aim to find $f^* = \arg\max_{f \in D_i^u} E_i[X_f]$. Using x_f^s to denote the value of X_f when $S_f = s$, where s is a particular score, $E[X_f]$ is thus:

$$E[X_f] = \sum_s \Pr(S_f = s) x_f^s \tag{4}$$

[Efficient computation of x_f^s] x_f^s is the probability of the result $\hat{R}_{i+1}$ being the Top-K based on D_{i+1}, where the x-tuple representing f in D_{i+1} is assumed to be cleaned and its score is s and certain. Therefore, x_f^s can be calculated on top of p_i (Equation 3) by removing the uncertainty of f, based on how the actual score s of f influences the Top-K result:

$$x_f^s = \begin{cases} \dfrac{H(S_{\Bbbk_i})}{F_f(S_{\Bbbk_i}) \prod_{f' \in D_i^c} F_{f'}(S_{\Bbbk_i})} = \dfrac{\hat{p}_i}{F_f(S_{\Bbbk_i})} & s \leq S_{\Bbbk_i} \\ \dfrac{H(s)}{F_f(s) \prod_{f' \in D_i^c} F_{f'}(s)} & S_{\Bbbk_i} < s \leq S_{\mathbb{p}_i} \\ \dfrac{H(S_{\mathbb{p}_i})}{F_f(S_{\mathbb{p}_i}) \prod_{f' \in D_i^c} F_{f'}(S_{\mathbb{p}_i})} & s > S_{\mathbb{p}_i} \end{cases} \tag{5}$$

The idea of Equation 5 is that:

- when $s \leq S_{\Bbbk_i}$, frame f is not qualified to be in Top-K; the Top-K result would not change, and the threshold score is still $S_{\Bbbk_i}$; So, discounting the uncertainty of f suffices.
- when $S_{\Bbbk_i} < s \leq S_{\mathbb{p}_i}$, frame f enters the Top-K and but its score is lower than the penultimate frame $\mathbb{p}_i$ in the Top-K result, i.e., the one ranks $(K-1)$-st, so frame f gets the K-th rank; the new "threshold" score is changed to s;
- when $s > S_{\mathbb{p}_i}$, frame f enters the Top-K with a score greater than the original penultimate frame, the new threshold frame is $\mathbb{p}_i$, the new threshold score is changed to $S_{\mathbb{p}_i}$.

Putting Equations 5 and 4 together, we get:

$$\begin{aligned} E[X_f] = \hat{p}_i + & \sum_{s=S_{\Bbbk_i}+1}^{S_{\mathbb{p}_i}} \frac{\Pr(S_f = s) H(s)}{F_f(s) \prod_{f' \in D_i^c} F_{f'}(s)} \\ & + \frac{(1 - F_f(S_{\mathbb{p}_i})) H(S_{\mathbb{p}_i})}{F_f(S_{\mathbb{p}_i}) \prod_{f' \in D_i^c} F_{f'}(S_{\mathbb{p}_i})} \end{aligned} \tag{6}$$

Equation 6 greatly reduces the cost of computing $E[X_f]$ compared to Equation 4 because the summation sums only

over the range from $S_{\mathbb{k}_i}+1$ to $S_{\mathbb{p}_i}$.

[Finding f^* by early stopping] To find f^* in an iteration i, a baseline implementation of Select-candidate(D_i^u) has to compute $E_i[X_f]$ *for every frame* f in D_i^u. It is inefficient because D_i^u is large.

Fortunately, we can deduce an upper bound, $U(X_f)$, for each $E[X_f]$ and process frames in descending order of $U(X_f)$ to early stop the computation of f^*. Specifically, starting from Equation 6, we have:

$$\begin{aligned}
E[X_f] &= \hat{p}_i + \sum_{s=S_{\mathbb{k}_i}+1}^{S_{\mathbb{p}_i}} \frac{\Pr(S_f=s)H(s)}{F_f(s)\prod_{f'\in D_i^c} F_{f'}(s)} \\
&\quad + \frac{(1-F_f(S_{\mathbb{p}_i}))H(S_{\mathbb{p}_i})}{F_f(S_{\mathbb{p}_i})\prod_{f'\in D_i^c} F_{f'}(S_{\mathbb{p}_i})} \\
&\leq \hat{p}_i + \sum_{s=S_{\mathbb{k}_i}+1}^{S_{\mathbb{p}_i}} \frac{\Pr(S_f=s)H(S_{\mathbb{p}_i})}{F_f(S_{\mathbb{p}_i})\prod_{f'\in D_i^c} F_{f'}(S_{\mathbb{p}_i})} \\
&\quad + \frac{(1-F_f(S_{\mathbb{p}_i}))H(S_{\mathbb{p}_i})}{F_f(S_{\mathbb{p}_i})\prod_{f'\in D_i^c} F_{f'}(S_{\mathbb{p}_i})} \\
&= \hat{p}_i + \frac{(1-F_f(S_{\mathbb{k}_i}))H(S_{\mathbb{p}_i})}{F_f(S_{\mathbb{p}_i})\prod_{f'\in D_i^c} F_{f'}(S_{\mathbb{p}_i})} \\
&= \hat{p}_i + \gamma\psi_i(f) = U(X_f) \qquad (7)
\end{aligned}$$

where the last line factors the terms into: $\gamma = \frac{H(S_{\mathbb{p}_i})}{\prod_{f'\in D_i^c} F_{f'}(S_{\mathbb{p}_i})}$ and $\psi_i(f) = \frac{1-F_f(S_{\mathbb{k}_i})}{F_f(S_{\mathbb{p}_i})}$.

