Towards Query Processing in Video Database Management Systems

by

Wenjia He

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2024

Doctoral Committee:

Principal Research Scientist Michael Cafarella, MIT, Co-Chair
Professor H. V. Jagadish, Co-Chair
Assistant Professor David Fouhey, New York University
Professor Lu Wang

Wenjia He

wenjiah@umich.edu

ORCID iD: 0000-0002-4661-9314

To my parents, my love, and my friends.

ACKNOWLEDGEMENTS

Five and a half years ago, driven by my strong passion for research, I embarked on a journey to pursue a Ph.D. destined to be challenging. Along this path of doctoral studies, self-doubt, self-deprecation, and anxiety easily became unwelcome companions. Two years of the pandemic have made everything even more difficult. Thankfully, I was fortunate to be surrounded by a wonderful group of people, supporting me in successfully navigating and completing this impressive journey.

Foremost, I would like to express my gratitude to my advisor, Principal Research Scientist Michael Cafarella at MIT CSAIL. Mike is a smart and passionate scientist, continually exploring the most interesting and cutting-edge research directions. He serves as the mentor in my Ph.D. journey, nurturing my growth and independence by guiding me to learn every facet of research, from generating innovative ideas to writing papers. Mike is very responsible for students, including but not limited to keeping track of the progress of our projects, providing financial support for participation in conferences, and so on. In addition, Mike cares about our mental well-being, encouraging us to maintain a healthy work-life balance. From my perspective, Mike is unquestionably the best advisor.

I am very grateful to Professor H. V. Jagadish, Professor David Fouhey, and Professor Lu Wang for serving on my dissertation committee. They provided valuable and insightful

comments and suggestions for improving the quality of my dissertation, especially the fourth work.

I am very thankful to my undergraduate research advisor, Principal Researcher Mike Chieh-Jan Liang at Microsoft Research Asia. His guidance in my undergraduate thesis was enlightening, leading me to understand and appreciate research for the first time. This great experience under his mentorship ultimately influenced my decision to pursue a Ph.D. degree.

Next, I would like to express my appreciation to professors and labmates from the Data Systems Group at MIT CSAIL: Professor Samuel Madden, Professor Michael Stonebraker, Professor Tim Kraska, Professor Elkindi Rezig, Professor Ibrahim Sabek, Professor Brit Youngmann, Dr. Chunwei Liu, Dr. Oscar Moll, Dr. Jialin Ding, Dr. Kapil Vaidya, Peter Baile Chen, Darryl Ho, Ferdinand Kossmann, Eugenie Lai, Tianyu Li, Markos Markakis, Amadou Latyr Ngom, Matt Perron, Matthew Russo, Sivaprasad Sudhir, Joana M.F. da Trindade, Ziniu Wu, Geoffrey Yu, Anna Zeng, Xinjing Zhou, and Sylvia Zhang, my labmates from the Database Research Group at the University of Michigan: Dr. Mike Anderson, Professor Abolfazl Asudeh, Professor Yongjoo Park, Dr. Jiamin Huang, Dr. Nikita Bhutani, Dr. Dong Young Yoon, Dr. Chris Baik, Dr. Zhongjun Jin, Dr. Jie Song, Dr. Rui Liu, Professor Yuval Moskovitch, Boyu Tian, Junghwan Kim, Yin Lin, Yuze Lou, Junjie Xing, Jinyang Li, Jie Liu, Farima Fatahi Bayat, and Tianji Cong, and other collaborators: Maxwell Strome from University of Michigan, Eileen Chau from MIT, and Dr. Abdul Wasay from Intel Labs. These talented people are truly amazing. I feel honored to work with them, engage in meaningful conversations, and share enjoyable moments in off-site events. My lab life has been an unforgettable, cherished, and wonderful experience

that will forever hold a special place in my heart.

I am also deeply thankful to my roommate, Dr. Yibo Wang, my friends since the middle school, Meng Yuan and Fanghan Lei, and my undergraduate friends, Aowen Li, Yumeng Liu, Qi Wang, Chen Chen, and Yang Wang. We frequently share our lives and stories with each other, even when separated by distance, which is a tremendous source of support particularly during the pandemic.

I would like to give my special thanks to my partner, Xumiao Zhang. I feel lucky to meet with Xumiao in 2018 and we have been together since the beginning of our Ph.D. journey. Coming from the same undergraduate university and entering the CSE Ph.D. program in the same year, our shared experiences have helped us understand each other, enabling us to overcome challenges and grow up together. I am so grateful for Xumiao's consistent presence by my side, offering invaluable emotional support. He is indeed a precious gift in my life.

Finally, I want to thank my parents, Wangping Liu and Zhi He, who always support my decisions and encourage me to earn the most advanced degree and chase after my dreams. Their guidance has empowered me to develop into a independent and brave female.

Go Blue!

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Videos have been widely adopted across various applications, highlighting the increasing importance of database management systems that can support video queries. However, achieving effective query processing in video database management systems is challenging due to factors such as the substantial size of video databases and the unstructured nature of video content. To address these challenges, I demonstrate that *video-specific algorithms that support query processing in video database management systems are essential for performance (accelerating video selection queries), policy (balancing competing query requirements), and explanation (supporting queries for real-world explanation)*. To support this statement, my dissertation focuses on four key parts: (1) building a new indexing mechanism that captures visual similarity for filtering items that are likely to satisfy the query predicate, (2) developing a video degradation-accuracy profiling system, helping administrators to choose an appropriate degradation setting for competing requirement trade-off in video analytics, (3) proposing a commonsense knowledge-enhanced indexing method, which initially constructs a lossy but inexpensive index and subsequently patches it to quickly identify query result candidates, and (4) implementing a causal inference system that uncovers confounding variables within images to solve confounding bias and compute more accurate average treatment effects (ATE).

CHAPTER I

Introduction

Videos are ubiquitous in our daily life — they can be extensively collected from various sources such as publicly deployed surveillance cameras [62], dashcams [31], online video platforms [4], etc., thus constituting a major part of contemporary data collection. Due to the rich and useful information contained in videos, they are steadily gaining popularity in a range of applications, including traffic monitoring [46], building automation systems [208], homeland security [92], self-driving techniques [21], robotics [193], and so on.

In these applications, video databases are subjected to various queries to serve diverse purposes. For instance, in order to improve a self-driving car model, 100 videos of crosswalks might need to be retrieved from a dashcam video corpus. Such a query can be written in SQL, with a crosswalk classifier integrated as a user-defined function (UDF) in the predicate:

```
SELECT * FROM dashcamVideos
WHERE HasCrosswalkUDF(video) = True
```

```
LIMIT 100
```

Another example is in traffic monitoring. The average number of cars per frame in surveillance videos during weekends might be computed for scheduling road construction work. In this case, the SQL query can be expressed as follows:

```
SELECT AVG(CarNumUDF(video)) FROM trafficVideos
WHERE time = "weekend"
```

Database management systems (DBMS) are crucial for maintaining data and supporting query processing. Traditional database management systems [168, 42] are primarily designed for relational databases, where data is stored in tables. In these tables, each row represents an entity instance and each column represents an attribute value associated with this instance. This relational data can be straightforwardly queried using standard operations such as selection, projection, join, grouping, etc.

However, traditional database management systems are unsuitable for videos mainly because videos are unstructured data. The information and attributes of videos, e.g., the object categories contained in videos, are not explicitly stored in columns, thereby making video queries harder to process. To extract video information, UDFs might be required as shown in the above examples. While UDFs enhance the expressiveness of queries, they can adversely affect query performance. In the above crosswalk example, the query optimizer in traditional relational DBMSs or "big data" engines [185, 199] usually cannot understand the semantics of UDFs, so the video data needs to be scanned sequentially, as shown in the top half of Figure 1.1. In contrast, an index (e.g., B+ tree [34]) can be created on attribute values for relational data to accelerate the process. If such an index is designed

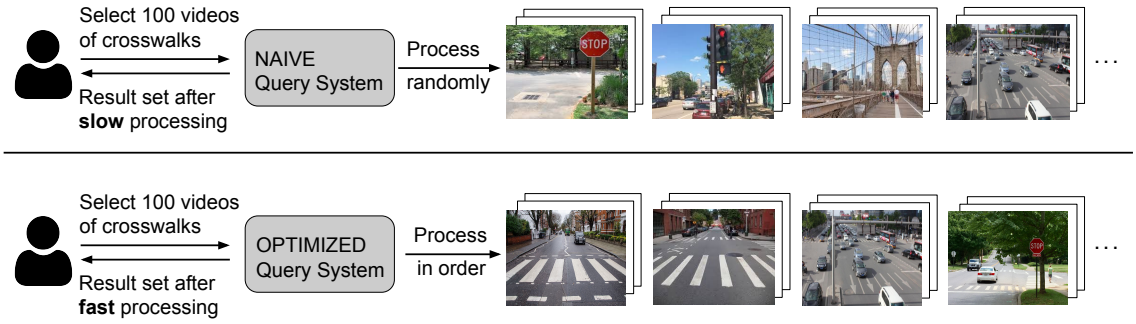


Figure 1.1: A comparison of video query processing with and without optimization

for video data, the satisfying videos can be easily selected as depicted in the bottom half of Figure 1.1. Moreover, these UDFs for video analytics are typically neural network models [209, 70, 9], yielding long inference time on each video, further exacerbating the problem.

