Optimizing Video Selection LIMIT Queries With Commonsense Knowledge

Wenjia He[1], Ibrahim Sabek[2], Yuze Lou[1], Michael Cafarella[3]
[1]University of Michigan, Ann Arbor, [2]University of Southern California, [3]Massachusetts Institute of Technology
wenjiah@umich.edu,sabek@usc.edu,yuzelou@umich.edu,michjc@csail.mit.edu

ABSTRACT

Video is becoming a major part of contemporary data collection. It is increasingly important to process video selection queries — selecting videos that contain target objects. Advances in neural networks allow us to detect the objects in an image, and thereby offer query systems to examine the content of the video. Unfortunately, neural network-based approaches have long inference times. Processing this type of query through a standard scan would be time-consuming and would involve applying complex detectors to numerous irrelevant videos. It is tempting to try to improve query times by computing an index in advance. But unfortunately, many frames will never be beneficial for any query. Time spent processing them, whether at index time or at query time, is simply wasted computation.

We propose a novel index mechanism to optimize video selection queries with *commonsense knowledge*. Commonsense knowledge consists of fundamental information about the world, such as the fact that a tennis racket is a tool designed for hitting a tennis ball. To save computation, a lossy index can be intentionally created, but this may result in missed target objects and suboptimal query time performance. Our mechanism addresses this issue by constructing probabilistic models from commonsense knowledge to patch the lossy index and then prioritizing predicate-related videos at query time. This method can achieve significant performance improvements comparable to those of a full index while keeping the construction costs of a lossy index. We describe our prototype system, Paine, plus experiments on two video corpora. We show our best optimization method can process up to 97.79% fewer videos compared to baselines. Even the model constructed without any video content can yield a 75.39% improvement over baselines.

PVLDB Reference Format:
Wenjia He, Ibrahim Sabek, Yuze Lou, Michael Cafarella. Optimizing Video Selection LIMIT Queries With Commonsense Knowledge. PVLDB, 17(1): XXX-XXX, 2024.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://anonymous.4open.science/r/video-commonsense-optimization-E7AB/.

Proceedings of the VLDB Endowment, Vol. 17, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

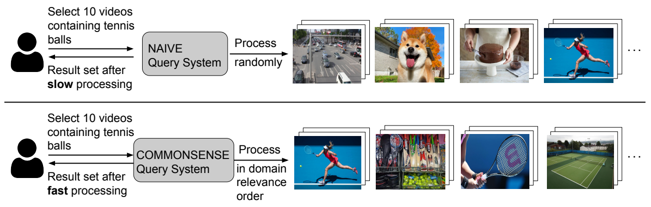

Select 10 videos containing tennis balls
Result set after slow processing
NAIVE Query System
Process randomly
Select 10 videos containing tennis balls
Result set after fast processing
COMMONSENSE Query System
Process in domain relevance order

Figure 1: A comparison between naïve query system and our query system based on commonsense knowledge

1 INTRODUCTION

With the increased availability and popularity of video databases [2, 31, 36], video selection queries have emerged as a growing area of research interest [3, 11, 26, 27, 39]. These queries are utilized for selecting desired videos that satisfy certain predicates, especially containing target objects. This kind of query can help video search in consumer-facing systems (e.g., social media platforms, albums in personal smartphones), in systems for filtering purposes (e.g., video censoring for privacy reasons), in the training set construction systems for machine learning pipelines (as with self-driving cars), etc. In principle, the object information in videos is unknown and needs to be extracted by applying an object detector.

A naïve method to process this type of query is to scan the video corpus. For each video, feed it to the frame-level object detector, which is typically a neural network model with a deep architecture, and add the video that satisfies the query's predicate to the result set. It is a common practice to include the LIMIT clause in video selection queries [26, 28] in the above applications due to the large size of existing video databases (e.g., over 500 hours of videos are uploaded to YouTube, an online media platform, every minute [40]). The LIMIT clause does not impose any ranking of results as long as records satisfy the query predicates. The above process is repeated until the result set meets the LIMIT size requirement. However, processing videos in random order leads to lots of wasted effort — the detection model would process a significant number of videos that do not satisfy the predicate. [R3.D2] As a consequence of the detector's long inference time [59] and long preprocessing time (e.g., decoding) [29], the query execution would be very slow.

Example 1. *Sansa is a tennis lover and wants to search for 10 videos with tennis balls from a video corpus comprising 100 various videos so as to study the trajectory of tennis serves. In this corpus, 20 videos can satisfy her requirement. She specifies the target object's name and the LIMIT number, yielding the following video selection query:*

```
SELECT * FROM videoCorpus
WHERE DetectedObject = ''Tennis Ball''
LIMIT 10
```

She utilizes a naïve query system that simply scans the corpus as shown in the upper part of Figure 1. In this way, only one out of five *videos on average will satisfy her query. Therefore, the query processor must run the detection model against roughly half of the corpus.*

System Goals — Optimizing selection queries using an index is common in traditional database management systems [10, 23]. One simple way to build an index for videos is to process every frame by the object detector and record all the detectable object information in each video. When a query arrives, the query processor can operate based on the index rather than invoking detectors so as to save query time. However, it is not practical to merely shift the burden of computational cost from query time to index time. In contrast to traditional index building based on available attribute values, this index for videos involves processing consecutive frames with an expensive detector. Because the vast majority of frames are not useful for queries, processing them is just a waste of valuable time and computation resources. Unlike previous video processing systems that rely on a traditional index, we intend to build a system that can achieve efficient query processing for object-based video retrieval as well as a low index cost.

Technical Challenge — Videos are processed by object detectors on a per-frame basis. To reduce the index cost, a straightforward option is to process fewer frames at index time. In this way, the index will contain objects that frequently appear in the video. However, objects that can only be detected in a small fraction of frames are likely to be missed, yielding a lossy index. Depending on the query, this may lead to worse query performance compared to a full index.

Solving the problem caused by the lossy index is a challenging task. Improving the quality of the index itself (i.e., making it more complete) while adhering to a limited index budget seems like an option, but current techniques are inadequate: the difference detector method [27] only works when the video is very static, and the specialized neural network model method [3, 26, 27] would reduce the results' accuracy and require running a bunch of binary classifiers for each potential queried object. Another option is to maintain a lossy index and then intelligently utilize it to quickly identify predicate-related videos at query time. Focus [20] clusters video frames based on approximate object information at ingest time and selects promising clusters at query time. However, the limited information obtained at index time remains insufficient. With the development of large language models (LLM, e.g., GPT-4 [48]), the task of selecting predicate-related videos from an incomplete index might appear to be simplified. However, the scalability and result quality of such models remain a limitation when applied to large video corpora. LLMs are superficially appealing and offer some advantages, but we found better results through other mechanisms based on commonsense knowledge.

Our Approach — [R3.D1] In this paper, we propose a novel data indexing mechanism: **at index-time, save resources by intentionally creating a lossy and sparse index; then at query-time, effectively "patch" the index by exploiting "commonsense" to estimate what information is missing.** As a result, an inexpensive index can be used to obtain query-time performance that is equivalent to using a much more expensive index.

What do we mean by commonsense knowledge? It refers to the basic understanding of the world that can be used to explain video content. Any human being watching a video can tell that objects in videos are not randomly arranged but are semantically correlated. For example, it is commonly known that people play tennis by hitting a tennis ball with a tennis racket. A human being who sees a tennis racket in a video frame can predict that this video is likely to contain a tennis ball sometime soon. Therefore, whether a selection predicate is satisfied by a video can often be inferred by observing just a few frames and applying commonsense knowledge. Previous works [27] have exploited the *frame-level* correlation to avoid processing frames that are almost identical visually, but they can only work in limited situations. Our approach leveraging *semantic-level* correlation in videos is more often applicable.

Based on this observation, we propose a method that, at index time, only a few frames of each video in the corpus are processed by object detection models to cheaply build a lossy index. At query time, our mechanism predicts the probability that a query object exists in each video from the lossy index and a commonsense knowledge-integrated probabilistic model and prioritizes videos with high probabilities, thereby avoiding unnecessary processing of irrelevant videos, much like traditional index methods.

[R3.O2] The core technical difficulty is how to **build a commonsense knowledge-integrated probabilistic model that is accurate enough to infer the missing objects and remains compact enough to ensure computational efficiency at query time.** Commonsense knowledge can be acquired through various sources — general commonsense from knowledge graphs and text, and queried video-specific commonsense from video content. Considering that videos which have the same object distribution as the queried videos may not always be accessible, we build two probabilistic models, one incorporating videos and the other without videos. When such videos are not available, we estimate the object existence probability through Bayes' theorem [25] and Fréchet inequalities [16] based on the object similarity in knowledge graphs. When such videos are available, we model it as a regression problem and adapt the BERT model [12] pre-trained on text to our scenario. We further fine-tune it with video-specific commonsense. These models can be constructed offline and do not rely on any query information from users.