In the i-th iteration, the order of frames' upper bound is only determined by the "sort-factor" $\psi_i(f)$ because the frames share the same $\hat{p}_i$ and γ. This suggests Select-candidate(D_i^u) to examine the frames in the descending order of their $\psi_i(f)$. When it examines a frame f^- whose $U(X_{f^-})$ is smaller than any examined frame f_{seen}'s $E[X_{f_{seen}}]$, Select-candidate(D_i^u) would stop early and return f^* from those that have been examined. However, since $S_{\mathbb{k}_i}$ (and thus $U(X_f)$) changes with i, we might have to re-compute and re-sort the frames per iteration. To avoid this overhead, we further re-write Equation 7 to:

$$E[X_f] \leq \hat{p}_i + \gamma\psi_j(f) \qquad (8)$$

where $j \leq i$. The inequality still holds because $S_{\mathbb{k}_j} \leq S_{\mathbb{k}_i}$ and $S_{\mathbb{p}_j} \leq S_{\mathbb{p}_i}$.

With Equation 8, we can simply set $j=0$ so that we only need to compute $\psi_j(f)$ and sort frames in the first iteration. However, to balance between tighter bounds and efficiency, in the first 100 iterations, we set $j = \lfloor \frac{i}{10} \rfloor$, i.e., we update $\psi_j(f)$ and sort frames every 10 iterations. For iterations thereafter, we update $\psi_j(f)$ whenever $S_{\mathbb{k}_i}$ or $S_{\mathbb{p}_i}$ change. The idea is that $S_{\mathbb{k}_i}$ and $S_{\mathbb{p}_i}$ change more in early iterations but are relatively stable afterwards.

4.3 Top-K Windows

Videos are spatial-temporal in nature and thus users may want to split a video into *time windows* of finite size, compute their scores, and examine the Top-K ones. For example, an urban planner may be interested in the Top-50 30-second windows, where the score of a window is the average number of cars of its frames.

EVEREST supports Top-K over *tumbling windows* like the example given above. Specifically, the video-of-interest is divided into consecutive non-overlapping time windows w_1, w_2,...,w_n, each of which contains L frames. The score of a window w, denoted by S_w, is the average of the scores of the frames in it, i.e., $S_w = \frac{1}{L}\sum_{f\in w} S_f$. For Top-K-window queries, we find the Top-K windows $\hat{R}$ such that $\hat{p} = \Pr(\hat{R} = R) \geq$ `thres`, where R is the set of true Top-K windows.

To support this type of query, EVEREST builds another uncertain table whose schema is akin to Table 1a: `(window, avg(num of cars), prob)`. Let the i-th frame in w be f^i. The distribution of S_w can be calculated based on the the distributions of $S_{f^1}, S_{f^2}, \ldots, S_{f^L}$. From Section 4.1, the distribution of S_{f^i}, as obtained from the DCMDN, is a g-component Gaussian mixture with a density of:

$$\sum_{j=1}^{g} \pi_{ij}\mathcal{N}(x;\mu_{ij},\sigma_{ij}^2)$$

where π_{ij}, μ_{ij}, σ_{ij} are the weight, mean and variance of the j-th component in the mixture distribution of S_{f^i}, respectively. The difference detector effectively divides the window into l segments where the frames in the same segment are *similar* to a *reference frame*. We thus approximate the distributions of frames of the same segment by the distribution of the segment's reference frame. Moreover, reference frames of different segments are judged as sufficiently dissimilar by the difference detector and hence we assume their distributions are independent. Let r_1, r_2, ..., r_l denote the l reference frames in w, we approximate the distribution of S_w by

$$S_w \sim \mathcal{N}(x; \frac{1}{L}\sum_{i=1}^{l}|s_i|\bar{\mu}_{r_i}, \frac{1}{L}\sum_{i=1}^{l}|s_i|\bar{\sigma}^2_{r_i}) \qquad (9)$$

where $\bar{\mu}_{r_i}$ and $\bar{\sigma}_{r_i}$ are the mean and the total variance of S_{r_i}. Suppose r_i is the t-th frame in w, then

$$\bar{\mu}_{r_i} = \sum_{j=1}^{g}\pi_{tj}\mu_{tj},$$

$$\bar{\sigma}^2_{r_i} = \sum_{j=1}^{g}\pi_{tj}(\sigma^2_{tj}+\mu^2_{tj}-\bar{\mu}^2_{r_i})$$

By discretizing the distribution in Equation 9, EVEREST obtains an uncertain table to support tumbling windows on mean. That uncertain table is compatible with the algorithms in Phase 2. When confirming a window using the GTOD (to compute the average score of a window) during Phase 2, a large window size (e.g., 1 min) may require cleaning a lot of frames. Therefore, we calculate the window average by sampling the frames.

5. IMPLEMENTATION

EVEREST is implemented in Python 3.7 with Numpy 1.17. We use a Pytorch implementation of YOLOv3 [45] as the ground-truth object detector, whose weights are pretrained using the COCO dataset[32] with 416×416 image resolution. Training and inference of the DCMDN are implemented by Pytorch 1.4.

DCMDN Training. The sampled frames are resized to 128×128 resolution and pixel values are normalized to the range from 0 to 1. As mentioned, our DCMDN uses the CNN layers of YOLOv3-tiny as the backbone to extract image features. To allow faster convergence, we use a YOLOv3-tiny pretrained by Darknet [44] using the COCO dataset, and then train it to detect the object-of-interest in the first three epochs. Afterwards, we discard the output layer of YOLOv3-tiny and connect its backbone to the MDN layer to form our DCMDN. The DCMDN is then further trained for seven more epochs using the sampled frames. Overall, the whole lightweight DCMDN training converges in ten epochs (around two minutes).