In addition to the UDF-challenges discussed above, there are challenges due to privacy and system requirements, and intersections with related concerns, such as commonsense knowledge and causality, as we discuss further in Section 1.1. Consequently, a new video database management system with video-specific query processing techniques is desired.

In this dissertation, we present our work on supporting query processing in video database management systems. In Section 1.1, we present the challenges and opportunities in video query processing. Next, in Section 1.2, we summarize our contributions to this research area in the dissertation — we propose four different algorithms/systems for improving video query performance or supporting the process of new query types for visual content. Finally, we provide an outline of our work presented in this dissertation in Section 1.3.

1.1 Challenges and Opportunities

There are four main challenges that make video query processing difficult:

The large size of video databases — Typically, popular video databases [4, 89, 99, 31] contain a substantial amount of videos, partially due to how easy it is to collect videos nowadays. For instance, from a single 30 fps surveillance camera, more than 2.5M video frames can be collected within one day. If this camera records 1080p videos, each video frame amounts to approximately 6MB, resulting in a total of around 15TB of frames each day. On online media platforms, such as YouTube, over 500 hours of videos are uploaded every minute [118]. These tremendous video databases present a new challenge — processing these videos consumes a significant amount of time. In addition, large databases require the support of adequate storage space, high bandwidth, and sufficient power, but there are always system constraints [33, 11].

Privacy protection requirement in videos — Cameras are powerful sensors that can easily capture private information, such as facial imagery, license plates, etc., which needs to be protected. This information can raise public concern because it might be leaked during the shipment of video off-camera or during malicious query execution. There are legal regulations regarding privacy protection that must be adhered to during query processing. For example, according to the EU General Data Protection Regulation [180], face blurring is required when any closed-circuit television (CCTV) footage is shared with a third party.

Information extraction from unstructured video data with UDF — Unlike traditional relational data that is well-organized in tables with explicit attribute values, video data is unstructured with hidden information that needs to be extracted by tools. With the development of computer vision techniques, neural network models are widely adopted to

extract video content, such as object detection [182], action recognition [205], etc. They can be integrated into video queries as UDFs, but the inference time of a model is long compared to math operators in traditional SQL queries, yielding extremely long query processing time. For example, the processing speed of YOLOv7 [182], the state-of-the-art object detection model, is only 56 FPS when achieving 55.9% AP. Due to the unknown semantics of UDFs, traditional optimization methods (e.g., B+ trees [34]) are not applicable for accelerating the process of UDFs in the predicates. Moreover, recent research works dedicated to optimizing UDF execution [146, 149, 78] still make strong assumptions about the programming language, design, or possible semantics in the user-defined component.

Special query types for visual content — Basic SQL query types, such as selection queries and aggregate queries, are important in video queries as demonstrated in the above two examples. Due to the richness of information contained in videos, complex queries beyond the above basic types are also desired, for example, leveraging external visual content to enhance the accuracy of causal inference, or understanding object interaction in video scenes and reasoning an event [200]. Unfortunately, current video database management systems are unable to execute these special query types.

To build practical video database management systems that provide powerful query processing features, there are many potential directions — designing novel index mechanisms, developing video query languages, designing video-specific storage methods, supporting multiple video types, maximizing the use of limited resources, etc. Our research focuses on four specific opportunities in this area:

- The opaque filter query, a query with a selection predicate that is implemented with a UDF, is an important class of video queries in machine learning and data science

workloads, for example, filtering video frames. Even though the predicate's semantics are unknown to the optimizer, it is likely to filter similar video frames. It is promising to design appropriate indexes to accelerate the processing.

- In video analytical tasks, governments have a range of policy goals — preserving privacy, reducing bandwidth use, and legal compliance — that may be obtained by degrading the video at some potential cost to analytical accuracy. It is beneficial to profile video degradation-accuracy for administrators to employ controlled intentional video degradation in order to balance competing goals.

- As a subset of opaque filter queries, we focus on video selection queries that select videos containing desired objects. In order to achieve efficient query processing, the semantic-level correlations in videos can be leveraged to quickly locate videos that are relevant to predicates.

- Besides pure record keeping, videos as general-purpose sensors may also be used for causal inference. Conducting causal inference using observational data often encounters confounding bias, where hidden variables distort causal relationships. The confounding variables may be observed in video frames, so it would be helpful to integrate causal tables with external visual content.

1.2 Summary of Contributions

This dissertation addresses the above challenges by designing video-specific algorithms for supporting video query processing in database management systems with efficiency, accuracy, competing requirements, and potential query types considered.

The first work we present is VOODOO INDEXING, a two-phase method for optimizing opaque filter queries. Before any query arrives, the method builds a hierarchical "query-independent" index of the database contents, which groups together similar items. At query time, the method builds a map of how much each group satisfies the predicate, while also exploiting the map to accelerate execution. Unlike past methods, VOODOO INDEXING does not require insight into predicate semantics and does not require in-query model training. We implemented both standalone and SparkSQL-specific systems, plus experiments on more than 100 distinct opaque predicates on image data. VOODOO INDEXING can yield up to an 88% improvement over standard scan behavior, and a 79% improvement over the previous best method adapted from the research literature.

In the second work, SMOKESCREEN, we propose a video degradation-accuracy profiling model for the problem of controlling the appropriate amount of degradation in analytical video systems. It offers administrators a profile that illustrates the tradeoff between increased analytical accuracy and increased amounts of degradation. Computing the true tradeoff curves requires full access to the non-degraded video stream, so a primary technical contribution of this work lies in methods for accurately approximating the curves with only limited information. In addition, we propose a profile repair policy to further improve tradeoff curves' accuracy. We conducted experiments on two video datasets, two detection models, and four aggregate query types. Compared with competing methods, our upper bound estimation of analytical error can enable up to 88% more accurate tradeoffs.

Next, we present PAINE, an indexing 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 tested our prototype system on two video corpora, showing up to 97.79% fewer videos processed compared to baselines. Even the model constructed without any video content can yield a 75.39% improvement over baselines.

Lastly, we propose a novel approach to deal with the incomplete causal table by utilizing the information embedded in video frames, in order to uncover confounders in causal inference. Our system collects user-defined confounder names through a user interface that covers a wide range of image features and recommends domain-specific features. It employs a visual language model, GPT-4V, alongside lightweight classifiers based on CLIP embeddings for accurate feature extraction from images. Our pipeline creates the enriched causal table to compute the average treatment effect (ATE). Extensive testing across three causal graphs with varying levels of extraction difficulty demonstrates our system's robustness and effectiveness in identifying confounders from images and quantifying causal effects, yielding up to 66.7% more accurate ATE.

1.3 Dissertation Outlines

The rest of this dissertation is organized as follows:

- In Chapter II, we present the research background and discuss previous works related to the dissertation.
- In Chapter III, we present a two-phase index method, VOODOO INDEXING, for optimizing opaque filter queries.
- In Chapter IV, we present a video degradation-accuracy profiling model, SMOKESCREEN, to control the appropriate amount of intentional degradation in analytical video systems.
- In Chapter V, we present a novel index mechanism, PAINE, for optimizing video selection queries that select desired videos containing target objects.
- In Chapter VI, we present a new approach to mitigate confounding bias in causal inference by uncovering potential confounders from video frames.
- In Chapter VII, we conclude this dissertation and discuss potential avenues for future research.

CHAPTER II

Background and Related Work

In this chapter, we give a brief overview of the research background and related work. The discussion is organized around four research domains: *video data management*, *query processing in relational database management system*, *video analysis in computer vision*, and *causal inference*.

2.1 Video Data Management

A variety of video database management systems are emerging nowadays. They make the data management process more suitable for video data and improve the performance. Most of the previous works focus on query processing efficiency. They apply shallower neural network models than full models in predicates [85, 84], select appropriate input video settings (e.g., the frame resolution and frame rate) [8, 80, 17, 85, 204], propose probabilistic predicates to prefilter items [115], construct approximate indexes of possible object classes [73], enable parallel processing for large-scale video analysis [114, 140], design new tile layouts for efficient video decoding [38], or materialize and reuse user-

defined functions' results for exploratory video analytics [190]. Our works, VOODOO INDEXING and PAINE, fall into this category, exploring the index opportunity to accelerate query processing without error tolerance. Other previous works also design systems for specific video types, such as virtual reality videos [64], and develop video compression techniques [122]. Our works, SMOKESCREEN and our final project, pursue new directions — profiling video analytical accuracy under video degradation to balance competing goals, and leveraging images to address the issue of confounding bias for causal inference.