Example 2. *Sansa employs our commonsense query system to decrease the runtime of the tennis ball selection query. At index time, one out of thirty video frames is processed to build the sparse index. At query time, videos that contain objects related to tennis balls in the index (e.g., rackets, players, courts, etc.) are processed first, as shown in the bottom half of Figure 1. Due to the fact that these videos are more likely to contain a tennis ball, only 15 videos are processed before selecting 10 desired videos, decreasing Sansa's wait time.*

Contributions — Our main contributions are as follows:

- We propose a novel index mechanism to optimize video selection queries based on commonsense knowledge. (Section 2)
- We design two commonsense knowledge probabilistic models, a conditional probability formula-based model without videos, and a neural network-based model that incorporates videos, to predict the probability of finding target objects in the unobserved video frames for different commonsense knowledge sources. (Section 3)
- We implement a prototype system, Paine, that embodies our algorithms, and evaluate it on two video datasets. Our

optimization method can save up to 97.79% query processing time compared to baselines. Even the commonsense model without any video content can yield up to a 75.39% improvement over baselines. (Section 4 and 5)

2 PROBLEM FORMULATION

In this section, we define the video selection query optimization problem in Section 2.1, introduce the query optimization strategy with commonsense knowledge in Section 2.2, discuss our design considerations in Section 2.3, and elaborate the domain assumptions in Section 2.4. All of the notations are listed in Table 1.

Parameter	Description	Example
$\mathcal{V} = \{V_i\}$	Video Corpus	YouTube videos
D	Object detector	YOLO9000
$\mathcal{O} = \{O_1, ..., O_r\}$	Target objects	{tennis ball}
k	LIMIT number	10 videos
n	# processed videos in query processing	20 videos processed by D at query time
$\mathcal{L}_i = [L_{i,1}, ..., L_{i,m_i}]$	m_i objects detected from sampled frames in V_i	[tennis racket, person]
$\mathcal{I} = \{\mathcal{L}_i \rightarrow V_i\}$	Index (observed object list → video)	{[tennis racket, person] → V_1}
$P(\mathcal{O}\|\mathcal{L}_i)$	Conditional probability that describes video contents	Probability that a tennis ball exists in a video containing [tennis racket, person]
M	Commonsense knowledge model that estimates $P(\mathcal{O}\|\mathcal{L}_i)$ to match true probabilities' rank ordering	An extension of a BERT model

Table 1: Frequently used notation

2.1 Optimization Problem Definition

A video selection query is characterized by a tuple of 4 parameters ($\mathcal{V}$, D, $\mathcal{O}$, k). The video corpus $\mathcal{V}$ usually contains a large number of videos. We focus on a general situation — the video corpus only includes pure videos without any textual information (e.g., titles, scripts, etc.). The object detector D, e.g., a neural network such as YOLO9000 [53], is applied to this corpus to identify videos of interest where the target objects $\mathcal{O}$ exist, e.g., searching for videos with a "tennis ball". The LIMIT clause with the number k restricts videos that should be returned to the user. [R1.W2] We assume users do not have error tolerance so all the returned videos should contain target objects $\mathcal{O}$. The result set can consist of any k satisfying videos instead of the top k videos that are most related to the target objects. Values of k can vary depending on the user. k might be small in consumer search applications. k can also be large when an engineer or a machine collects videos for a downstream training task.

In general, it takes a long time to process video selection queries through a simple scan; the number of processed videos n is proportional to the LIMIT number k, and the object detector D needs to be invoked for all the frames of the processed videos after decoding. Both model inference and video preprocessing are computationally expensive. If fewer videos are processed, the query processing time can be reduced. This leads to the following optimization problem:

Query Optimization Problem: [R1.W2] *Given a video selection query ($\mathcal{V}$, D, $\mathcal{O}$, k), minimize the number of videos n that will be processed while guaranteeing all the k videos in the result set satisfy the predicate for a 100% accuracy.*

2.2 Optimization Strategy

For query optimization, an index $\mathcal{I}$ consisting of the observed object list $\mathcal{L}_i$ (e.g., $\mathcal{L}_i$ = [tennis racket, person]) for each video V_i can help decrease n. Due to the limited index computation budget, only a fraction of frames can be indexed. It would make $\mathcal{L}_i$ an incomplete list, either due to rare objects or the detection model's inaccuracy. At query time, our mechanism applies a commonsense knowledge-integrated probabilistic model M to predict the conditional probability $P(\mathcal{O}|\mathcal{L}_i)$ from the imperfect index $\mathcal{I}$, indicating the probability of the event that videos containing $\mathcal{L}_i$ will also contain target objects $\mathcal{O}$. The predicted values are supposed to match the true ranking of these probabilities. Videos with higher values will be processed first in the hopes of quickly locating videos that can satisfy the predicates. Our algorithmic task is *how to design such a commonsense knowledge-integrated probabilistic model M to solve the video selection query optimization problem.*

2.3 Design Consideration

In our design, we consider choosing appropriate index content. We extract the incomplete object list $\mathcal{L}_i$ from a few frames of each video and store (object list $\mathcal{L}_i$, video V_i) pairs as the index. We do not include information about which videos are likely to contain target objects in the index. It is because target objects cannot be known in advance at index time and there are too many potential target object combinations (i.e., $2^{\text{\# of distinct objects}}$). This design also allows us to update the commonsense knowledge model M without changing the index. We defer the conditional probability prediction and video selection process to the query time.

Besides the incomplete object lists, other types of textual information from videos may also be utilized as the index to predict objects' existence, such as descriptions generated by video captioning models [67] or video topics identified through video classification [61]. These two options are at two extreme ends of the spectrum when it comes to information richness and computation cost — video topics can be obtained from small-scale classification models but only offer general domain information, while dense video captioning comprises rich information (e.g., actions) but obtaining it from image data requires a longer time. In contrast, object extraction is in between them and is more flexible due to the adjustable frame rate. [R2.W3, R2.D1] In addition, the extracted objects can be directly used for video selection queries that search for frequent objects at query time.

Visual features (e.g., embeddings from a visual model), which are widely used for content-based search, can also serve as the index [1, 30, 55]. Because these features are designed for reasonably accurate outcomes, they have larger dimensions compared to short object lists. When such features are used for video ranking, it would incur much extra overhead, especially for a large video corpus. Given these considerations, we choose object lists as the index.

2.4 Domain Assumption

Our strategy would be useful for a broad range of video selection queries with the following assumptions:

Video Type — Our techniques are designed for conventional videos where frames are semantically correlated. For some exotic video types, our methods can still "work" but we would not expect much speedup. These videos might include: (1) videos in which frames seem to be generated randomly and there is no normal logic among frames, for example, science fiction movie clips depicting an imaginary world like Avatar; (2) videos in which common objects do not exist, for example, typography videos with just text rather than visuals[1]; (3) videos in which frames do not change much across time, for example, surveillance video in an elevator at night. The difference detector method [27] is already designed to handle this static video type. Fortunately, the above video types constitute only a small proportion of existing videos.

Performance of the Object Detector — The results from the underlying detection model are viewed as the ground truth — we do not aim to improve them and we always use these results to evaluate our system's quality. Therefore, whether the detection models are accurate or relatively inaccurate does not actually impact our system. However, our system would not be helpful when a detection model is extremely inaccurate because there may not be a correlation between the detection results.

Target Object — We expect our techniques to accelerate most video selection queries except in two cases: (1) target objects that are semantically vague (e.g., "thing", "mechanism", etc.). These high-level objects are associated with a large number of other objects. It is not clear that these queries are very useful for users. (2) target objects that are hard to infer from single frames (e.g., "communicator", "leader", etc.), but are best inferred from visual clues that appear across sequential frames. In contrast, our current work focuses on frame-level detection. We might extend our approach to multi-frame sequences in future work.

3 ALGORITHMS

Now, we introduce our video selection query optimization algorithms. In Section 3.1, we introduce the overall procedure — our system's three-stage architecture. In Section 3.2, we focus on the model preparation stage and elaborate on the construction of commonsense knowledge models. In Section 3.3, we further improve the model based on the ground truth collected online.

3.1 Overall Procedure

We design a three-stage architecture in our system as shown in Figure 2. In the **indexing stage**, an index is built for the whole video corpus $\mathcal{V}$. For each video V_i, a few frames are processed by a detector that can detect various kinds of objects to produce an incomplete object list $\mathcal{L}_i$. These object lists mapped to videos as key-value pairs compose the index $\mathcal{I}$. [R3.D2] The indexing frame rate is adjustable according to varying index time budgets (e.g., sampling one out of thirty frames). We evenly distribute the sampled frames within each video so as to process a wider variety of frames, yielding more objects in the index. When a new video selection query arrives, the system enters the **query processing stage** — the query optimizer gives precedence to videos that are likely to contain target objects and avoids processing irrelevant videos. Algorithm 1 describes this procedure. In lines 1-7, videos in which target objects O are observed at index time are added to the result set directly. In lines 8-10, if there are already k videos in the result set, they are returned to the user. In line 11, the existence probability of target objects O conditioned on $\mathcal{I}$ is predicted by the prepared probabilistic model M. A high-quality model will assign high probabilities to predicate-related videos. In line 12, the video corpus is sorted in descending order of the probabilities in $\mathcal{P}$. After that, they are processed sequentially by the object detector D to determine whether they contain target objects. Desired videos are added to the result set until the set's size has reached the LIMIT number k or all the videos have been explored in lines 13-18.