Difference Detector. Various difference detectors can be used in EVEREST. In the current implementation, we follow [29] and use mean-square-error (MSE) among pixels to measure the difference between two frames. To eliminate similar consecutive frames, in principle we need to sequentially scan through the video and discard a frame if its MSE with the last retained frame is lower than a threshold. In order to parallelize this step, we split the video into clips of c frames each. Each frame in a clip is compared with the middle one in the clip (i.e., the $\lfloor \frac{2}{c} \rfloor$-th frame) and is discarded if their MSE is lower than a threshold. The clips are then processed in parallel. Although similar frames at the boundaries of clips may be retained or two adjacent clips may be too similar, c can be adjusted to ensure these situations are rare.

Numeric Stability. A naive implementation of Equation 3 is not numerically stable as it involves many multiplications and divisions on small floating-point values and the results might be too small to be representable by 64-bit floating-point numbers. Therefore, we keep all the probability values in log-scale and convert multiplication and division into addition and subtraction, respectively. Computing Equation 6 is also not numerically stable for the same reason. In addition, it involves additions which cannot be computed in log-scale. Directly taking `exp` on each log-scaled term and summing them up still suffers numeric instability because the values after `exp` are also too small. We resolve this by using `log_sum_exp` function implemented in Scipy library that internally shifts each term before taking `exp` and shifts back after taking the `log` of summation.

Batch Inference. The Select-candidate function in Phase 2 selects the most promising frame and infers its ground-truth score using the ground-truth object detector in each iteration. However, selecting and confirming only one frame at a time might not fully utilize the abundant GPU processing power. Therefore, our implementation selects a batch of b frames that have the highest expectations based on Equation 6 and carries out batch inference. The value of b depends on the FLOPS and the memory bandwidth of GPU. Although the GPU is better utilized for larger b, setting b too large may cause cleaning unnecessary frames. Therefore, we choose b based on a measurement of inference latency to ensure that the latency of cleaning b frames has no significant difference from cleaning one frame.

Prefetching. Deep network inference may have I/O overheads when fetching the frames from the disk to the main memory and thus stalling the GPU. The baseline scan-and-test approach can alleviate that easily by prefetching the frames because it accesses them sequentially. Our Top-K algorithm, however, selects frames to clean, which is non-sequential. Fortunately, EVEREST can achieve high throughput by also prefetching the input frames based on the sort-order of ψ_j in Equation 8. Therefore, batches of frames with the highest ψ_j would be pre-fetched while the GPU is carrying out computation.

6. EVALUATION

We performed our experiments on an Intel i9-7900X server with 64GB RAM and one NVIDIA GTX1080Ti GPU. The server runs CentOS 7.0.

Evaluation Metrics. For each experiment, we report (a) the end-to-end query runtime (including both Phase 1 and Phase 2), and/or the speed-up compared with an exact scan-and-test method that uses YOLOv3, and the result quality in terms of (b) the *precision* (the fraction of results in $\hat{R}$ that belongs to R)[3], (c) the *rank distance* (the normalized footrule distance between the ranks of the $\hat{R}$ and their true ranks in R), and (d) the *score error* (the average absolute error for scores between $\hat{R}$ and R). We have carried out six sets of experiments:

- An overall evaluation result using the default setting (Section 6.1).
- Evaluating the impact of K (Section 6.2).
- Evaluating the impact of the query's probability threshold `thres` (Section 6.3).
- Studying the impact when changing to a complex scoring function (Section 6.4).
- Evaluating the Top-K window feature (Section 6.5).
- Evaluating the impact of the number of objects appeared in a video (Section 6.6).

Queries and Datasets. We evaluate EVEREST on six real video in which four of them are also used in prior work and we add one static camera video (`Grand Targhee`[4]) and one moving camera video (`Irish-Center`[5]) collected from Youtube. Table 6 shows the characteristics of the real datasets. In addition, we also include synthetic datasets generated by the latest Visual Road benchmark[17]. The synthetic datasets are used in the last experiment (Section 6.6) to evaluate the impact of the number of objects appeared in the video since we cannot control the number of objects in real videos.

Except Experiment 4 that evaluates EVEREST using a scoring function with multiple objects (Section 6.4), we evaluate EVEREST using Top-K queries where frames are ranked by the number of main object-of-interest shown in Table 6. The main object-of-interest varies by the video content. For example, `Archie` and `Taipei-bus` are traffic footages so that their main objects are cars. By default, K=50 and `thres`=0.9.

System configurations. In Phase 1, we randomly sample 5000 frames as the training set for the lightweight DCMDN. The beauty of our DCMDN is that it is general. We are

[3] The *recall* (the fraction of results in R that were covered by $\hat{R}$) is the same as the precision because both R and $\hat{R}$ contain K elements.

[4] https://www.youtube.com/watch?v=NNBptRtWMDc

[5] https://www.youtube.com/watch?v=MXqKk4WEhsE

able to use the same number of hypothesis (h=20) and the same number of Gaussians (g=10) for all datasets. Although the MSE threshold in the different detector can be tuned based on the speed of the moving objects or the speed of the moving camera, we are also able to use a unified MSE threshold of 0.001 for all real data. For Visual Road data, we use a smaller MSE threshold of 0.0001 because the synthetic data contains no noise. The same clip size ($c = 30$) of the difference detector is set for both real and synthetic videos. In Phase 2, we set the batch inference size b to be 8 after measuring the inference latency on our server.

6.1 An overall result

Our first experiment aims to give an overall evaluation of EVEREST. We run the default Top-50 query (`thres`=0.9) on all the real datasets.