2.2 Query Processing in Relational Database Management System

In this section, we mainly discuss three aspects in RDBMS that are related to our work:

2.2.1 Query Optimization

Query optimization based on indexes — Traditional index structures [34, 121] have been widely applied in modern database management systems [129, 126, 41] for query optimization. Some works have designed novel index techniques. Database cracking [76] gradually cracks databases for building indexes according to users' queries. Learned index structures [96] adopt machine learning models for index construction in order to reduce time. However, these indexes are not applicable to video and other unstructured data.

UDF optimization — Database researchers have been working on optimizing queries with UDFs for decades [69]. The main approaches can be divided into three categories. The first one is to analyze the semantic of UDF code so as to leverage traditional optimization techniques, such as sub-query optimization [146], operator reordering [149, 74, 75] and index selection [78, 30]. However, due to the opaqueness of UDF code, these meth-

ods are not applicable to all the UDFs. The second one is to create new predicates for pre-filtering without careful semantic analysis [115, 8, 84, 85]. These works rely on historical queries, otherwise they need to train binary classifiers at cold start which stage can not be optimized. The third one is to sample data from indexed groups by learning mapping between UDF and groups at query time. Zombie [7] leverages this idea for feature engineering acceleration.

Query optimization with machine learning — With the development of reinforcement learning [169] and deep learning [104], machine learning methods have been popular in improving traditional query optimization methods. These works can help with join ordering [97, 177], subquery representation [135], cardinality estimation [91], selectivity estimation [102], index structures [96], and query hint selection [119].

In two of our works, VOODOO INDEXING and PAINE, we design novel indexing mechanisms that utilize machine learning models to optimize video queries with UDF.

2.2.2 Data Management with Privacy and System Requirements

The privacy-preserving problem in data publishing has been explored, summarized by [52]. Numerous models are proposed for guaranteeing k-anonymity [156, 106], ϵ-differential privacy [43, 32], etc. Apart from the non-interactive data publishing, studies have integrated the privacy protection into the query processing engine [19, 175]. For the system requirements, such as bandwidth management, energy saving, and storage capacity limitation, popular solutions include database compression [57, 152] and adjusting hardware and software configurations [178]. In comparison, our system SMOKESCREEN is suitable for video data with all of the above requirements and reveals how analytical

accuracy changes with these requirements.

2.2.3 Approximate Query Processing

Approximate query processing (AQP), aiming to approximate aggregate query answers in online analytical processing, has been researched for decades [29]. AQP methods comprise two categories: online aggregation and offline synopsis generation [110]. Works about online aggregation [68, 132, 5, 203, 28] select samples online to estimate the answers of aggregate functions, such as `COUNT`, `SUM`, and `AVG`. The estimation performance can be further improved by recent developments in concentration inequality [16, 123, 154, 93] which have been used in many areas, such as solving the multi-armed bandit problem [113]. Methods about offline synopsis generation [150] can be applied to more aggregate functions but require prior knowledge. Other than the above distributive and algebraic aggregate functions, holistic aggregate function approximation, such as `MEDIAN` and `RANK`, mainly rely on summary statistics [53, 163, 61, 60]; only some of the works are based on sampling [108, 117]. These algorithms can be applied for profiling video degradation-accuracy, but our estimation in SMOKESCREEN is more accurate.

2.3 Video Analysis in Computer Vision

With the increasing popularity of videos, video analysis is an important research direction in computer vision. Effective models and pipelines have been designed for various applications: object detection [182], action recognition [205], video segmentation [189], virtual reality video processing [107], human pose estimation [137], scene graph construction [82], etc. These video models are typically neural network models [104] that

are composed of multiple various kinds of layers (e.g., convolutional layers). After learning the weights of computation, these models can transform raw videos to desired results (e.g., the contained object categories). Nevertheless, powerful neural network models are usually along with long inference time.

Commonsense knowledge, the facts that are expected to be known by all humans, has been applied to many computer vision tasks in order to achieve human-level performance. It is collected and modeled by knowledge graphs (e.g., WordNet [125], ConceptNet [166], Wikidata [181], and CSKG [77]), natural language corpus (e.g., NELL [27] and OMCS [165]), and multiple choice QA problems (e.g., VCR [200] and SWAG [201]) through crowdsourcing, mining and other advanced methods. Commonsense knowledge can be utilized to improve the accuracy of object detection [44, 143], action recognition [54], and image classification [120], and enable visual commonsense reasoning [200].

2.4 Causal Inference

The development of modern causal inference methods have been ongoing for decades. Two well-known frameworks, Pearl's graphical causal model [138] and Rubin's potential outcome framework [153], provide formalized approaches to estimating causal effects. Because confounding bias is a crucial issue in causal inference, data integration for addressing missing confounders has been investigated [196, 162]. Integrating causal tables with different types of data presents distinct challenges. For tables, previous literature has supported efficient correlated column discovery and table join [157], combination of randomized clinical trial (RCT) with observational data when the primary outcome cannot be measured in RCT [13], and incomplete data sample combination [59]. Some works

also mine confounding attributes from knowledge graphs [195]. When it comes to unstructured data, previous research generated term frequency representations [202] and designed causally sufficient document embeddings [179] to uncover confounders from text, but these techniques cannot work when images confounds the causal inference.

While visual information was not examined for confounder identification, as a rich and popular data source, it has been explored in the field of causal inference to achieve cognition-level visual understanding. Recent works studied causal discovery among environmental and object variables in physical systems directly from videos [111], presented Bayesian grammar model for high-level causal induction from visual data [50], and embedded structural causal models into machine learning models to learn visual representations as causal variables [158]. In addition to extracting the explicit causal relationships for constructing causal graphs, there are also research projects about commonsense reasoning on visual data. These works designed algorithms to answer multiple-choice reasoning questions [109, 174], different types of causal questions [194], and give rationales to justify answers [200].

CHAPTER III

A Method for Optimizing Opaque Filter Queries

3.1 Introduction

Opaque filter queries constitute an important workload for processing unstructured data like videos. In these queries, LIMIT clauses commonly occur as users navigate massive databases. For example, an engineer trying to improve a self-driving car might want to retrieve 100 videos of crosswalks from a large database of dashcam video. She could train an classifier that detects crosswalks, package it as a user-defined function (UDF), and then run the following SQL query:

```
SELECT * FROM dashcamVideos
WHERE HasCrosswalkUDF(video) = True
LIMIT 100
```

A naïve but traditional method [101] for these queries is to scan the whole database, apply the UDF to every row, and filter the result set according to the UDF's boolean answers. Obviously this method is time-consuming when the UDF runtime is long and the

database is large. Canziani, *et al.* showed that the deep model inference time per image is at least 5 milliseconds and can already reach up to 200 milliseconds [25]. Moreover, extremely large video databases are easy to collect, not only from online collections but via the growing class of video applications; a single 30fps camera can obtain more than 2.5M frames a day.

Query optimization methods are crucial for good performance, but most have historically relied on semantic understanding of the query, which can pose a challenge when the query includes user-defined components. Recent research into optimizing UDF execution has tried to reduce query time through transforming or reordering UDFs for semantic equivalence [146, 149] and selecting indexes by code inspection [78]. However, these techniques remain limited, largely because of strong assumptions they make about the UDF's programming language, design, or possible semantics. These methods cannot help in the example below.

EXAMPLE 1. *Harry is a self-driving car engineer whose job is to debug car behavior under specific failure scenarios. In this case, he needs to identify 100 examples of crosswalks in the video database, but in general there are always new scenarios to debug, so Harry cannot rely on a one-time video frame labeling step; instead, he must provide novel trained classifiers as user-defined predicates and run opaque filter queries. Unfortunately, all real-life systems, whether relational or "big data" engines such as Hadoop [185] and Spark [199], execute these queries with a simple scan. Even worse, the inference time of each classifier invocation is long, making him wait a long time.*

Technical Challenge — Existing methods cannot optimize our opaque filter query workload for two core reasons, both tied to the fact that the selection predicate is not

only a UDF, but most likely a trained deep network. First, the selection predicates have semantics that are unclear to the database system and cannot be easily analyzed. As a result, it is not clear to the optimizer when a possible query optimization would be safe or effective. Second, the training procedure that yields the UDF selection predicate is likely an ongoing process, which constantly yields updated and novel predicates. As a result, it is not feasible to simply apply the predicate once to the entire database.

There has been some work on query systems for images and videos [8, 84, 85] that do not have full semantic information about the data, but still save time as many traditional indexing methods do: by avoiding processing irrelevant data. Unfortunately, these methods require expensive and error-prone online model retraining [8, 85].