Algorithm 1: Query processing

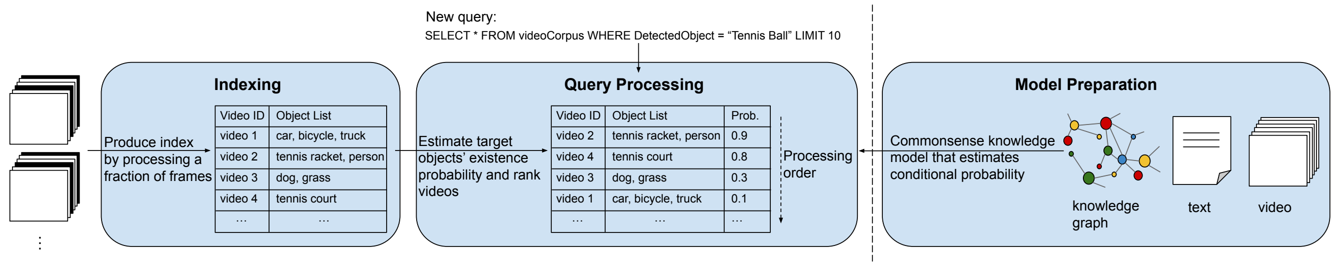
```
   Input: Video corpus $\mathcal{V}$, object detector $D$, target objects $O$,
          LIMIT number $k$, probabilistic model $M$, Index $\mathcal{I}$
1  for $V_i$ in $\mathcal{V}$ do
2      if $O \subseteq \mathcal{L}_i$ then
3          $resultSet$.append($V_i$);
4          $\mathcal{V}$.remove($V_i$);
5          $\mathcal{I}$.remove($\mathcal{L}_i \rightarrow V_i$);
6      end
7  end
8  if $|resultSet|$ >= $k$ then
9      return $resultSet$[:$k$]
10 end
11 $\mathcal{P} = M(\mathcal{I}, O)$;
12 $\mathcal{V} = \mathcal{V}[\mathcal{P}$.argsort()[::-1]];
13 repeat
14     $V_{select} = \mathcal{V}$.getNext();
15     if $O \subseteq D(V_{select})$ then
16         $resultSet$.append($V_{select}$);
17     end
18 until $|resultSet|$ == $k$ or $\mathcal{V}$.hasNext() == False;
   Output: $resultSet$
```

In the **model preparation stage**, as an offline step, we develop probabilistic models M integrating commonsense knowledge for the above procedure. This model is designed to predict the probability that any combination of objects exists in a video conditioned on the fact that another combination of objects is observed in this video. Model construction details will be introduced in Section 3.2.

3.2 Commonsense Knowledge Model

In this section, we describe the construction of the commonsense knowledge probabilistic model M in the model preparation stage. If the observed object list $\mathcal{L}_i$ contains target objects O, this video is handled in lines 1-7 in Algorithm 1 before applying the probabilistic model. In the subsequent parts of this section, we will only consider the index that does not contain target objects. In Section 3.2.1, we introduce a conditional probability formula-based model when there are only off-the-shelf knowledge graphs and text

[1]Video example: https://www.youtube.com/watch?v=qZEPs3vmYB4


New query:
SELECT * FROM videoCorpus WHERE DetectedObject = "Tennis Ball" LIMIT 10
Indexing
Produce index by processing a fraction of frames
Query Processing
Estimate target objects' existence probability and rank videos
Processing order
Model Preparation
Commonsense knowledge model that estimates conditional probability
knowledge graph
text
video

Video ID	Object List
video 1	car, bicycle, truck
video 2	tennis racket, person
video 3	dog, grass
video 4	tennis court
...	...

Video ID	Object List	Prob.
video 2	tennis racket, person	0.9
video 4	tennis court	0.8
video 3	dog, grass	0.3
video 1	car, bicycle, truck	0.1
...	...	...

Figure 2: The architecture of the overall procedure. In the model preparation stage, an offline stage, commonsense knowledge models that estimate objects' conditional existence probability are constructed from knowledge graphs, text or videos. The video corpus goes through the indexing stage; only a fraction of frames (black frames) are processed to create the incomplete object lists as the index. In the query processing stage, videos are ranked based on the index and the probabilistic model.

as commonsense knowledge sources. In Section 3.2.2, we propose a neural network-based model when videos are also available to provide commonsense knowledge.

3.2.1 *Model Built Without Videos.*

A wealth of commonsense knowledge is embodied in various existing sources, such as text corpora [56], knowledge graphs [44], etc, which are easy to obtain. In this section, we utilize these text-based sources to construct probabilistic models that incorporate basic and general commonsense knowledge.

Knowledge graphs (e.g., WordNet [44], ConceptNet [57], Wikidata [63], etc.) are networks of real-world entities, where each node represents an entity (e.g., object), and edges connecting these nodes represent the semantic relationships between them. For example, in ConceptNet [57], a node corresponding to "tennis racket" and another node corresponding to "ball" are connected by an edge labeled as "RelatedTo". The closeness between two nodes in a knowledge graph, reflecting the semantic closeness, can often indicate the likelihood of their coexistence in a video. For instance, based on the above knowledge graph triplet, we may deduce that "tennis racket" and "ball" probably exist in the same video. In recent years, there has been significant attention to knowledge graph embedding [64], which maps knowledge graph components into continuous vector spaces. The node embeddings, represented as numerical vectors, preserve the inherent structure of knowledge graphs — nodes that are closely connected in a knowledge graph tend to be mapped to proximal vectors in the embedding space. Based on this trend and the above finding, we estimate the conditional existence probability from knowledge graph embeddings.

First, we estimate the pairwise conditional probability $P(O|\mathcal{L}_i)$ in which both the target object list O and observed list $\mathcal{L}_i$ have only one object. The closeness between two node embeddings can be measured by cosine similarity [15], capturing objects' semantic similarity. On the other hand, the Jaccard coefficient for probability measures is a useful tool for gauging the similarity of two events. In our scenario, each event represents an object's existence in a video. Due to the connection between two objects' semantic similarity and their coexistence possibility, we take the cosine similarity between O and $\mathcal{L}_i$'s word embeddings to estimate the following ratio:

$$\frac{P(O \cap \mathcal{L}_i)}{P(O \cup \mathcal{L}_i)} := \max\left(\frac{embedding(O) \cdot embedding(\mathcal{L}_i)}{\|embedding(O)\| \cdot \|embedding(\mathcal{L}_i)\|}, \epsilon\right). \quad (1)$$

To avoid probabilities from quickly diminishing to zero, we set a small threshold ϵ. In this estimation, we took into account two factors: (1) the formulas on both sides are supposed to be symmetric with respect to O and $\mathcal{L}_i$; (2) the semantic similarity should not be used for estimating intersection probability directly, as this involves the influence of objects' own popularity.

The existence probability of a single object in a video can be influenced by its level of popularity, which can be reflected in Wikipedia pageview statistics [7]. We estimate the probability by the relative value of the average daily pageview:

$$P(O) := \sqrt{\frac{\text{Avg daily pageview of } O}{\max\{\text{Avg daily pageview of detectable objects}\}}}. \quad (2)$$

Probability $P(\mathcal{L}_i)$ can also be estimated in this way. According to the set's and probability's characteristics, conditional probability can be derived by plugging in the above estimations:

$$P(O|\mathcal{L}_i) = \frac{(P(\mathcal{L}_i) + P(O))\frac{P(O\cap\mathcal{L}_i)}{P(O\cup\mathcal{L}_i)}}{(1 + \frac{P(O\cap\mathcal{L}_i)}{P(O\cup\mathcal{L}_i)})P(\mathcal{L}_i)}. \quad (3)$$

In most cases, multiple distinct objects would be observed from videos at index time, and there may also be multiple target objects. We predict the conditional probability $P(O|\mathcal{L}_i)$ for this common situation based on the above pairwise probability estimation. Objects in the observed list $\mathcal{L}_i$ are denoted as $L_{i,1}, L_{i,2}, ..., L_{i,m_i}$, and target objects in O are denoted as $O_1, O_2, ..., O_r$. Similar to the "naïve" conditional independence assumptions in naïve Bayes classifiers [54], we adopt the following assumption: the existence of $L_{i,1}, L_{i,2}, ..., L_{i,m_i}$ is mutually independent, conditioned on the existence of O. According to Bayes' theorem [25],

$$P(O|\mathcal{L}_i) = \frac{P(O)P(\mathcal{L}_i|O)}{P(\mathcal{L}_i)} = \frac{P(O)\prod_{j=1}^{m_i} P(L_{i,j}|O)}{P(\mathcal{L}_i)}. \quad (4)$$

In this formula, $P(L_{i,j}|O)$ is calculated by Equation (3) if $r = 1$ or calculated by Equation (4) if $r > 1$.