Figure 4 shows the evaluation results. The variation of the speedup on different datasets could be attributed to many factors such as the video quality as well as the distributions of the object-of-interests. Nonetheless, we are able to observe a significant speedup of 10.8× to 17.9×. The result quality of EVEREST is also excellent. The query results have precisions all over 0.9 (90%), which are coherent with the 0.9 probability threshold of the queries. More specifically, all queries get results of very small rank distance, which indicates that EVEREST returns almost perfect result with an order of speedup. That can be further evidenced by observing that the score errors are all less than 0.2.

Table 7 shows a breakdown of the end-to-end query runtime. As expected, a large fraction of time is spent on populating the initial uncertain table because of the high volumes of frames processed by the DCMDN. The steps of labeling the sample frames and training the DCMDN are relatively fast because of the small training set and the lightweight network design, respectively. In fact, Phase 1 can be done offline while ingesting the data into the system [21] or even at the edge where the videos were produced [23]. But we make no such assumption here in this paper.

Similarly, Phase 2 spent most time on confirming the frames using the GTOD. Nonetheless, that is almost minimal because the fraction of frames being cleaned is very small (1.46%–5.25%). The algorithmic overheads are also minimal (the time percentage of functions Top-K(D_i^c) and Topk-prob(D_i, $\hat{R}_i$) are not shown because they are all less than 0.001%), indicating that our algorithmic optimizations are very effective.

6.2 Impact of K

The objective of this second experiment is to understand the impact of K on EVEREST. We run Top-K queries on all real datasets with different K values: 5, 10, 25, 50, 75, and 100.

Figure 5 shows that EVEREST offers better speedup when K is small. That is because a smaller K results in a smaller result set, and thus the "threshold" frame tends to have a higher score (i.e., a higher $S_{\mathbb{k}_i}$). That in turns implies a higher $\hat{p}_i$ based on Equation 2 and so that it can stop earlier by reaching `thres` easier.

While the accuracy remains high for different values of K, we observe that small K values tend to influence the precision more. That is natural because the precision is a fraction influenced by the result size. For example, the precision of EVEREST drops below 90% on `Grand Targhee` even though EVEREST missed only one frame out of the Top-5 result, resulting in a 80% precision. Nonetheless, we observe that the result quality is actually high from the other two quality metrics when K is small. The rank distance conversely increases when K increases because of a similar reason. For large K values, missing one frame in rank i-th would cause a rank difference for all the frames after it. Therefore, if we look at the result accuracy using all three quality metrics, we can see that EVEREST produces high quality results across different values of K.

6.3 Impact of `thres`

The objective of this third experiment is to understand the impact of the probability threshold on EVEREST. We run Top-50 queries on all real datasets with different `thres` values: 0.5, 0.75, 0.9, 0.95, and 0.99.

Figure 6 shows that the value of `thres` is not crucial as long as it is over 0.5. This is expected because we mentioned that $\hat{p}_i$ actually improves exponentially with the number of frames cleaned according to Equation 2. In our experiments, it took 99% of iterations to reach a probability threshold of 0.5 but only 1% of iterations to reach 0.99. This is indeed a nice result because it confirms that EVEREST can hide this parameter to users in real deployment.

6.4 Complex Scoring Function

The objective of this fourth experiment is to demonstrate that EVEREST is able to handle general user-defined scoring functions other than object counting. We therefore run Top-50 queries with the following scoring function:

$$S_f = \frac{\texttt{area(cars) + area(buses)}}{\texttt{total area}}$$

This scoring function is continuous and involves two objects-of-interest. The areas of cars and buses are referring to the pixel areas of their bounding boxes reported by the ground-truth object detector. The total area is the area of the image. This scoring function is a simplified metric to evaluate the degree of congestion of the road. We run this experiment with discretization step size set to 0.01 on `Taipei-bus`, `Archie` and `Irish-Center` because only these videos contain both types of objects.

Figure 7 shows that EVEREST maintains high result quality with that scoring function. The result quality of EVEREST on `Archie` is slightly dropped when compared with the results of using the standard counting function. We manually inspected the reason and found that the GTOD (YOLOv3) could perform accurate *object detection* on `Archie` but its *bounding box detection* was unstable (e.g., it returns bounding boxes of different sizes for the same input randomly). As the scoring function in this experiment relies on the bounding box areas whereas the previous experiments rely only on the object labels, that explains the drop in the result quality.

In Figure 7 we observe a minor drop in speedup when with compared using the standard counting function. That is actually reasonable because a more complex scoring function is more challenging to the DCMDN, causing its output distribution has a higher variance. That causes EVEREST to clean more frames in order to pass the probability threshold.

6.5 Top-K Windows

Video (Used in / From)	Object-of-interest	Resolutions	FPS	# of frames	Length (hrs)
`Grand Canal` ([29, 28])	boat	1920×1080	60	2324k	10.7
`Coral` ([29])	person	1280×720	30	1844k	17.1
`Archie` ([28])	car	1920×1080	30	2130k	19.7
`Grand Targhee`	person	1920×1080	30	814k	7.5
`Irish-Center`	car	1920×1080	30	1569k	14.5
`Taipei-bus` ([29, 28])	car	1920×1080	30	1600k	14.8

Table 6: Real Dataset Characteristics

Dataset	Phase 1			Phase 2			
	Label sample frames by GTOD	DCMDN training	Populate D_0 by DCMDN inference	Num of iterations	Select-candidate	Confirm frames using GTOD	% of frames cleaned
`Grand Canal`	3.85%	7.18%	61.04%	4245	0.27%	27.58%	1.46%
`Coral`	3.52%	6.47%	43.62%	6890	5.39%	40.91%	2.99%
`Archie`	2.89%	4.26%	42.29%	9098	6.17%	44.31%	3.41%
`Grand Targhee`	10.29%	14.67%	42.23%	1836	0.86%	31.87%	1.81%
`Irish-Center`	5.07%	6.39%	47.38%	4541	2.25%	38.82%	2.31%
`Taipei-bus`	3.39%	4.11%	32.09%	10491	0.42%	59.89%	5.25%

Table 7: A Detailed Performance Breakdown

The objective of this fifth experiment is to evaluate the window feature of EVEREST. We run Top-50 window queries with different window sizes: no window (i.e., a window of 1 frame), 1 second, 10 seconds, 30 seconds, 1 minute, and 3 minutes.[6] The probability threshold is still 0.9. In Phase 2, each window samples 10% of its frames to infer the ground-truth.