Our challenge is to build an optimization method that has no access to detailed UDF semantics and avoids use of online model retraining.

Our Approach — In this paper, we propose a new mechanism for accelerating opaque filter queries that we call VOODOO INDEXING. Like many optimization methods, it obtains speedups by avoiding processing irrelevant data that will not be returned to the user. Unlike past methods for optimizing UDF-based queries, VOODOO INDEXING does not require any semantic insight into the UDF code. Further, VOODOO INDEXING does not require an online model retraining procedure.

VOODOO INDEXING's basic mechanism works in two phases. At index time, it clusters records into *index groups* so that similar objects are physically grouped together. At query time, it simultaneously *builds* a model of the rate at which different groups' records satisfy the UDF predicate, and also *exploits* the model to oversample records from the high-satisfy-rate groups. (Correspondingly, the method undersamples from groups where

the UDF is satisfied comparatively rarely.) VOODOO INDEXING adapts this approach from the ZOMBIE system, which was originally designed to support feature engineering tasks [7]; ours is the first indexing mechanism with this two-phase architecture for video analytics.

A core technical difficulty lies in quickly identifying and reading from the most-productive regions of the indexed database (that is, the database elements most likely to satisfy the UDF selection predicate). In addition to employing a multi-armed bandit policy [15] for deciding among index groups, VOODOO INDEXING further adds a *hierarchical sampling* procedure. The core observation is that drawing a sample record from one index group yields evidence about the sampled group as well as weaker evidence about other similar groups. By exploiting this weak but plentiful evidence at query time, VOODOO INDEXING can identify useful groups with very few samples, thereby yielding substantial performance gains beyond a naive adaptation of ZOMBIE method.

EXAMPLE 2. *In order to decrease waiting time, Harry employs voodoo indexing. At index time, voodoo indexing clusters together visually-similar frames from the dashcam video database. At query time, the system samples from a variety of index groups, sending each to the crosswalk detector UDF for evaluation. It finds that one index group contains a high fraction of data items that pass the selection filter, and oversamples from this group, quickly yielding 100 results.*

Under this design, the system can identify many records that satisfy the UDF predicate early in the query execution process. It is intended to be most effective on "Goldilocks" LIMIT sizes that are neither extremely small nor extremely large. Extremely small result sizes will not give the system much opportunity to exploit the query-time information it

collects, while extremely large result sizes will require the system to draw records from even very low-probability index groups. Fortunately, these mid-sized LIMIT sizes should be typical for the analytical and machine learning workloads we expect to see alongside UDF-based predicates.

Contributions — Our main contributions are as follows:

- We propose a novel method, VOODOO INDEXING, for optimizing opaque filter queries. We model query processing as a hierarchical multi-armed bandit problem [198] and design an efficient algorithm for sampling.
- We evaluate our algorithm with real-world image databases, MNIST and ImageNet, showing that VOODOO INDEXING can yield up to an 88.2% improvement over standard scan, and a 79.0% improvement over adapted ZOMBIE.
- We prototype our method in a standalone system as well one integrated with SparkSQL, and test its functionality on MNIST, showing up to 86.6% improvement.

3.2 Problem Formulation

In this section we introduce notation and a formal model for the opaque filter query optimization task, as well as for our proposed solution framework. All of the notation is listed in Table 3.1.

3.2.1 Problem Definition

An **opaque filter query** is characterized by a 3-tuple of parameters (D, F, k). The raw data D is a large dataset that needs to be filtered, such as the dashcam video dataset in the

Parameter	Description	Example
D (Given)	Raw dataset	Driving dashcam videos
F (Given)	UDF predicate	Crosswalk detector
k (Given)	LIMIT number	3 items
I (Designed)	Clusters	Data clusters created by K-Means or similar alg
T (Designed)	Dendrogram	A balanced tree describing data similarities
μ (Designed)	Reward	Crosswalk detector returns True
n (Designed)	# samples	14 items are sent to F

Table 3.1: Frequently used notation in VOODOO INDEXING

above example. The function F is user-defined function used as the selection predicate. It can be written using any language or programming framework. For example, it can be a TensorFlow or a PyTorch model, or even a hand-written piece of Python code. Items in dataset D will be selected when function F returns true. The LIMIT k is the number of items that the query should return. Usually k is driven by application concerns, such as the screen size, or the amount of time available to the user, or the required number of training examples for a downstream learning pipeline.

The value n is the number of items in D that are sent to F at query time for evaluation, in hopes of finding k items where F returns true. We expect that F will be time-consuming to run, so the runtime of the query is substantially driven by the time needed to run n invocations of F. We can now define the optimization problem as follows:

PROBLEM 1: *Given input dataset D and user-defined function F, find k items that satisfy F while minimizing n.*

3.2.2 UDF Optimization Strategy

In order to solve the above problem, our model aims to intelligently choose the items in D that are evaluated with F, such that the function returns false as infrequently as possible. At index time we organize the dataset D into a similarity structure. For example, the structure can be a dendrogram T in which the leaves are clusters of items, notated as I. At query time, our approach estimates the average utility of the clusters in order to select the useful ones to answer user's query. For each cluster it accumulates a reward μ, computed as the fraction of records that return true when given to F. We can now describe our algorithmic task as follows: *Design a similarity structure for raw dataset D, as well as a selection strategy, that will solve the above optimization task.*

3.2.3 Design Considerations

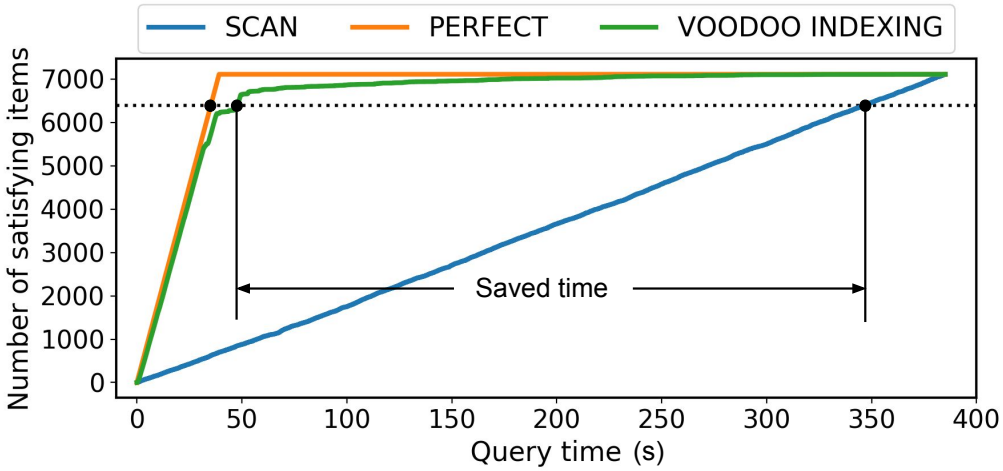


Figure 3.1: Typical execution curves for PERFECT, VOODOO and SCAN

Here we discuss some considerations in formulating a design.

LIMIT Size — Our goal is to optimize opaque filter queries with a LIMIT value k. Figure 3.1 shows how quickly different approaches can satisfy an opaque filter query on MNIST data [105]. The x-axis describes query runtime, while the y-axis indicates the number of data items retrieved by the system for which F returns true. The dotted horizontal line indicates the query's LIMIT value (in this case, 90% of the total database size). This figure is derived from executing a real query. The PERFECT line indicates an ideal (but probably unrealizable) system that processes solely data items that return true when sent to the UDF predicate F. The VOODOO line indicates the method described in this paper.

Note that if the query's LIMIT value is extremely large or small, it is challenging for competing methods to improve over SCAN. Fortunately, we expect most k values to fall in the broad middle due to databases' large size. Experiments in Section 3.6.2 show that while our method is effective across LIMIT values ranging from 10% to 90% of the satisfying item size, it shows best results for middle-range values.

UDF Behavior — We expect the UDF's runtime to be broadly consistent with published runtimes of trained classifiers running on GPU hardware. The longer the UDF runtime, the more useful it is to minimize invocations of F. We expect the UDF to be accurate enough to serve the purposes of the user's query; the work here does not try to improve the UDF's standalone accuracy.