Our next step involves estimating $P(\mathcal{L}_i)$ and $P(O)$ in Equation (4). According to Fréchet inequalities [16]:

$$\max(\sum_{j=1}^{m_i} P(L_{i,j}) - (m_i - 1), 0) \leq P(\mathcal{L}_i) \leq \min_j P(L_{i,j}). \quad (5)$$

Since we can compute the probability of two objects existing in the same video by plugging in Equation (1) and (2), i.e., $\forall j, k \in [1, m_i], j, k \in \mathbb{N}$,

$$P(L_{i,j} \cap L_{i,k}) = \frac{(P(L_{i,j}) + P(L_{i,k}))\frac{P(L_{i,j}\cap L_{i,k})}{P(L_{i,j}\cup L_{i,k})}}{1 + \frac{P(L_{i,j}\cap L_{i,k})}{P(L_{i,j}\cup L_{i,k})}}, \quad (6)$$

a tighter lower bound and upper bound of $P(\mathcal{L}_i)$ can be derived by pairing objects in $\mathcal{L}_i$:

$$P_{LB}(\mathcal{L}_i) = \max(\frac{1}{m_i - 1} \sum_{1 \le j < k \le m_i} P(L_{i,j} \cap L_{i,k}) - (\frac{m_i}{2} - 1), 0), \quad (7)$$

$$P_{UB}(\mathcal{L}_i) = \min_{1 \le j < k \le m_i} P(L_{i,j} \cap L_{i,k}). \quad (8)$$

Note that there are $m_i - 1$ unique combinations of non-repeating pairs, meaning that each pair occurs only once among the combinations (if m_i is not an even number, we add the universal set to the intersection $\mathcal{L}_i$). By summing the inequality (5) for all the combinations, we can obtain Equation (7). $P(\mathcal{L}_i)$ is estimated by the mean value of $P_{LB}(\mathcal{L}_i)$ and $P_{UB}(\mathcal{L}_i)$ in our model, and $P(O)$ will be derived similarly. As a result, $P(O|\mathcal{L}_i)$ in Equation (4) can be computed.

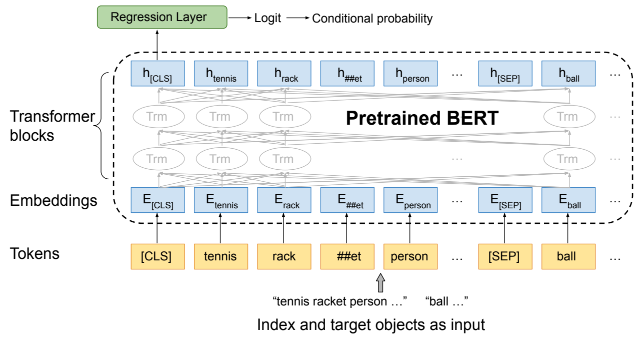
3.2.2 *Model Built With Videos.*

Although text-based commonsense knowledge sources are easy to acquire, they are constructed from crowd-sourced knowledge or online text collections, resulting in a generic knowledge base that may not accurately reflect the object associations depicted in queried videos. Previous studies show that commonsense knowledge can also be obtained from videos [60]. The distribution of objects portrayed in videos can provide video- and domain-specific knowledge, that is likely more valuable for our system. For example, "sun" and "tennis racket" are only loosely linked in conventional commonsense, but may be highly correlated in videos since most tennis matches occur outdoors on sunny days. It needs to be noted that the object distribution in the videos as the source of commonsense knowledge should be close to that in the queried video corpus. For instance, if this data is only gathered from traffic surveillance videos, it would not be very helpful for various videos on YouTube. Since these videos are not always accessible, the model built without videos in Section 3.2.1 is designated as the default model.

In this section, we aim to estimate the probability $P(O|\mathcal{L}_i)$ when videos are also available as the source of commonsense knowledge. To accomplish this, we collect a series of object lists, each containing objects observed in the same video. Potential ways to obtain these object statistics include accessing the ground-truth labels of an external similar video corpus or retrieving video selection query results from historical videos (a query's non-empty result indicates that the objects specified in the selection predicate indeed exist in the same video).

It is easy to think of estimating the conditional probability using the conditional relative frequency of objects in videos, i.e., $\frac{\text{\# of videos containing } O}{\text{\# of videos containing } O \text{ and } \mathcal{L}_i}$ for any possible O and $\mathcal{L}_i$, but this method has limitations. There could be several object combinations that either do not exist or occur infrequently. Such combinations can lead to inaccurate probability estimations. Moreover, there are too many object combinations, requiring a significant amount of storage space. Another way to estimate probability is by first estimating the pairwise coexistence probability and then applying the derived probability formula as in Section 3.2.1. However, this formula only integrates pairwise relationships and does not fully explore the potential of videos. In contrast, the algorithm below shows better performance.

Videos offer a unique opportunity to learn a commonsense knowledge model rather than explicitly derive theoretical formulas.



Regression Layer
Logit
Conditional probability
Transformer blocks
Pretrained BERT
Trm
Embeddings
Tokens
h[CLS] h tennis h rack h ##et h person ... h[SEP] h ball ...
E[CLS] E tennis E rack E ##et E person ... E[SEP] E ball ...
[CLS] tennis rack ##et person ... [SEP] ball ...
"tennis racket person ..." "ball ..."
Index and target objects as input

Figure 3: [R2.W2] Structure of the model built with videos. The input is a sequence of the indexed objects and target objects; the output is conditional probability prediction.

With the development of deep learning, neural network models have emerged as effective tools for learning extremely large and complex probability distributions [37]. In line with this trend, we develop a neural network model that takes the target objects and observed object lists in the index as inputs and is trained on object statistics to determine whether the target objects exist in the video.

Our neural network model is constructed based on the state-of-the-art language model BERT (Bidirectional Encoder Representations from Transformers) [12], specifically the uncased BERT base model. This model consists of 12 Transformer blocks and 110M parameters and has been pre-trained with two tasks (masked language modeling and next sentence prediction) on BookCorpus [71] and English Wikipedia. There are three notable benefits to adopting this model. First, the length of observed object lists in the index may vary, but this model is capable of accepting input sequences of different lengths. Second, BERT's pre-training on large text corpora means that it contains generic commonsense knowledge from text, which would be helpful for our task through transfer learning [62]. Third, as a Transformer-based model, BERT can process the entire sequence in parallel, making it faster than other models that process the sequence sequentially, such as LSTM [19].

[R2.W2] We start with the case when there is only one target object in a query's predicate. Figure 3 shows the structure of our neural network model. To construct the input, we organize the observed object list in the index and the target object as two sentences for each video. We concatenate the observed objects, $L_{i,1}, L_{i,2}, ..., L_{i,m_i}$ to form a sentence. For example, if the observed object list is [tennis racket, person], the sentence would be "tennis racket person". These observed objects are arranged in the order of their occurrence, which means $L_{i,u}$ occurs before $L_{i,v}$ or in the same frame as $L_{i,v}$ in video V_i if $u < v$. We believe this time sequence can provide useful semantic information. For instance, [cookie, flour, butter, chocolate] in sequence could indicate a video on how to make chocolate cookies, while [flour, chocolate, cookie, butter] in sequence could indicate a video about grocery store products. The observed object list is deduplicated, and we do not include the occurrence frequency of each object or additional separation tokens between every two objects in the sentence, as including these does not result in significant performance improvement. This reduces the sequence length and saves computational costs.

[R2.W2] These two sentences are transformed into tokens through the BERT tokenizer, and special tokens ([CLS] and [SEP]) are added to the beginning of them. For instance, two sentences "tennis racket person" and "ball" are tokenized as [[CLS], tennis, rack, ##et, person, [SEP], ball]. These tokens are then inputted into the pre-trained BERT model. Since our goal is to accurately rank videos by predicting a higher probability for the videos where the target object exists, we model it as a regression problem. The regression layer is connected to the pre-trained model, with the output vector representing [CLS] being fed into this layer. This layer comprises a dropout layer and a fully connected layer, which is fine-tuned for our task. We use MSE (mean squared error) as the loss function.

[R2.W2] During the model preparation stage, the target object is not pre-defined, and the goal of the commonsense knowledge model is to work for any possible target object. To achieve this, we construct the training data in the following manner: for each video object list in the collected object statistics, we select each item in this list as the target object O in turn and consider the remaining items as the observed objects, creating the training data with ground truth 1. Additionally, we randomly sample an object not included in the object list as the target object and use the entire object list as the observed objects, creating the training data with ground truth 0.

Besides random sampling, we also attempted to use knowledge graphs to identify related objects that do not exist in the videos and make them as the target objects to construct challenging examples for the model to learn. However, the model's performance suffered as a result. We speculate that this may be due to the fact that objects exist with a certain probability, and the way in which we constructed the data emphasized their non-existence, causing the model to learn in the opposite manner. More parameter settings and training details will be discussed in Section 5.1.

[R2.W2] During the query processing stage, when the model is applied to predict the conditional probability, we take into account the probability range by constraining the outputs to be between 0 and 1. If the output falls outside this range, we set the probability to either 0 or 1, depending on whether the output is less than 0 or greater than 1, respectively.

[R2.W2] Now we consider the case when there are multiple target objects in a query's predicate. Since the BERT model can accept sequences of varying lengths, it is straightforward to add multiple target objects to the end of the input. We can concatenate the target objects to construct the second sentence and then reuse the model that was trained on a single target object. Because the model was only trained on single target object cases, there are two other competitive options. Option 1 is to derive the conditional probability of multiple target objects from single target objects (e.g., $P(O_1, O_2|\mathcal{L}_i) = P(O_1|O_2, \mathcal{L}_i) \cdot P(O_2|\mathcal{L}_i)$). However, this option did not bring performance improvement in experiments and required longer inference time due to the computation of multiple probabilities (equal to the number of target objects). Option 2 is to train new models by generating training data with multiple target objects, but this approach is not scalable because the model needs to be trained for all potential numbers of target objects. Therefore, we choose the first algorithm rather than these two alternatives because it is the most practical and efficient solution.