Figure 5 shows that EVEREST performs better or as good as frame-based Top-K when the window sizes are small (<10s). For 1s window, EVEREST performs even better because we only sample frames in the window to infer the ground-truth, thereby reducing the GTOD overhead. But that advantage is offset when the window size gets larger (>10s). First, a larger window essentially reduces the number of windows. As shown in the experiments above, our Top-K algorithm is smart at picking the most promising frame out of millions of frames to clean. A reduced number of windows would, however, reduce the number of choices EVEREST has. For example, a 10-hour video and a window size of 3 minutes give EVEREST only 200 windows to pick. This factor, however, is not crucial in practice as real deployments usually analyze longer videos (e.g., months of surveillance videos). We however cannot afford experiments of that scale in this paper. Second, a larger window implies more frames have to be confirmed by the GTOD per selected candidate (compared with only one frame has to be confirmed per selected candidate when no window is specified). Nonetheless, the various factors start to reach an equilibrium when the window sizes are larger than 30 seconds.

In terms of accuracy, the result quality improves with the size of window because of the Gaussian approximation in Equation 9 is more accurate for larger window size.

6.6 Impact of the number of objects

In this last experiment, we aim to evaluate EVEREST based on the number of objects in a video. Since we cannot control that in real videos, we generate five 30 fps videos using the Visual Road benchmark [17], each of which is ten hours long in 416×416 resolution. All five synthetic videos share the same setting except the total number of cars. Specifically, the videos are all taken by the same "camera" shooting from the same "angle" of the same "mini-city". The only moving objects in the videos are cars and we control the total number of cars in that mini-city from 50 to 250. During the data generation process, we uncovered a problem in the Visual Road benchmark in which we could only stably generate at most 15 minutes long video. After discussing with the authors of Visual Road, we concatenated 40 clips of 15-minute videos to form each ten-hour video.

Figure 9 shows the evaluation result on Visual Road under the default Top-50 query. We observe a speedup of 15× to 22× with excellent accuracy. This experiment indicates that the good performance of EVEREST would not be affected by the object density in the videos.

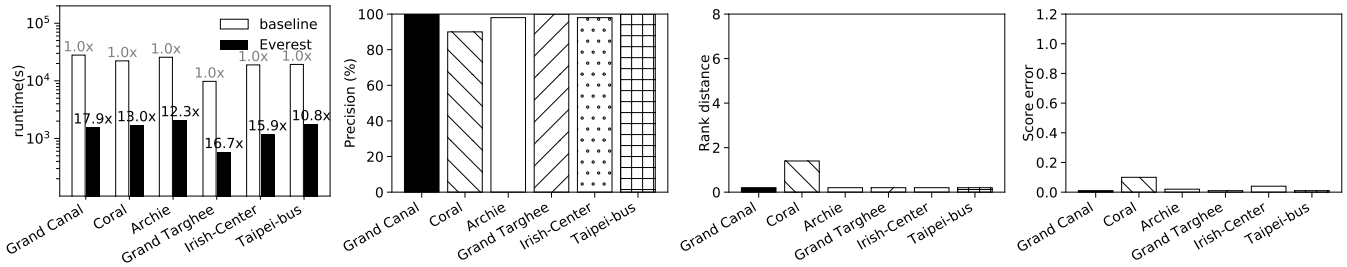
7. RELATED WORK

EVEREST builds upon research in modern video analytics and uncertain Top-K query processing. While we have covered some of them throughout the paper, the following is a more comprehensive discussion.

Modern Video Analytics. Video analytics span across a number of system fields. Optasia [33] is a system that allows users to express video analytic queries as dataflow graphs. VideoStorm [55] also allow users to express distributed analytical workloads over video as graphs and focuses on automatic tuning of knobs like resolution and frame rate to maximize the output quality based on the computing resource availability in a cluster. Chameleon [27] is a system similar to VideoStorm but it tunes the knobs at runtime. Scanner [40] is a system that takes as input a graph that connects various pixel processing operations and schedules them onto heterogeneous computing hardware, such as CPUs, GPUs, and media processing ASICs.

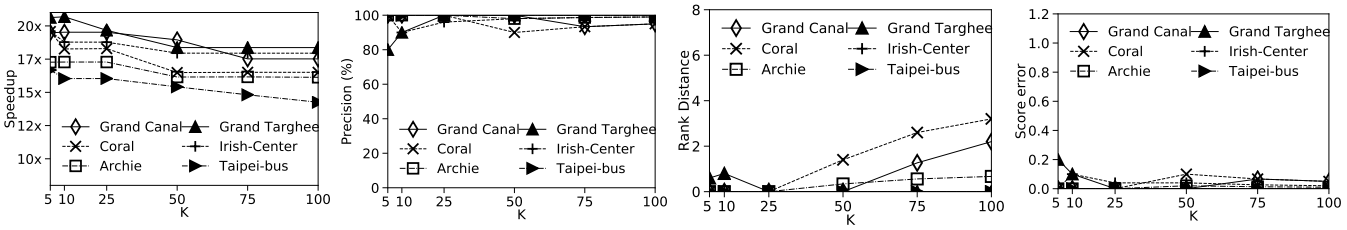
Noscope [29] is a system based on CNN specialization and has an optimizer to select the best model architecture (e.g., number of layers) that maximizes the throughput subject to

[6]Based on 30 fps video. The window size for 60 fps `Grand Canal` is scaled accordingly.