3.2.4 Illustrative Example

In this section we illustrate the behavior of three possible implementations of opaque filter queries: standard SCAN, the ZOMBIE method, and our proposed VOODOO INDEX-

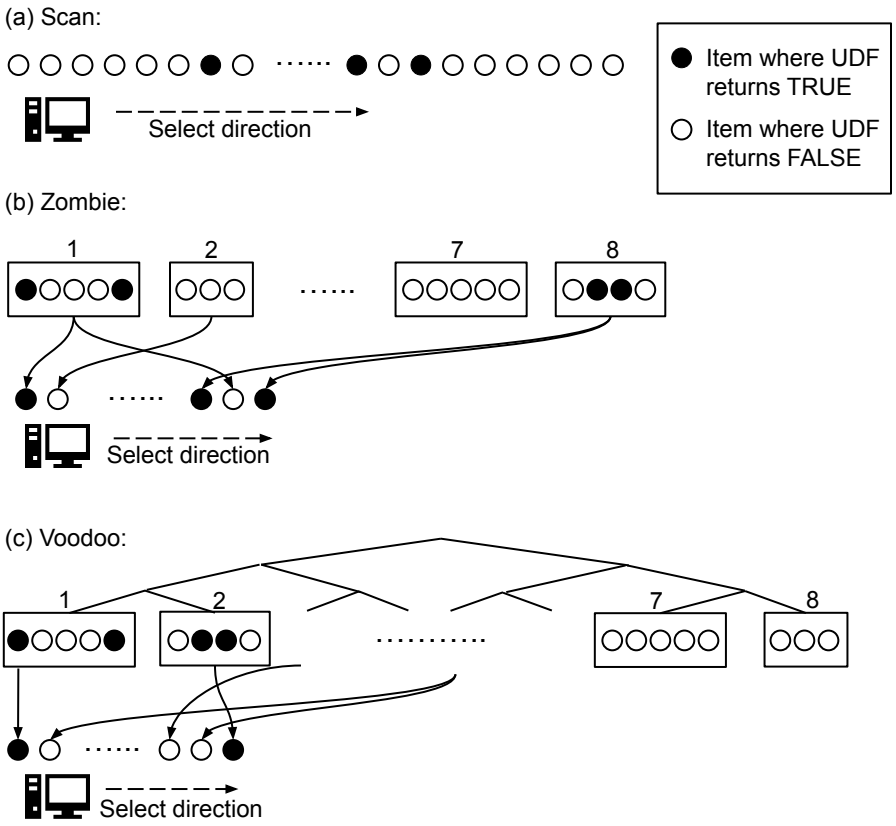


Figure 3.2: Three methods for the illustrative example

ING method.

Example 1: SCAN. *This example is shown in Figure 3.2 (a). Harry is now filtering crosswalk video frames by applying a learned classifier as UDF, and has indicated a LIMIT value of $k = 3$. There are 32 records in the database; 4 of them return True when sent to the UDF. In this case, the UDF needs to be applied 24 times in order to obtain 3 useful video frames.*

Now consider the following example about the adapted ZOMBIE method:

Example 2: ZOMBIE. *This example is shown in Figure 3.2 (b). Harry now tries*

utilizing a research query system that employs a method adapted from the Zombie algorithm. At index time, these data records are clustered into 8 index groups. Some have a large fraction of data items where F will return True. At query time, the system repeatedly draws each item from a chosen index group. By monitoring the rate at which index group yields a record where F returns True, the UDF only needs to be applied 18 times in order to obtain 3 useful video frames (a 25% reduction from standard practice).

Now consider our VOODOO INDEXING approach:

Example 3: VOODOO. *This example is shown in Figure 3.2 (c). At index time, the data records are again clustered into index groups. In addition, those clusters are placed in the leaves of a dendrogram, in which similar clusters are closer to each other in the tree. At query time, the system again repeatedly draws each item based on the dendrogram. In this way, the system can learn to ignore useless clusters (from cluster 3 through 8) with fewer accesses than in the above method. In this case, the UDF needs to be applied just 14 times in order to obtain 3 useful video frames (a 41.7% reduction from standard practice, and a 22.2% reduction from Zombie.)*

These scenarios show actual examples on small datasets. Our method can show larger advantages when applied to larger databases, as shown in Section 3.6.

3.3 Background: Multi-Armed Bandits

Before presenting our work for optimizing opaque filter queries, we will introduce some needed background about the multi-armed bandit problem.

The multi-armed bandit problem is from the area of reinforcement learning [23]. An

agent is confronted with a set of "one-armed bandits", or slot machines, each of which has a different probability of payout. The agent has a limited number of pulls to obtain payout from these machines, and could choose to spend those pulls identifying the best bandit (that is, exploring) or spend on the bandit currently known to be best (that is, exploiting). The goal of the problem is to balance these two motives to maximize the total payout.

The UCB algorithm [15], a formal way to simultaneously explore and exploit, is commonly used to solve this bandit problem. The upper confidence bound (UCB) of each slot machine's utility is composed of the current average reward (the exploitation term) and the one-sided confidence interval that may contain the true utility (the exploration term):

$$UCB_{a,t} = \mu_a + \alpha\sqrt{\frac{2\ln n}{n_a}} \tag{3.1}$$

where $UCB_{a,t}$ is the upper confidence bound of machine a at each time t, μ_a is the average reward of machine a, α is the exploration-exploitation balancing parameter, n is the overall number of inserted coins, and n_a is the number of coins inserted in machine a. According to the UCB algorithm, the machine that currently maximizes the upper confidence bound is always selected for the next action.

When we adapt this solution in our situation, each bandit corresponds to an index group, and each "pull" is the drawing of a data record from an index group. The reward at each time indicates whether the UDF predicate returns True for the chosen data record. We will discuss this bandit model in detail in the next section.

3.4 Algorithms

Now we describe our novel VOODOO INDEXING algorithm that helps users optimize opaque filter queries. In Section 3.4.1, we introduce a generalized version of ZOMBIE that we call ZOMBIE-G. ZOMBIE was designed to solve a particular feature engineering task, which can be seen as an example of the generic opaque query optimization problem. In Section 3.4.2, we propose the basic version of VOODOO INDEXING utilizing hierarchical multi-armed bandit algorithm. It samples from one cluster and sheds partial evidence about other clusters, leading to faster convergence on a good policy. In the remaining sections we discuss additional components that can benefit performance under particular situations.

3.4.1 Zombie-G

Overview — ZOMBIE-G contains two stages as shown in Figure 3.3: the indexing stage and the query processing stage. Its core idea is to construct a query-independent index before any query arrives, and then when executing queries select from high-payoff groups more frequently. Its structure is similar to two-phase operation in approximate query processing [55, 6]. The algorithm is shown in Algorithm 1.

Indexing — In the first stage, the raw dataset is re-arranged in a one layer index structure. A basic method is to cluster them into a set of index groups I as shown in line 2 of Algorithm 1. This initial process tries to group similar data together and divide dissimilar data into different groups. For example, crosswalk video frames may be clustered together.

The clusters are not built with a particular query in mind: they simply group together data objects that are generally similar. We employ general-purpose clustering methods,

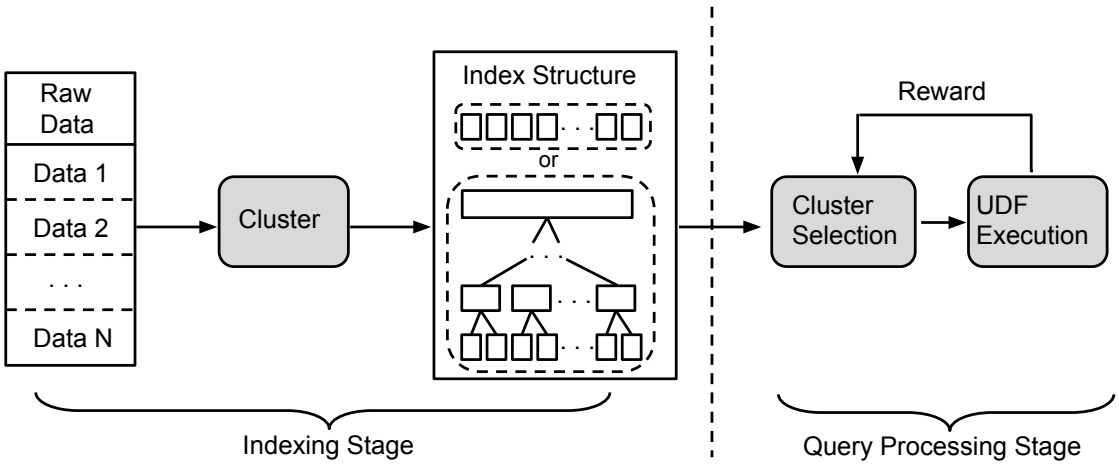


Figure 3.3: The architecture for opaque filter query optimization

such as K-Means. We also use task-independent features, such as pixel values. Task-independent features are widely used; for example pixel value vectors as visual contents are correlated with digit classification [105], different object classifications [39, 98], indoor-outdoor scene classification [170], etc. If insight about the database domain or query workload is known at indexing time, it can be used.

An important hypothesis of this approach is that downstream UDFs will filter records using properties that are captured in the index groups. This seems reasonable; most UDFs are not purely random and general features are related with a wide range of filter properties. If the UDF's classification and the index groups are correlated, then the query processing stage has a chance to identify these high-payoff groups. If the index happens to be totally unrelated with the UDF, ZOMBIE-G will be unable to identify high-payoff groups, so its performance will be similar to (but not greatly worse than) simple scan.