[R1.W1, R1.D2] It needs to be noted that the input sentences that we construct are word concatenation but not real natural language. Even though BERT is pre-trained on natural language processing tasks, it cannot perform well on our task before being fine-tuned by videos, as demonstrated by experiments in Section 5.2.1. Therefore, we do not use pre-trained BERT for the case when videos are not available for providing commonsense knowledge in Section 3.2.1. It also needs to be noted that our model strategies are not exclusively developed for BERT. Therefore, even with the advent of more advanced language models in the future, our techniques remain suitable for integration into these newer models.

3.3 Online Learning

Based on the commonsense knowledge during the model preparation stage, we can generate primary models by applying the above algorithms. Subsequently, we can gather additional object statistics from videos during one or more rounds of query processing, which enables us to enhance the model built with videos. This information is especially beneficial because (1) compared with previously collected commonsense knowledge, the object distribution in these videos is more up-to-date and closely resembles that of videos specified by future queries; (2) since each frame of a video needs to be processed for identifying satisfying videos (line 12 in Algorithm 1), this object information may be more comprehensive and precise. To facilitate this approach, we have devised an online learning strategy whereby the BERT regression model is further updated by incorporating visual information extracted from the query processing stage.

4 SYSTEM PROTOTYPE

Our prototype system, Paine, implemented in Python, embodies our two optimization methods based on different commonsense knowledge sources for video selection queries. The system was deployed on Amazon EC2 g3.4xlarge instance with 16 vCPUs, 122GB memory, 1 NVIDIA Tesla M60 GPU, and 8GB GPU memory.

A major component of the system is the detection model. It is utilized to extract object information from videos for commonsense knowledge collection from videos, index construction, and predicate processing. We used the pre-trained YOLO9000 [53] model based on the neural network framework Darknet [52], implemented in C and CUDA. Just like the per-item processing in traditional DBMSs, this detection model was invoked for each frame, and batch processing was not enabled.

In the model preparation stage, we implemented the BERT regression model with Hugging Face Transformers [65] and PyTorch [49]. We loaded the pre-trained bert-base-uncased model from Hugging Face for sequence regression and fine-tuned it on the collected object statistics. After training, we stored the best checkpoint, which included the model architecture and weights, on disk along with knowledge graph embeddings for the model built without videos.

The whole video corpus for querying was processed by the detection component at a certain frame rate, and the detected objects were stored on disk as the index, mapping to each video. When a query arrived, the stored index and the commonsense knowledge (in the form of knowledge graph embeddings or the BERT regression model) were sent to the query optimizer, which would apply the model and prioritize videos with high predicted existence probabilities of the target object set.

5 EXPERIMENTS

We evaluated two core claims about Paine:

(1) The performance of Paine is better than state-of-the-art baselines — Paine can process fewer video clips during query processing time, yielding faster execution. Our model built with videos achieves the best performance. (Section 5.2).
(2) The performance of Paine varies with different experimental scenarios — different index time budgets, varying amounts of object statistics for model construction, the use of the online learning strategy, and different LIMIT values. Paine is effective in a wide range of scenarios (Section 5.3).

5.1 Experimental Setting

Here, we describe our workloads, baselines, and evaluation metric.

Workloads — We evaluated our Paine on multiple workloads. Each workload consists of the target object(s) in the predicate and a commonsense knowledge probabilistic model, plus a video corpus.

Video corpus — We tested two datasets that consist of diverse videos: YouTube-8M [2] and HowTo100M [43]. These YouTube-style videos comprise important workloads for applications introduced in Section 1. We cut videos into 60-second clips and selected clips that contain at least five distinct objects in 60 uniformly sampled frames. When using our system, this segmentation process applies universally to long videos. This step yields 19611 video clips from the YouTube-8M dataset and 31971 video clips from the HowTo100M dataset (note that original videos are not too long so that there would not be too many clips from the same video). In each dataset, 80% of video clips were used for the model preparation stage, and 20% of video clips were used in the index and query processing stages. Within the videos used for query processing, we used the ground truth of half of them for online learning and used the other half as the test data. When we split the dataset, we put clips from the same original video in the same category.

Object detection model and target object — YOLO9000 [53] was applied for building indexes at the rate of one frame per second. It was also used for obtaining the ground truth and collecting commonsense knowledge. In our experiments, we selected target objects that were in the training data and ConceptNet Numberbatch and existed in at least 10 queried videos. We also removed those with vague meanings, which are hyponyms of the synset "object.n.01" with path lengths of at most 3 from "object.n.01" in WordNet. Overall, there are 445 objects and 430 objects used as target objects in YouTube-8M and HowTo100M.

Commonsense knowledge probabilistic model — For the model built without videos, we used the 19.08 English-only version of ConceptNet Numberbatch [57], containing knowledge graph embeddings. We also used the daily pageview statistics of Wikipedia in 2022 to estimate individual probabilities. The small threshold ϵ in Section 3.2.1 is set to 0.01. [R2.W2, R3.D4] For the model built with videos, we constructed the training dataset with binary ground truth labels (1 for existence and 0 for non-existence) as explained in Section 3.2.2. To balance the distribution of positive and negative data, we randomly sampled the same number of objects that do not exist in the videos as the number of objects that exist in the videos for each video object list. This approach ensures that the quantity of negative video-target object pairs matches that of positive pairs. In the training process, the batch size is 128, the number of epochs is 4, the optimizer is Adam algorithm with weight decay [32], the initial learning rate is 2×10^{-5}, the learning rate warmup ratio is 0.1, and the model is evaluated every 500 steps. During the online learning, we split the ground truth as 4 : 1 for training and validation, and we set the batch size as 64. The epoch number is 2 on the YouTube-8M and 4 on the HowTo100M dataset.

Baselines — We evaluated Paine against best-known baselines:

Scan with lossy index — Scan the video corpus with the same inexpensive index that is built in our system. Videos with target objects in the index are processed first. The remaining videos are scanned in sequence which is adopted by most modern database management systems, such as SparkSQL [4].

Difference Detector — This method is from NoScope [27]. At index time, we processed frames with large mean squared errors to obtain the index. The query processing procedure is the same as Scan.

Adapted FOCUS — Because we focus on a substantial number of distinct objects and videos from diverse streams, we adapt Focus to our scenario while retaining its core idea of clustering. Videos are clustered based on the observed object lists at index time. At query time, the system prioritizes the clusters containing target objects.

When learning the commonsense knowledge from videos, traditional machine learning methods like association rule mining [34] and Bayesian network [58] can also be employed. However, they are less effective than our neural network model in our tests. [R2.W3, R2.D1] We also tried the specialized neural network method from BlazeIt [26] at index time in order to process more frames within the budget, but the shallow model would yield poor results.

Evaluation metric — With the same index time budget for all the compared methods, we computed the number of videos processed by the detection model divided by the optimal number as the evaluation metric. Here optimal means the result set's size — only the satisfying videos would be accessed by a perfect method. This metric is directly affected by different optimization methods and approximately reflects the slowdown ratio compared to the perfect situation. This holds true because the extra overhead of our optimization method is negligible, as explained in Section 5.2.

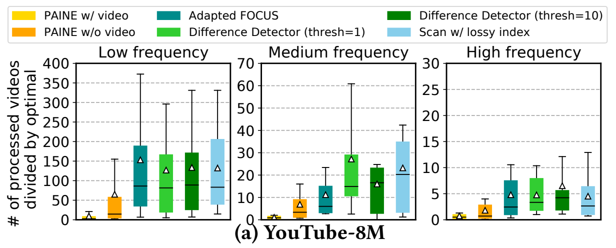
5.2 Overall Performance

This section shows the overall comparison results between Paine and the baselines on single-target-object and multiple-target-object workloads, as detailed in Section 5.2.1 and 5.2.2.

5.2.1 *Single Target Object.*

Summary — Paine can beat all the baselines when there is one single target object in each query. When comparing the model built with videos to baselines, we observed a significant improvement of up to 97.79% over Scan with lossy index while ensuring the same index cost (around 3% of the video collection). Even without videos, our model can still achieve up to a 75.39% improvement.

Overview — To test the performance on target objects of different frequency levels, we divided them into three groups: a low-frequency group when 10 - 50 queried videos contained the target object in the ground truth, a medium-frequency group when 50 - 100 queried videos contained it, and a high-frequency group when at least 100 queried videos contained it. We tested a moderate and



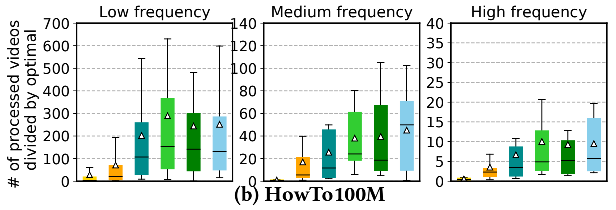


Figure 4: Comparison of our system, Paine, with baselines Adapted FOCUS, Difference Detector (the difference threshold is set to 1 and 10), and Scan with lossy index.