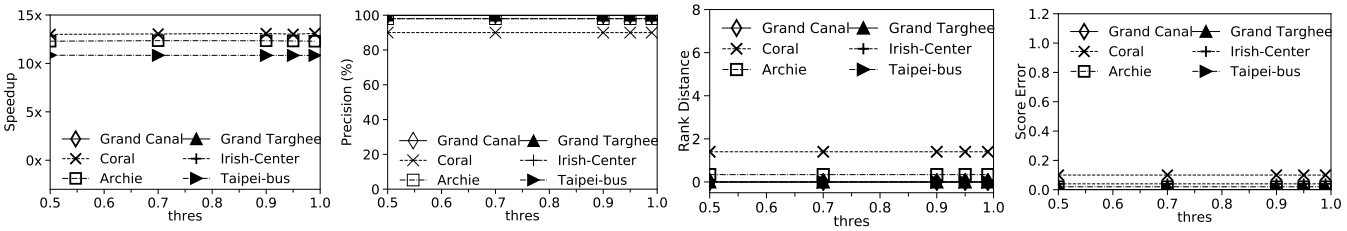
(a) Speedup (b) Precision (c) Rank distance (d) Score error

Figure 4: Overall result under the default setting



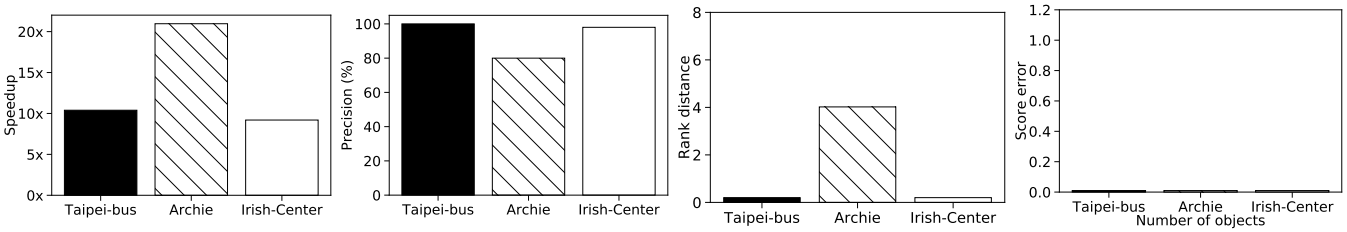
(a) Speedup (b) Precision (c) Rank distance (d) Score error

Figure 5: Impact of K



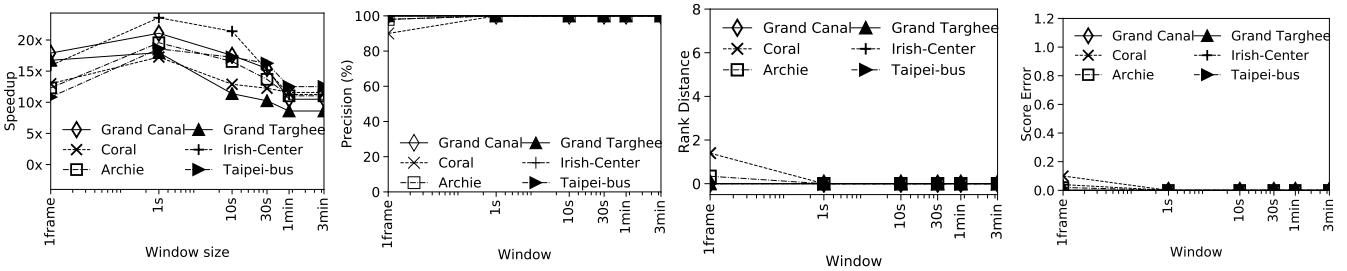
(a) Speedup (b) Precision (c) Rank distance (d) Score error

Figure 6: Impact of the confidence threshold `thres`



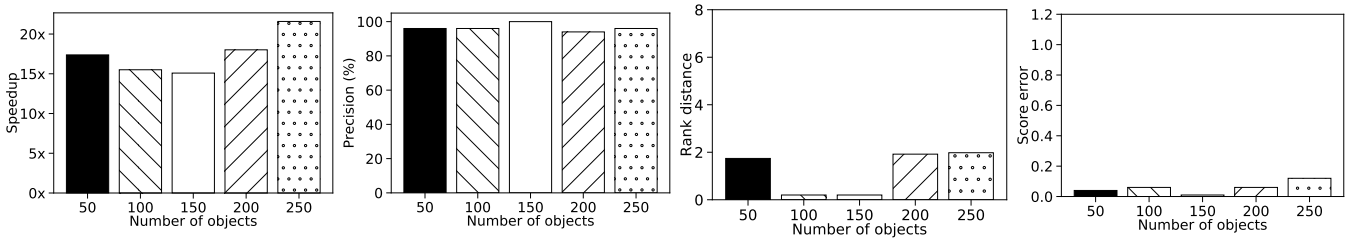
(a) Speedup (b) Precision (c) Rank distance (d) Score error

Figure 7: Complex scoring function



(a) Speedup (b) Precision (c) Rank distance (d) Score error

Figure 8: Varying the window size



(a) Speedup (b) Precision (c) Rank distance (d) Score error

Figure 9: Varying the number of objects in Visual Road

a specified accuracy target. BlazeIt [28] develops a SQL-like language for declarative video analytics. However, BlazeIt does not support Top-K query and its proposed language has no ranking semantic. BlazeIt uses CNN specialization and Approximate Query Processing [11] to support fast selection and aggregate queries. Focus [21] is a system that pre-builds indexes for objects in videos. During the ingestion time, a video is fed into a lightweight CNN to obtain the classification results and build an inverted-index between a class and the frames that contain objects of that class. At query-time, a query that looks for a particular object class is physically translated to an index-lookup operation instead. Focus also focuses on only selection queries.