We will examine only "insight-free" cases in Section 3.6, in which there is no knowledge ahead of time of the database domain or query workload.

Query Processing — In the second stage, the goal is to identify and draw from index groups that contain larger numbers of data records where F returns True. We will treat a group's fraction of items where F is True as its payoff. The true payoff of each group is an unknown parameter, because the UDF is potentially new with each query. We must simultaneously sample from groups to figure out their payoff, as well as use this information to improve future index group choices.

This problem is similar to multi-armed bandit problem, the purpose of which is to learn from "one-armed bandit" and maximize total rewards. We use the UCB algorithm to solve the problem. The algorithm works as follows: the UCB of every group is calculated in lines 4-6, the formula of which is the same as equation (3.1) in the last section, except that a is the index for each group, n and n_a are the number of sampled records in total and for group a respectively. μ_a, the averaged reward of group a, is initialized to be 1 and updated as the system obtains counts during query execution. One item from the group with the highest UCB is selected in lines 7-8 (ties are broken by selecting randomly). This item is taken as the input of UDF F to determine if it should be filtered, and to obtain the reward μ in lines 9-12. Related parameters are updated in line 13 for the next round. The algorithm terminates when the result set size meets the user's LIMIT requirement or when all the data have been selected.

The ZOMBIE-G algorithm can be useful (as we show in Section 3.2.4), but it has some weaknesses. In particular, it can take a long time to identify high-payoff index groups, especially when the space of bandits is large. Executing the user's query will entail broad sampling from many clusters for a long time, thereby processing many data items that do not satisfy the filter predicate. Only after sampling and processing records from many

Algorithm 1: Zombie-G

Input: Raw dataset D, UDF F, LIMIT number k

$resultSet$ = [ ];
I = Cluster(D);
repeat
  for Every group a **do**
    $UCB_a = \mu_a + \alpha\sqrt{\frac{2\ln n}{n_a}}$
  end
  $bestIdx = \arg\max_a UCB_a$;
  $item = I(bestIdx)$.getNext();
  if $F(item)$ == True **then**
    $resultSet$.append($item$)
  end
  μ = rewardFromUDF($item$);
  Update $\mu_{bestIdx}$, n, $n_{bestIdx}$
until $|resultSet|$ == k or n == $|D|$;

Output: $resultSet$

clusters does the system find high-value clusters to focus on. As we will see below, our VOODOO INDEXING algorithm can do better.

3.4.2 Voodoo Indexing

The above algorithm treats each index group — each one-armed bandit — as entirely separate. But data items in different clusters still have a similarity relationship. Some clusters' contents are more similar to each other than to others, and a sample from one cluster may shed information about another. We can use this fact to better exploit the information we get from each sample — each "pull" of a bandit arm — and accelerate the agent's convergence to a good policy, thereby running the query much more quickly than ZOMBIE-G can. Authors have previously proposed a hierarchical bandit algorithm

[198], which mainly explores in a coarse low-dimensional space to greatly decrease the amount of unnecessary exploration and exploit fine-grained space to guarantee accuracy. This approach can potentially capture the correlations that exist between our data clusters. Different from the linear reward assumption and the feature vector representation in [198], we propose a new algorithm to leverages this coarse-to-fine hierarchical idea.

Overview — Like ZOMBIE-G, VOODOO INDEXING has the two-phase architecture shown in Figure 3.3. However, in order to leverage similarity between data clusters, we design a different index structure in the first stage and a different algorithm for item selection in the second stage. Algorithm 2 shows the VOODOO INDEXING algorithm.

Indexing — In the indexing stage, the one-layer index structure in ZOMBIE-G cannot capture the clusters' similarity. Instead, in VOODOO INDEXING, raw data are arranged into a hierarchical structure T in line 2. To be specific, a dataset is first organized in clusters according to the Euclidean distance between items. These clusters are added as leaf nodes to a dendrogram, and then connected according to the Euclidean distance between groups' cluster centers. Each internal node can be thought of as a "virtual cluster" that contains all of the records in its child nodes. Various cluster method, such as K-Means, and dendrogram construction methods, such as agglomerative hierarchical clustering, can be utilized in this stage.

Query Processing — How can we use this structure to exploit the similarity between groups? Our novel algorithm is inspired by hierarchical optimistic optimization (HOO) in [24]. HOO is an arm selection policy designed for continuous reward functions, which we have modified for our discrete situation. First, the upper confidence bound is calculated for every group a including leaf nodes and inner nodes at each time t in line 4-6. In contrast to

that previous work, inner nodes in the dendrogram only contain information of leaf nodes that have not yet been fully explored. Next, in line 7-10, the algorithm will choose a target cluster by starting at the root and repeatedly comparing the UCB of the current node's nonempty children, selecting the winner:

$$BestChild = \underset{c \in nonempty\ children}{\arg\max} UCB_c \tag{3.2}$$

until reaching the bottom or a full index group that has not been touched before. Then after selecting the index group and updating the result set in line 11-14, the reward from applying the UDF to the selected item is received in line 15, which is given by:

$$\mu_{item} = \begin{cases} 1, & \text{if this item satisfies the predicate} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Subsequently, parameters n, μ_a and n_a in the searching path are updated in line 16-19 in preparation for next UCB update. Crucially, this procedure allows a sample from one leaf index group to influence future behavior for all subtrees containing it. The loop is repeated until meeting the stopping criteria.

3.4.3 Dynamic Index Recovery

VOODOO INDEXING relies on the hierarchical index structure to better exploit sample information. However, this mechanism can only help if the dendrogram carries useful information. Dendrogram examples of different qualities are shown in Figure 3.4, where index groups that satisfy the UDF predicate 100% of the time are indicated in black, the

Algorithm 2: Voodoo Indexing

Input: Raw dataset D, UDF F, LIMIT number k

$resultSet$ = [ ];
T = Dendrogram(D);
repeat
  for Every group a **do**
    $UCB_a = \mu_a + \alpha\sqrt{\frac{2\ln n}{n_a}}$
  end
  $bestIdx$ = Root(T);
  repeat
    $bestIdx = \arg\max_{c \in nonemptyChildren(bestIdx)} UCB_c$
  until $n_{bestIdx}$ == 0 or $bestIdx \in$ Leaf(T);
  $item = T(bestIdx)$.getNext();
  if $F(item)$ == True **then**
    $resultSet$.append($item$)
  end
  μ = rewardFromUDF($item$);
  Update n;
  for Every group a on searching path **do**
    Update μ_a, n_a
  end
until $|resultSet|$ == k or n == $|D|$;

Output: $resultSet$

50% level is indicated in grey, and 0% is indicated in white. Even though every example tree is composed of the same clusters, not all of the trees are useful. In the good dendrogram, low-payoff clusters are concentrated in the right subtree, so only a few samples in the right subtree can tell the algorithm to decrease sampling frequency for these clusters. In the useless dendrogram, low-payoff clusters are intermixed with high-payoff clusters, so high-level internal nodes of the dendrogram provide almost no useful information. In the bad dendrogram, the low overall payoff of the right subtree will hide the actually best cluster, so the algorithm may actively avoid sampling from this high-payoff cluster.

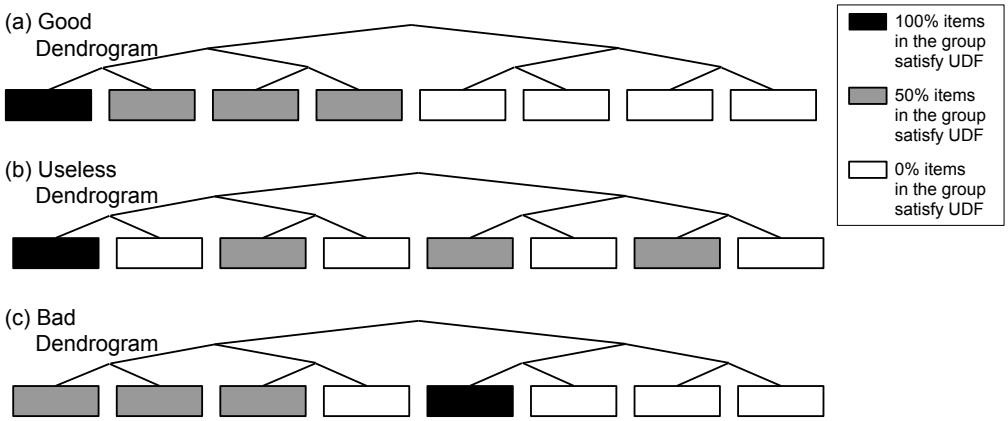


Figure 3.4: Dendrogram examples of different qualities

Without any evidence about the user's UDF filter, we can only use index-time information when constructing the dendrogram. Unfortunately, if this dendrogram does not group together clusters with similar filter behavior, a sample from one cluster does not give useful information about its sibling cluster, as with the last two dendrogram types above, and so VOODOO INDEXING does not yield speedups. It can potentially do worse than ZOMBIE-G or even a simple scan.