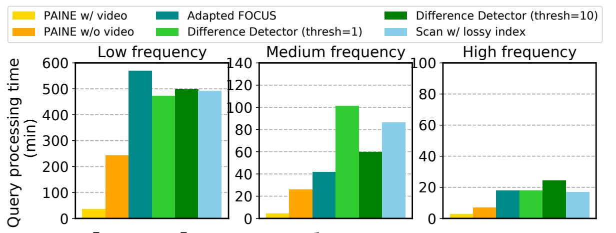


Figure 5: [R3.D11] Compare the average query processing time of Paine and baselines on YouTube-8M dataset.

representative LIMIT number, 20% of the total number of satisfying video clips for each query. [R3.D8] We removed target objects that have satisfying videos of the required size observed at index time, resulting in 125 target objects for YouTube-8M and 89 target objects for HowTo100M. For a fair comparison, all the baselines have the same index budget as ours. For the Difference Detector method, we set the difference threshold to 1 and 10.

To infer unseen objects from the lossy index, leveraging large language models seems like a promising approach. Because the quality of LLM depends on the prompt, if we just wrote the prompt ourselves, the results might primarily reflect the prompt rather than LLMs in general. [R3.D7] Therefore, we collected various prompts from 10 students (these prompts are shown with the source code on Github), spanning three solution types: direct object list ranking, video index number ranking, and video scoring. We evaluated GPT from OpenAI [6] with the highest maximum token capacity, gpt-3.5-turbo-16k, which allows up to 16,384 tokens. For the first two solution types, we truncated each object list in the index; for the third solution type, we partitioned the entire video corpus into multiple queries to ensure compatibility with the token limit.

Results — We show the performance of our system Paine with two models, Adapted Focus (baseline), Difference Detector with two difference thresholds (baseline), and Scan with lossy index (baseline) for target objects with different levels of frequency in Figure 4. [R3.D12] For each optimization method in the figure, the box spans from the first quartile to the third quartile values of the results, with a line marking the median and a triangle marking the mean value, and the whiskers extend from 10% to 90% of the data.

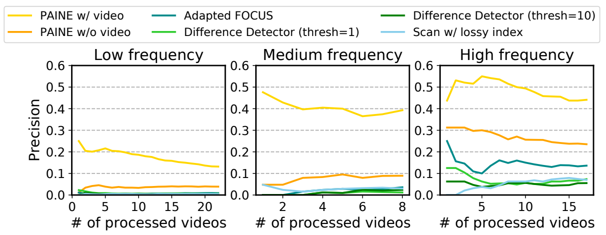


Figure 6: [R1.W2, R2.W3, R2.D1] The average precision of Paine and baselines with varying numbers of processed videos on YouTube-8M dataset.

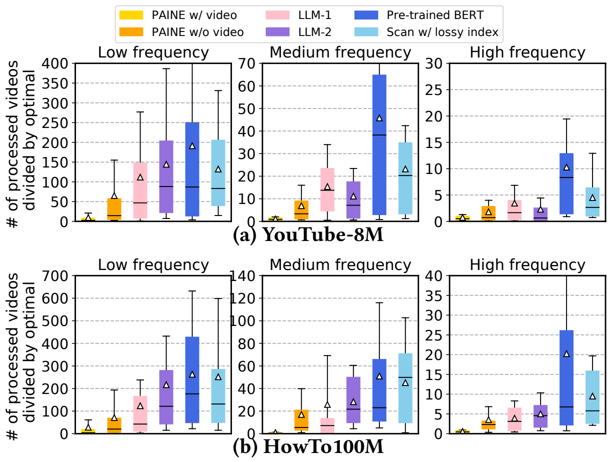


Figure 7: [R1.W1, R1.D2, R3.D5]Comparison of our commonsense knowledge models against the pre-trained BERT model and a large language model (GPT-3.5) using the optimal two prompts from a selection of ten.

From left to right, the evaluation results of baselines decrease, indicating easier queries with the increase of the target object's frequency. Same as the baselines, our system also delivers better performance when the frequency becomes higher. In each frequency group, Difference Detector methods (green boxes) show comparable performance to Scan with lossy index (blue box) because they do not bring many changes to index building — only around 0.7% of frames and 1.5% of frames can be skipped in our index when the difference threshold is 1 and 10 respectively. It indicates that there are always obvious motions in these video clips and the one-second interval is long enough to capture visual differences. Adapted Focus (dark cyan box) is better than other baselines as it can capture similarities between video clips. Our system (yellow and orange boxes), especially the model built with videos (yellow box), significantly outperforms all the baselines. This is particularly evident in the low- and medium-frequency groups, where Paine exhibits up to a 97.79% improvement and shows a lower variance. Drawing a comparison to Adapted Focus, it can be inferred that incorporating external commonsense knowledge does bring notable benefits.

When comparing our two models, as expected, the model built with videos achieves better results, yielding 83.48% - 97.79% improvement over baselines. Even though the other model only integrates general commonsense knowledge and does not include any video information, it still shows a strong performance — 33.82% - 75.39% improvement over baselines. In addition, when the frequency of target objects gets higher, the difference between these two models becomes less significant. This is reasonable because the relations between frequently-occurring objects and other objects are more well-known and easier to be captured by knowledge graphs or online text, making the video information less important.

The number of processed videos is a good proxy for query performance because the detection model's runtime dominates the total time. Our commonsense knowledge models bring extra overhead, but this overhead is minor. For example, it takes our BERT regression model around 5.3 seconds to predict probabilities for the YouTube-8M dataset. Considering the video clip length, frame rate, and the inference time of the detector, the total detection time significantly outweighs this extra overhead. Figure 5 shows the average query processing time of Paine against baselines on YouTube-8M dataset. The results are aligned with Figure 4a.

[R1.W2, R2.W3, R2.D1] Paine can process fewer videos because it can successfully assign high ranks to satisfying videos, that is, increasing the hit rate among the top videos. To validate this claim, we compared the average precision between Paine and baselines in three frequency groups on the YouTube-8M dataset in Figure 6. In each subfigure, we varied the number of processed videos from 1 to the number at which our model with videos can reach the LIMIT value. Our results show that the precision of Paine, especially the model with videos, is consistently higher than that of the baselines. As expected, precision tends to decrease as the number of processed videos increases and the frequency of the target objects becomes lower. This observation emphasizes the challenge of identifying satisfying videos in such cases.

[R3.D5] Within our query optimization framework, we also explored other options of the commonsense knowledge model design for the model preparation stage for an in-depth investigation of the component design. We compared our commonsense knowledge models with the LLM (GPT-3.5) and the pre-trained BERT model, as depicted in Figure 7. For the LLM results, we evaluated ten diverse prompts and showed the best two in this figure. [R3.D7] In the solution type of direct object list ranking, we find that it can return a few "obvious" answers, i.e., videos containing objects that are closely associated with target objects, but then stops generating object lists or repeats the previous ones or groups of object lists. In the solution type of video index number ranking, LLM will just output nonsensical number sequences eventually. These results may be attributed to two factors: firstly, the truncated index results in information loss; additionally, as the response tends to rely on nearby text, LLM outputs meaningless text at the end of the response. Furthermore, in the solution type of video scoring, although there is no information loss in the input and no concern about output repetition, LLM struggles to provide accurate scoring. Overall, none of these prompts can lead to satisfactory performance from LLM. [R1.W1, R1.D2] We also tried applying the pre-trained BERT model directly to compute the existence probability and sort videos. We can see that its performance is even worse than the baseline scan with lossy index. It demonstrates that when building a model without videos, our method is more effective than pre-trained BERT, and when building a model with videos, the fine-tuning process is beneficial.

5.2.2 *Multiple Target Objects.*

Summary — Paine works well for video selection workloads with multiple target objects. In two-target-object scenarios, Paine can achieve a 92.23% improvement compared with baselines; in three-target-object scenarios, it can yield a 91.43% improvement.

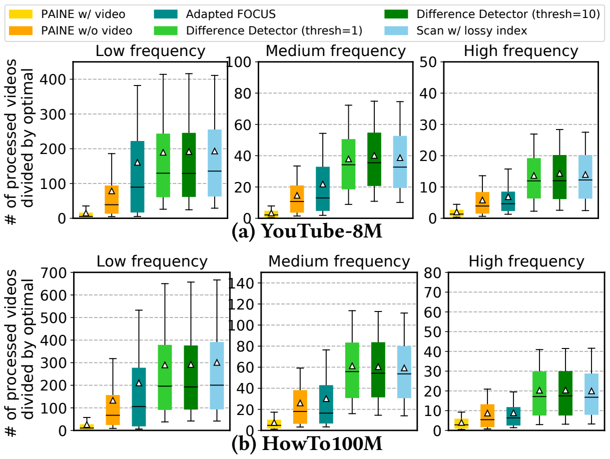


Figure 8: Compare Paine with baselines when there are two target objects in each query.

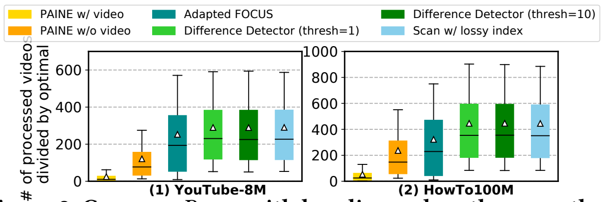


Figure 9: Compare Paine with baselines when there are three target objects in each query.