"Probabilistic predicates" [34] can be viewed as a generalization of CNN specialization. A probabilistic predicate is a binary classifier that groups the input blobs into those that disagree and those that may agree with the query predicate. The former are discarded and the latter are passed to further processing. Our lightweight DCMDN can thus viewed as a probabilistic predicate where the frames are the input blobs. Nonetheless, it has no support of probabilistic predicates on Top-K query yet. TAHOMA [4] also works on selection, but its focus is on reducing the per-image inference cost (e.g., data loading and image transformation costs).

Recently, Panorama [57] discusses the support of unbounded vocabulary querying over videos. Specifically, all systems we mentioned above and Everest also inherit a limitation of the existing object detectors where the set of object classes is fixed. Panorama alleviates that by asking users to label an unknown object at run-time. Then, by designing a network that output embeddings, it can identify the other appearance of the same unknown object by using nearest neighbor search. The above method is compatible with Everest and we regard that as one of our future work. So far, video analytic systems focus on 2D videos. Yet, analytics on 3D videos should not be too far from the future [16].

Uncertain Top-K Processing The query processing part of Everest is related to traditional uncertain data cleaning [12, 35] and uncertain Top-K query processing. We have discussed the former and thus here we focus on the latter.

There are different notions of uncertain Top-K queries. The notion of *U-TopK* [54] returns a result set which has the highest probability of being Top-K under the possible world semantic. Everest adopts the same notion although U-TopK finds the most probable answer without cleaning. Therefore, it is possible that U-TopK returns an answer whose probability is 10^{-6}. By contrast, Everest guarantees the answer meets a probability threshold. *U-KRanks* [48] is another notion. In an *U-KRanks* result set, the i-th result in the result set is the most probable one to be ranked i-th over all the possible worlds. However, that does not guarantee that the result set as a whole is the most probable Top-K answer. *Probabilistic threshold Top-K* [22] is yet another notion. A Top-K result of such kind consists of all the tuples (can be less/more than K tuples) whose probability of being one of the Top-K tuples larger than a given threshold. The result set of this type also does not guarantee that the result set as a whole is the most probable Top-K answer. For example, it may return an empty set when no tuple satisfies the threshold requirement.

8. CONCLUSIONS

With a massive amount of video data available and generated incessantly, the discovery of interesting information from videos becomes an exciting area of data analytics. Although deep neural networks enable information extraction from videos with human-level accuracy, they are unable to process video data at scale unless efficient systems are built on top. State-of-the-art video analytics systems have not supported rich analytics like Top-K. In response, we build Everest, a fast and accurate Top-K video analytics system. To our knowledge, Everest is the first video analytics system that treats the uncertain output of deep networks as a first-class citizen and provides probabilistic guaranteed accuracy. Finding Top-K from the visual world is like finding a needle in a haystack. Everest contributes to the field the first prototype by judiciously putting together techniques from deep learning, computer vision, uncertain data cleaning, and Top-K query processing. Technically, Everest creates not only a new class of application for uncertain data analytics, but also the first *ground-truth-in-the-loop analytics* system in the uncertain database area. Currently, Everest is a standalone system that supports only ranking. Richer analytics can be enabled by integrating it with an expressive video query language like FrameQL [28]. Everest is an open source project and the source code is available at `https://github.com/everest-project/everest`.

9. REFERENCES

[1] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42, 2013.

[2] C. C. Aggarwal. Trio a system for data uncertainty and lineage. In *Managing and Mining Uncertain Data*, pages 1–35. Springer, 2009.

[3] C. C. Aggarwal and S. Y. Philip. A survey of uncertain data algorithms and applications. *IEEE Transactions on knowledge and data engineering*, 21(5):609–623, 2008.

[4] M. R. Anderson, M. Cafarella, T. F. Wenisch, and G. Ros. Predicate optimization for a visual analytics database. *SysML*, 2018.

[5] F. Arbabzadah, G. Montavon, K.-R. Müller, and W. Samek. Identifying individual facial expressions by deconstructing a neural network. In *German Conference on Pattern Recognition*, pages 344–354. Springer, 2016.

[6] N. Bruno, L. Gravano, and A. Marian. Evaluating top-k queries over web-accessible databases. In *Proceedings 18th International Conference on Data Engineering*, pages 369–380. IEEE, 2002.

[7] P. Burlina. Mrcnn: A stateful fast r-cnn. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3518–3523. IEEE, 2016.

[8] M. J. Carey and D. Kossmann. On saying enough already! in sql. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 219–230, 1997.

[9] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Region-based image querying. In *1997 Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 42–49. IEEE, 1997.

[10] C. Charles and V. Kerry. Store location in shopping centers: Theory and estimates. *Journal of Real Estate Research*, 27(3):237–266, 2005.

[11] S. Chaudhuri, B. Ding, and S. Kandula. Approximate query processing: No silver bullet. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 511–519, 2017.

[12] R. Cheng, E. Lo, X. S. Yang, M. Luk, X. Li, and X. Xie. Explore or exploit? effective strategies for disambiguating large databases. *PVLDB*, 3(1):815–825, 2010.

[13] N. Chopin. Fast simulation of truncated gaussian distributions. *Statistics and Computing*, 21(2):275–288, 2011.

[14] A. D'Isanto and K. L. Polsterer. Dcmdn: Deep convolutional mixture density network. *Astrophysics Source Code Library*, 2017.