However, we note that as the user's query executes, we gain information about cluster

similarity that was not available at index time. It may be possible to use this information to dynamically construct a new dendrogram that will help later stages of VOODOO INDEXING execution. Algorithm 3 is our implementation of dynamic VOODOO INDEXING. It is the same as algorithm 2 in the indexing stage in line 2. However, in the second stage, rather than retain the initial dendrogram for the entire execution, this algorithm may switch to a new one. If the algorithm detects that it is finding good data items less efficiently, possibly due to the inappropriate dendrogram structure, it holds a "contest" to determine whether it should switch to a new dendrogram. Two dendrograms compete in the contest: one is the initial dendrogram computed at index time, and the other is generated by sorting the leaf clusters by average observed rewards. A certain number of items are sampled from these two dendrograms separately by executing VOODOO INDEXING, and the accumulated rewards are compared in lines 7-14. The algorithm will switch to the better dendrogram, until either a new contest is found to be needed, or the query terminates.

Although drawing items from the discarded dendrogram during the contest represents extra overhead, the predicate answers for these items can be stored for future sampling to decrease UDF running times. This is a greedy algorithm that makes the current optimal decision when unsatisfactory performance occurs. Even though it is possible that the greedy version may lead to a local optimum, comparing against the initial dendrogram guarantees that the dynamic version would not be worse than basic VOODOO INDEXING.

The dynamic recovery algorithm requires setting a few parameters to determine the frequency of running each contest, and how long each contest should be run. In order to formulate a policy that permits the system to set these parameters automatically, we tested a wide range of parameter values on various workloads. The policy sets the balancing

parameter α to a larger (1) value for smaller datasets (size $\leq$ 100,000), and smaller (0.1) for larger datasets (size $>$ 100,000). It runs a contest no more frequently than after processing a 5% chunk of the database. It runs a contest if the last 5% of data yield UDF-satisfying data records at a rate of less than 80% of the previous 5% chunk. Running a contest entails processing 1% of the database through each candidate dendrogram.

Algorithm 3: Dynamic Voodoo

Input: Raw dataset D, UDF F, LIMIT number k

```
1  resultSet = [ ];
2  T = Dendrogram(D);
3  repeat
4  |   while Performance not get worse do
5  |   |   resultSet.append(BasicVoodoo(T))
6  |   end
7  |   reward_1 = ContestReward(Initialize(T));
8  |   reward_2 = ContestReward(Sort(T));
9  |   if reward_1 > reward_2 then
10 |   |   T = Initialize(T)
11 |   end
12 |   else
13 |   |   T = Sort(T)
14 |   end
15 until |resultSet| == k or n == |D|;
```

Output: $resultSet$

3.4.4 Scan Failover

A good dendrogram can help accelerate data selection. However, sometimes it may not be possible to identify an effective dendrogram. In some cases it may unfortunately be true that our method's basic hypothesis — that the UDF predicate yields results correlated with one or more the index groups — is false. There is no correlation between what the

user wants and our index structure. In these cases, the best thing we can do is to switch over to standard scan in order to avoid the additional overhead associated with our method. Therefore, we design a new component to detect when this failover is appropriate.

After running query processing for a certain period of time (we set it to be 10% of dataset size) our detector works to compare the current ratio of the number of items where F returns True to number of processed items against that in simulated scan performance. The only challenge is how to determine a simulated scan sample size to minimize the sample overhead while accurately reflecting scan's expected performance. Confidence intervals can be leveraged to tackle this problem. We assume every item's predicate answer obeys the Bernoulli distribution $B(p)$ identically and independently, in which p is the true ratio in the scan process that we want to estimate. Given the expectation p and standard deviation $\sqrt{p(1-p)}$, we can get the convergence property according to the central limit theorem [103]:

$$\frac{\sqrt{n}(\hat{p}-p)}{\sqrt{p(1-p)}} \xrightarrow{d} \mathcal{N}(0,1) \tag{3.4}$$

where $\hat{p}$ is the sample average to estimate p. Therefore, the sample size under 95% confidence level should be:

$$n = \frac{1.96^2 p(1-p)}{(\hat{p}-p)^2} \tag{3.5}$$

where 1.96 is the z-score. In our system, the margin $|\hat{p}-p|$ in the denominator is set to be half of p for small p ($< 10\%$) and one fifth of p for large p ($> 10\%$). We hypothesize that the true ratio p equals the ratio $\overline{p}$ in VOODOO INDEXING to determine the sample size n. After n scan samplings, the confidence interval $[\hat{p} - 1.96\sqrt{\frac{\overline{p}(1-\overline{p})}{n}}, \hat{p} + 1.96\sqrt{\frac{\overline{p}(1-\overline{p})}{n}}]$ is supposed to contain the mean $\overline{p}$ under 95% confidence level. If $\overline{p} < \hat{p}$, the midpoint

of the interval, it is highly probable that the true parameter $p \geq \overline{p}$, which means simple scan sampling is not worse than VOODOO INDEXING, so we stop our VOODOO INDEXING algorithm and switch to scan.

3.4.5 Batch Mode

In the previous algorithm, we selected only one item at each time, which fits the tuple-level iterator model that is common in database software. However, this method leaves unexploited advantages of batch-loading multiple items at once into GPU memory. Batch loading is a common low-level optimization for GPU workloads [186]. It has been widely used in neural network training in order to improve memory utilization and the parallel efficiency of matrix multiplication. We would like to leverage this optimization in our algorithm. We therefore slightly modify VOODOO INDEXING during query processing to draw a batch of items from the index groups with the highest UCBs in each round, and feed into the UDF predicate a batch at a time.

The disadvantage of batch mode is that UCBs cannot be updated after each choice: the top picks in each round might be out of date. More concretely, in a particular batch of sampled items, the second pick cannot benefit from the learned information from processing the first pick. Nevertheless, batch mode can still help reduce runtime dramatically, which will be discussed in Section 3.6.

3.5 System Prototypes

We embodied our VOODOO INDEXING algorithm in two prototype systems: a standalone Python implementation and an implementation integrated with SparkSQL. The for-

mer allows us to conduct detailed experiments on the algorithm's behavior, while the latter allows us to demonstrate VOODOO INDEXING in a real-life working database system.

At a high level, our SparkSQL implementation allows users to ingest dataset files like they would for any standard SparkSQL database. Before any queries arrive, our system clusters the data elements into index groups. The indexed dataset now contains an additional field that identifies, for each row, an index group ID.

Standard SparkSQL behavior when processing an opaque filter query is to use an RDD iterator to scan the dataset in sequence, applying the UDF selection predicate to each element, and deciding whether to put the row into the result set. In contrast, we implemented a custom Scala SparkSQL operator that repeatedly interacts with a VOODOO INDEXING backend, which tells the custom operator which group ID it should read next. This backend implements the index group choice procedure from Algorithm 2 in Section 3.4.2. Each step of this iterator also updates the VOODOO INDEXING model about whether the last index group decision yielded a positive or negative result from the UDF predicate. (That is, it updates whether the bandit arm pull yielded a reward.)

We also implemented a new SparkSQL operator to perform correct but basic early stopping in LIMIT queries, regardless of whether the data is being processed in scan fashion or using VOODOO INDEXING. (The standard plan generated by SparkSQL when performing LIMIT queries will not perform basic early stopping even when the LIMIT is satisfied, but rather will scan the entire dataset and then throw away results beyond the LIMIT.)

3.6 Experiments

We evaluated four core claims about VOODOO INDEXING:

1. **Voodoo indexing performance is better than competing methods.** VOODOO INDEXING can execute opaque filter queries more quickly than competing methods. This holds true across a range of real-world image datasets, as well as a range of user-defined selection predicates (that is, trained UDFs) (Section 3.6.2).

2. **All voodoo extensions yield benefits.** The three extensions to VOODOO INDEXING — dynamic index recovery, scan failover, and batch mode execution — yield better results than the naïve VOODOO INDEXING algorithm (Section 3.6.3).

3. **Voodoo is effective across many scenarios.** VOODOO INDEXING can work over a very wide range of plausible real-world scenarios, which we demonstrate using a range of synthetic UDFs and clusterings (Section 3.6.4).

4. **Voodoo works in real systems, like SparkSQL.** We show our approach yields direct runtime benefits when integrated with the real-life SparkSQL system (Section 3.6.5).

3.6.1 Experimental Setting

In this section, we describe our workloads, baseline methods, evaluation metrics, and experimental configuration.

Workloads — We evaluated our algorithm on multiple workloads. Each consists of a popular image dataset, plus a trained network to provide the UDF filter.