Overview — We generated new queries that contain two or three distinct target objects. The object combination in two-target-object queries should satisfy the following requirements: (1) at least 10 arriving video clips contained it; (2) both objects in the combination existed in the ConceptNet NumberBatch; (3) both objects were not within a distance of length 3 with the entity 'object.n.01' in the WordNet in order to remove vague words. When creating three-target-object queries, due to the substantial amount of potential combinations, we only tested those combinations that exist in 10-15 arriving videos. [R3.D10] Overall, we constructed 8051 and 10004 queries containing two target objects (e.g., "sofa" and "gas oven") and 13298 and 17541 queries containing three target objects (e.g., "bottle", "bowl", and "pizza") for each dataset respectively.

Results — Figure 8 and Figure 9 show the comparison results for two-target-object and three-target-object workloads. In Figure 8, our system, especially Paine with videos, achieves the best result. Even though our BERT regression model was only trained with single-target-object examples, its advantage over baselines is still significant in the multiple-target-object workload, yielding up to a 92.23% improvement on YouTube-8M data and a 91.37% improvement on HowTo100M data. The model built without videos exhibits great performance in the low-frequency group. However, in the medium- and high-frequency groups, this model falls short of exceeding one of the baselines, Adapted Focus. Selecting videos containing frequent objects is a relatively easy task for baselines. In order to beat baselines, the object relations need to be well modeled, but general commonsense knowledge from knowledge graphs

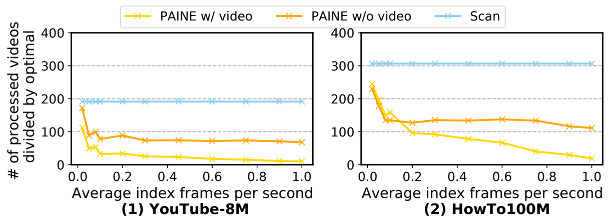


Figure 10: The performance of optimization methods when varying the index size.

and text may not be enough to achieve this goal. Our model built without videos does not aim to be helpful in this case.

Even though processing the three-target-object workload is a harder task, Paine outperforms baselines for infrequent target object combinations as shown in Figure 9. Due to the space limit, we do not show more results of queries with more than three target objects. However, according to the trend, we expect Paine can work well for multiple target objects.

5.3 Various Settings

In this section, we vary the experimental scenarios to test how our algorithms perform in a wide range of settings, including different index time budgets in Section 5.3.1, different amounts of object statistics for model training in Section 5.3.2, the effect of the online learning strategy in Section 5.3.3, and different LIMIT numbers in Section 5.3.4. We tested single-target-object queries in the low-frequency group set the LIMIT fraction to 20%.

[R3.D1] In a hard scenario, the target objects are quite rare and cannot be captured by the index. To test how our methods perform in this scenario, we ran experiments in a hard mode in this section, that is to say, for each query, the target object was deleted from the index. We did not test this mode in the overall comparison in Section 5.2 because all three baselines rely directly on the index and they would reduce to a simple scan method in the hard mode.

5.3.1 *Index Budget.*

Summary — In general, our optimization algorithms become more useful with the increase of the index size. They have a significant advantage over Scan even with a tiny index size.

Overview — In our default setting, frames are processed to build the index at the rate of one frame per second. As the index time budgets vary, it may be necessary to reduce the rate accordingly. We experimented with different average frame rates for the index, ranging from 0.02 to 1 frame per second.

Results — Figure 10 shows the performance of Paine and Scan under different index settings. We find that when the average index frame rate increases, our system's performance becomes better. It is reasonable because object information from more frames usually can promote a better understanding of the video content. We can also see that our system can achieve substantial improvement compared with Scan even with an extreme index budget limit: up to a 74.52% improvement on YouTube-8M and up to a 36.41% improvement on HowTo100M when only 0.05 frames are indexed per second on average. We notice that the model built with videos improves at a faster rate as the index size increases compared to the model built without videos, indicating that the commonsense knowledge from videos equips the former model with better capabilities to leverage complex relationships among multiple objects.

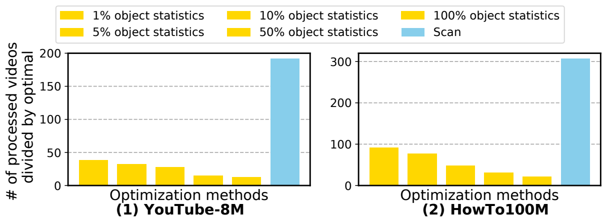


Figure 11: The performance of optimization methods trained with different amounts of videos

5.3.2 *Size of Videos for Commonsense Knowledge Collection.*

Summary — The performance of the model built with videos becomes better when the size of videos used for commonsense knowledge collection increases for model training.

Overview — In practice, in the model preparation stage, it might not be feasible to collect a large amount of videos that have the same object distributions as new-coming videos. To test this situation, we varied the size: using 1%, 5%, 10%, 50%, and 100% of the assigned videos to train the model. For a fair comparison, we adjusted the training epochs for each model to ensure the same amount of training steps. The online learning strategy was not applied in this experiment.

Results — Figure 11 shows the comparison of different probabilistic models trained with varying sizes of videos that are used for commonsense collection. When the size is larger, it can yield more accurate and comprehensive commonsense knowledge so that our optimization method can process fewer video clips. When the size is very small (e.g., 1%), the BERT regression model is not fully trained but it still substantially outperforms the scan method.

5.3.3 *Online Learning.*

Summary — Our model built with videos can be further improved by the online learning strategy. In general, the performance gets better with the increase in video size during online learning.

Overview — In order to test whether the online learning strategy is effective, we varied the fraction of data used for online learning from 0% to 100%. To make the comparison fair, we adjusted training epochs for the experiments with different data fractions to achieve the same training steps.

Results — Compared with no online learning, the model after full online learning can process up to 19.98% fewer videos on YouTube-8M and up to 7.81% fewer videos on HowTo100M. When the fraction increases from 10% to 100%, the overall performance is gradually enhanced. However, when the online learning fraction is too small, the performance could even become worse. A possible reason could be that the model has overfitted to the limited examples, resulting in unsatisfactory results on new data. It would be better to use this strategy after enough video information is gathered.

5.3.4 *LIMIT Values.*

Summary — Our optimization algorithms are effective across a wide range of LIMIT values, even when faced with the demanding task of retrieving all satisfying videos.

Overview — In the conducted experiments, we selected a moderate and representative LIMIT value, 20% of the videos that can satisfy the query's predicate. To assess the robustness of our models across varying LIMIT settings, we systematically tested the LIMIT fractions ranging from 10% to 100%, with intervals of 10%.

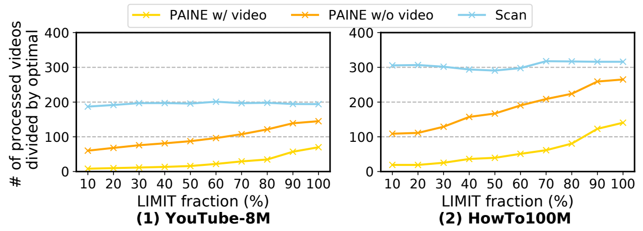


Figure 12: [R1.W2] The performance of optimization methods when varying the LIMIT value.

Results — [R1.W2] Figure 12 shows the comparative performance of different optimization methods on two datasets across a range of LIMIT fractions. The x-axis represents the LIMIT fraction, which is the fraction of LIMIT value relative to the total count of videos that can satisfy the predicate. Both of our models (yellow and orange lines) show significant performance across a wide range of LIMIT settings, outperforming the scan method. As the LIMIT fraction approaches 100%, the task becomes more challenging, but our models continue to be substantially faster than scan.

5.4 Discussion

In this work, we focus on video selection queries with a built-in object detector. Our optimization methods are orthogonal with the detection model, so as an extension, users can also provide their own detection model in the predicate. If the user-defined detector can detect new object categories, useful object statistics would not be collected in the model preparation stage. Fortunately, our general-purpose model built without videos may still perform well.

[R3.D9] Besides the benefit of accelerating query processing, our optimization method can also return more relevant videos to users. For example, when videos are processed by a simple scan, a video containing only a single frame of the target object might be returned, but it is unlikely to happen in our method. If relevance ranking is considered, we believe our method can still beat baselines. In addition, it is unlikely for Paine to return rare videos, for example, returning videos containing a tennis racket in a department store rather than on a tennis court when the target object is "tennis racket". However, users can specify the atypical target object combination (e.g., "tennis racket" and "department store") in their query if they are interested in rare situations. Even though target objects are not closely related in such cases, results from the low-frequency group in Section 5.2.2 demonstrate that Paine can effectively retrieve rare videos.

It is worth noting that the query processing procedure can be further improved during the invocation of the object detector, for example, batch processing in voodoo indexing [18], specialized NNs, the difference detector, etc. These techniques can be combined with our system or baselines. To ensure a fair comparison and focus on the video ranking mechanism before the detection model invocation, we did not include these techniques in our experiments.