[15] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.

[16] B. Haynes, A. Mazumdar, A. Alaghi, M. Balazinska, L. Ceze, and A. Cheung. Lightdb: A DBMS for virtual reality video. *PVLDB*, 11(10):1192–1205, 2018.

[17] B. Haynes, A. Mazumdar, M. Balazinska, L. Ceze, and A. Cheung. Visual road: A video data management benchmark. In P. A. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, and T. Kraska, editors, *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 972–987. ACM, 2019.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation OSDI 18)*, pages 269–286, 2018.

[22] M. Hua, J. Pei, W. Zhang, and X. Lin. Ranking queries on uncertain data: a probabilistic threshold approach. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 673–686, 2008.

[23] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose. Videoedge: Processing camera streams using hierarchical clusters. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 115–131. IEEE, 2018.

[24] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):1–58, 2008.

[25] M. T. Islam, M. A. Aowal, A. T. Minhaz, and K. Ashraf. Abnormality detection and localization in chest x-rays using deep convolutional neural networks. *arXiv preprint arXiv:1705.09850*, 2017.

[26] R. Jain and A. Hampapur. Metadata in video databases. *ACM Sigmod Record*, 23(4):27–33, 1994.

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

[28] D. Kang, P. Bailis, and M. Zaharia. Blazeit: optimizing declarative aggregation and limit queries for neural network-based video analytics. *Proceedings of the VLDB Endowment*, 13(4):533–546, 2019.

[29] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529*, 2017.

[30] B. A. Keating, M. Herrero, P. S. Carberry, J. Gardner, and M. B. Cole. Food wedges: framing the global food demand and supply challenge towards 2050. *Global Food Security*, 3(3-4):125–132, 2014.

[31] D. S. Kermany, M. Goldbaum, W. Cai, C. C.

Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[33] Y. Lu, A. Chowdhery, and S. Kandula. Optasia: A relational platform for efficient large-scale video analytics. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 57–70, 2016.

[34] Y. Lu, A. Chowdhery, S. Kandula, and S. Chaudhuri. Accelerating machine learning inference with probabilistic predicates. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1493–1508, 2018.

[35] L. Mo, R. Cheng, X. Li, D. W. Cheung, and X. S. Yang. Cleaning uncertain data for top-k queries. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 134–145. IEEE, 2013.

[36] S. P. Mohanty, D. P. Hughes, and M. Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[37] R. C. Moore. Possible-world semantics for autoepistemic logic. In *Proceedings of the Non-Monotonic Reasoning Workshop, Mohonk Mountain House, New Paltz, NY 12561, USA, October 17-19, 1984*, pages 344–354. American Association for Artificial Intelligence (AAAI), 1984.

[38] C. Norris, M. McCahill, and D. Wood. The growth of cctv: a global perspective on the international diffusion of video surveillance in publicly accessible space. *Surveillance & Society*, 2(2/3), 2004.

[39] C. Pakha, A. Chowdhery, and J. Jiang. Reinventing video streaming for distributed vision analytics. In *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*, 2018.

[40] A. Poms, W. Crichton, P. Hanrahan, and K. Fatahalian. Scanner: Efficient video analysis at scale. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[41] A. Raghu. Drones that do the work of 500 farmers are transforming palm oil. https://www.thestar.com.my/tech/tech-news/2019/11/20/drones-that-do-the-work-of-500-farmers-are-transforming-palm-oil, Nov 2019.

[42] C. Re, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 886–895. IEEE, 2007.

[43] C. Re, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 886–895. IEEE, 2007.

[44] J. Redmon. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013–2016.

[45] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[46] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg. Efficient processing of top-k spatial keyword queries. In D. Pfoser, Y. Tao, K. Mouratidis, M. A. Nascimento, M. F. Mokbel, S. Shekhar, and Y. Huang, editors, *Advances in Spatial and Temporal Databases - 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings*, volume 6849 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2011.

[47] H. Shen, S. Han, M. Philipose, and A. Krishnamurthy. Fast video classification via adaptive cascading of deep models. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2197–2205. IEEE Computer Society, 2017.

[48] M. A. Soliman, I. F. Ilyas, and K. C.-C. Chang. Top-k query processing in uncertain databases. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 896–905. IEEE, 2007.

[49] X. Sun, P. Wu, and S. C. Hoi. Face detection using deep learning: An improved faster rcnn approach. *Neurocomputing*, 299:42–50, 2018.

[50] D. Vasisht, Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. Sinha, A. Kapoor, M. Sudarshan, and S. Stratman. Farmbeats: An iot platform for data-driven agriculture. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 515–529, 2017.

[51] Y. Wang and M. Kosinski. Deep neural networks are more accurate than humans at detecting sexual orientation from facial images. *Journal of personality and social psychology*, 114(2):246, 2018.

[52] Z. E. Xu, M. J. Kusner, K. Q. Weinberger, M. Chen, and O. Chapelle. Classifier cascades and trees for minimizing feature evaluation cost. *J. Mach. Learn. Res.*, 15(1):2113–2144, 2014.

[53] Q. Ye and T.-K. Kim. Occlusion-aware hand pose estimation using hierarchical mixture density network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–817, 2018.

[54] K. Yi, F. Li, G. Kollios, and D. Srivastava. Efficient processing of top-k queries in uncertain databases with x-relations. *IEEE Trans. Knowl. Data Eng.*, 20(12):1669–1682, 2008.

[55] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 377–392, 2017.

[56] L. Zhang, R. Vaisenberg, S. Mehrotra, and D. V. Kalashnikov. Video entity resolution: Applying er techniques for smart video surveillance. In *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 26–31. IEEE, 2011.

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