6 RELATED WORK

Some previous related literature is outlined below:

Video Analytics — Due to the complexity of video analysis models and the growing size of video data, optimizing video queries to improve efficiency is an important research direction. There are two optimization categories. One category optimizes queries with error tolerance, i.e., some returned videos may actually not satisfy the predicates. These works apply binary classification models that are shallower than the object detection models in predicates [3, 26, 27], select appropriate input video settings and model settings (e.g., the frame resolution, the frame rate, and the model size) [3, 5, 24, 33, 46, 70], propose probabilistic predicates to prefilter items [39], or construct approximate indexes of possible object classes [20]. The other category accelerates query processing while not reducing the accuracy of query results. These works enable parallel processing for large-scale video analysis [38, 50], build query-independent index groups to select candidates for model processing [18], or materialize and reuse user-defined functions' results for exploratory video analytics [66]. However, none of the previous literature has exploited commonsense knowledge for video query optimization.

Commonsense Knowledge Applications on Visual Data — Over the past decades, commonsense knowledge has been collected and modeled by knowledge graphs (e.g., WordNet [44], ConceptNet [57], Wikidata [63], and CSKG [22]), natural language corpus (e.g., NELL [8] and OMCS [56]), and multiple choice QA problems (e.g., VCR [68] and SWAG [69]) through crowdsourcing, mining and other advanced methods. Because commonsense knowledge consists of facts that are expected to be known by all humans, it has been applied to many computer vision tasks in order to achieve human-level performance. It can be utilized to improve the accuracy of object detection [14, 51], action recognition [17], and image classification [41], and enable visual commonsense reasoning [68].

Query Optimization Based on Indexes — Traditional index structures [9, 42] have been widely applied in modern database management systems [13, 45, 47] for query optimization. Some works have designed novel index techniques. Database cracking [21] gradually cracks databases for building indexes according to users' queries. Learned index structures [35] adopt machine learning models for index construction in order to reduce time. However, these indexes are not applicable to video and other unstructured data.

7 CONCLUSION AND FUTURE WORK

It is common for video selection queries to include deep learning models for object detection in the predicates. Due to the long inference time of these models, it is crucial to develop optimization techniques. In summary, we propose a novel index mechanism that utilizes an inexpensive index and commonsense knowledge to prioritize videos that are likely to contain target objects. We have implemented a prototype system, Paine, and tested it on real-world video corpora with a wide range of settings.

Our work represents an important first step towards video query optimization based on commonsense knowledge. For future work, we could consider calibrating the predicted probabilities to enable skipping the processing of high-probability videos, if users can tolerate a small error. [R1.D1] Moreover, it would be exciting to expand our approach to tackle more general cases, such as video selection queries that filter objects with specific position constraints or filter temporal features (e.g., activities from multi-frame sequences). It is promising to reuse our framework — building corresponding indexes and commonsense knowledge models for these features, and prioritize videos that are likely to satisfy the predicate.

REFERENCES

[1] Sadiq H Abdulhussain, Basheera M Mahmmod, M Iqbal Saripan, SAR Al-Haddad, Thar Baker, Wameedh N Flayyih, Wissam A Jassim, et al. 2019. A fast feature extraction algorithm for image and video processing. In *2019 international joint conference on neural networks (IJCNN)*. IEEE, 1–8.

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. 2016. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675* (2016).

[3] Michael R Anderson, Michael Cafarella, German Ros, and Thomas F Wenisch. 2019. Physical representation-based predicate optimization for a visual analytics database. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1466–1477.

[4] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. 2015. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1383–1394.

[5] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Maddenee. 2020. MIRIS: fast object track queries in video. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1907–1921.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[7] Yuru Cao, Hely Mehta, Ann E Norcross, Masahiko Taniguchi, and Jonathan S Lindsey. 2020. Analysis of Wikipedia pageviews to identify popular chemicals. In *Reporters, Markers, Dyes, Nanoparticles, and Molecular Probes for Biomedical Applications XII*, Vol. 11256. SPIE, 24–41.

[8] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.

[9] Douglas Comer. 1979. Ubiquitous B-tree. *ACM Computing Surveys (CSUR)* 11, 2 (1979), 121–137.

[10] Brian F Cooper, Neal Sample, Michael J Franklin, Gisli R Hjaltason, and Moshe Shadmon. 2001. A fast index for semistructured data. In *VLDB*, Vol. 1. 341–350.

[11] Maureen Daum, Brandon Haynes, Dong He, Amrita Mazumdar, and Magdalena Balazinska. 2021. TASM: A tile-based storage manager for video analytics. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1775–1786.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Korry Douglas and Susan Douglas. 2003. *PostgreSQL: a comprehensive guide to building, programming, and administering PostgresSQL databases*. SAMS publishing.

[14] Yuan Fang, Kingsley Kuan, Jie Lin, Cheston Tan, and Vijay Chandrasekhar. 2017. Object detection meets knowledge graphs. International Joint Conferences on Artificial Intelligence.

[15] Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276* (2016).

[16] Maurice Fréchet. 1951. Sur les tableaux de corrélation dont les marges sont données. *Ann. Univ. Lyon, 3ˆ e serie, Sciences, Sect. A* 14 (1951), 53–77.

[17] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. 2019. I know the relationships: Zero-shot action recognition via two-stream graph convolutional networks and knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 8303–8311.

[18] 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.

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

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

[21] Stratos Idreos, Martin L Kersten, Stefan Manegold, et al. 2007. Database Cracking.. In *CIDR*, Vol. 7. 68–78.

[22] Filip Ilievski, Pedro Szekely, and Bin Zhang. 2021. Cskg: The commonsense knowledge graph. In *European Semantic Web Conference*. Springer, 680–696.

[23] Matthias Jarke and Jurgen Koch. 1984. Query optimization in database systems. *ACM Computing surveys (CsUR)* 16, 2 (1984), 111–152.

[24] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 253–266.

[25] James Joyce. 2003. Bayes' theorem. (2003).

[26] 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).

[27] 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).

[28] 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).

[29] Daniel Kang, Ankit Mathur, Teja Veeramacheneni, Peter Bailis, and Matei Zaharia. 2020. Jointly optimizing preprocessing and inference for DNN-based visual analytics. *arXiv preprint arXiv:2007.13005* (2020).

[30] Vadim Kantorov and Ivan Laptev. 2014. Efficient feature extraction, encoding and classification for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2593–2600.

[31] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 1725–1732.

[32] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

[34] Sotiris Kotsiantis and Dimitris Kanellopoulos. 2006. Association rules mining: A recent overview. *GESTS International Transactions on Computer Science and Engineering* 32, 1 (2006), 71–82.

[35] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*. 489–504.

[36] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. 2011. HMDB: a large video database for human motion recognition. In *2011 International conference on computer vision*. IEEE, 2556–2563.

[37] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

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

[39] 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.

[40] Thomas Marcoux, Nitin Agarwal, Recep Erol, Adewale Obadimu, and Muhammad Nihal Hussain. 2021. Analyzing cyber influence campaigns on YouTube using YouTubeTracker. In *Big Data and Social Media Analytics*. Springer, 101–111.

[41] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. 2016. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844* (2016).

[42] Ward Douglas Maurer and Ted G Lewis. 1975. Hash table methods. *ACM Computing Surveys (CSUR)* 7, 1 (1975), 5–19.

[43] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2630–2640.

[44] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[45] Ross Mistry and Stacia Misner. 2014. *Introducing Microsoft SQL Server 2014*. Microsoft Press.

[46] Oscar Moll, Favyen Bastani, Sam Madden, Mike Stonebraker, Vijay Gadepally, and Tim Kraska. 2022. Exsample: Efficient searches on video repositories through adaptive sampling. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2956–2968.

[47] AB MySQL. 2001. MySQL.

[48] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[49] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

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

[51] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. 2007. Objects in context. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 1–8.

[52] Joseph Redmon. 2013. Darknet: Open source neural networks in c.

[53] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7263–7271.

[54] Irina Rish et al. 2001. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, Vol. 3. 41–46.

[55] Jie Shao, Heng Tao Shen, and Xiaofang Zhou. 2008. Challenges and techniques for effective and efficient similarity search in large video databases. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1598–1603.

[56] Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 1223–1237.

[57] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

[58] Todd Andrew Stephenson. 2000. *An introduction to Bayesian network theory and usage*. Technical Report. Idiap.

[59] 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.

[60] Niket Tandon, Aparna S Varde, and Gerard de Melo. 2018. Commonsense knowledge in machine intelligence. *ACM SIGMOD Record* 46, 4 (2018), 49–52.

[61] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. 2019. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1207–1216.

[62] Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 242–264.

[63] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[64] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[65] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

[66] Zhuangdi Xu, Gaurav Tarlok Kakkar, Joy Arulraj, and Umakishore Ramachandran. 2022. EVA: A Symbolic Approach to Accelerating Exploratory Video Analytics with Materialized Views. In *Proceedings of the 2022 International Conference on Management of Data*. 602–616.

[67] Antoine Yang, Arsha Nagrani, Paul Hongsuck Seo, Antoine Miech, Jordi Pont-Tuset, Ivan Laptev, Josef Sivic, and Cordelia Schmid. 2023. Vid2Seq: Large-Scale Pretraining of a Visual Language Model for Dense Video Captioning. *arXiv preprint arXiv:2302.14115* (2023).

[68] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6720–6731.

[69] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326* (2018).

[70] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. 2017. Live video analytics at scale with approximation and delay-tolerance. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 377–392.

[71] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*. 19–27.