The **MNIST database** [105] contains 70,000 handwritten digit images. We trained a 10-digit image classifier, using the ResNet-18 [66] architecture modified to fit MNIST. It was trained from scratch, achieving 98.7% accuracy. It takes around 5ms to pro-

cess one image on a GPU. We used this classifier to build 10 different boolean UDFs: `isDigitZeroUDF()`, `isDigitOneUDF()`, and so on. One such query takes the form:

```
SELECT * FROM MNIST
WHERE isDigitZeroUDF(image) = True
```

When using **ImageNet data**[39], we use the Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) [155] dataset, comprising 1,281,167 images in 1000 categories from ImageNet. It is widely used for image classification training. A quarter of the images were randomly chosen for our queries. We used the pre-trained ResNeXt-101 (32 $\times$ 8d) [188] model in PyTorch [136]. We also trained 80 binary classifiers for 20 randomly chosen categories (Table 3.2), varying the model architecture (ResNeXt-101 (32 $\times$ 8d) and VGG-19 [164]), learning rate (0.1 and 0.01) and optimizer (SGD and Adam). These models achieve 97.6% accuracy on average on the sampled ImageNet dataset. It takes the ResNeXt-based model about 36 ms and takes the VGG-based model about 11 ms to process a single image on a GPU. Eight of the trained models never returned true for any tested input, so we removed them from the evaluation. We used these 72 trained classifiers and the 20 pretrained ones to build 92 boolean UDFs. One such query takes the form:

```
SELECT * FROM ImageNet
WHERE isBakeryUDF(image) = True
```

Baselines — We evaluated against two baselines.

- **Scan** — All DBMSes we are familiar with, including SparkSQL and PostgreSQL [128], will execute opaque filter queries with a simple scan, applying the UDF selection predicate to each row of data in sequence.

Category	Label ID	Category	Label ID
American lobster	n01983481	Lipstick	n03676483
Dugong	n02074367	Megalith	n03743016
German shepherd	n02106662	Pay phone	n03902125
Greater Swiss mountain dog	n02107574	Pop bottle	n03983396
Great dane	n02109047	Schooner	n04147183
Fly	n02190166	Screw	n04153751
Bakery	n02776631	Spotlight	n04286575
Cuirass	n03146219	Submarine	n04347754
Fountain	n03388043	Washer	n04554684
Honeycomb	n03530642	Corn	n12144580

Table 3.2: The 20 randomly-selected ImageNet categories we used to train UDF models.

- **Zombie-G** — This is a generalized method from the ZOMBIE system, introduced in Section 3.4.1.

Evaluation Metrics — We computed two core metrics to show the benefit of VOODOO INDEXING: the total number of items that were processed by the UDF predicate, as well as total query time. The first metric reflects directly what VOODOO INDEXING can change during execution. The second metric reflects our method's decisions, but also reflects the query's UDF runtime and our method's overhead. Although we report indexing time, we believe it is relatively unimportant compared to query time: the index structure is only computed once, then used for all queries that follow.

Experimental Configuration — We implemented our algorithm and all baseline methods in Python, and implemented SparkSQL-specific components in Scala. Experiments were run on a 64-core (2.10GHz) Intel Xeon Gold 6130 server with 512 GB RAM and 4 GeForce GTX 1080 Ti GPUs.

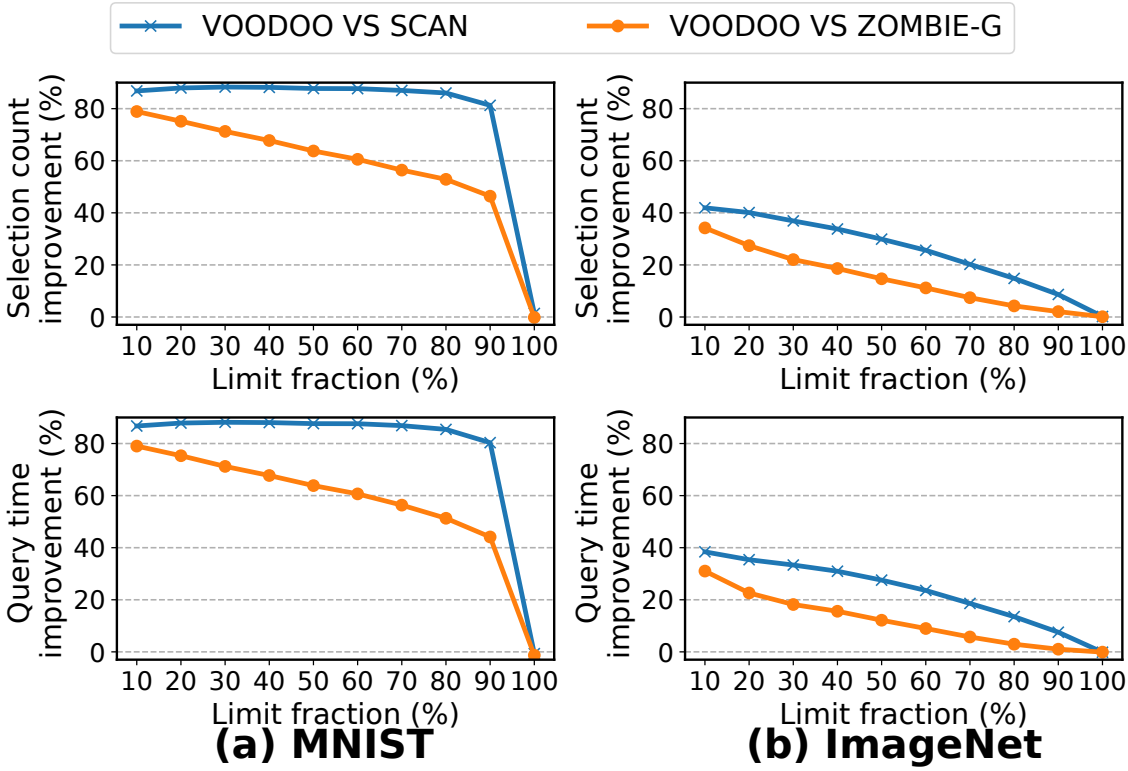


Figure 3.5: Overall voodoo performance gains on MNIST and ImageNet.

3.6.2 Overall Performance

Summary — Our method VOODOO INDEXING is better than competing efforts on a range of datasets and trained models. Over a range of query settings, VOODOO INDEXING shows average query time improvements over competing methods of up to 88.2% on MNIST and 38.4% on ImageNet.

Overview — We evaluated VOODOO INDEXING against ZOMBIE-G and SCAN on off-the-shelf image datasets. Before we executed any queries, the data was clustered as described in Section 3.4. We indexed each dataset just once — using generic pixel value features. MNIST images are small (a 28*28 grayscale image), so we used direct pixel

values to produce a length 784 feature vector; we resized the higher-resolution ImageNet images to grayscale 30*30 images for a feature vector of length 900. We clustered data using basic K-Means for MNIST (with number of clusters set to 1000), and Mini Batch K-Means [159] for ImageNet (with number of clusters set to about 2500). We built initial dendrograms in all cases using agglomerative clustering.

We tested full dynamic VOODOO INDEXING with scan failover. We did *not* use batch mode in these results, in order to make the various methods' behavior easier to understand. We cover batch mode performance in Section 3.6.3.3. We set all parameters according to the method in Sections 3.4.3 and 3.4.4.

We tested 102 distinct UDFs in total (10 MNIST UDFs and 92 ImageNet UDFs). These selection UDFs yield various observed selectivities: around 10% (MNIST) and from 0.1% to 8.0% (ImageNet).

Results — We show improvement over competing methods in Figure 3.5. The figure shows the average improvement of VOODOO INDEXING for all the UDFs in each dataset. The upper figures show reduction in processed item counts, while lower figures show reduction in query time.

VOODOO INDEXING shows improvements for all datasets and almost all LIMIT values, yielding up to an 88.2% improvement over SCAN and a 79.0% improvement over ZOMBIE-G. As we would expect, our method's advantage is smaller as the LIMIT size approaches 100%; VOODOO INDEXING needs to explore low-payoff clusters and may eventually process almost the entire dataset. In the case of MNIST, this led to slightly *longer*-than-competition query times when using a LIMIT of 100% of the database due to the extra overhead of UCB calculation and top-down group selection. However, even in

this extreme case, VOODOO INDEXING did not yield performance that was meaningfully below other methods.

For all datasets, our method's advantage over ZOMBIE-G is lower than its advantage over SCAN, but is still substantial. Since both VOODOO INDEXING and ZOMBIE-G use the same sets of clusters and UDFs, these results show that our method's novel dendrogram approach is yielding real benefits in terms of reduced item selection and query runtimes.

Our method shows the best performance gains on MNIST. For a range of LIMITs from 10% to 90% of the data, VOODOO INDEXING spends much less time than the other two methods. Because the MNIST UDF has a selectivity of 10%, the best possible query processor — processing solely those data records where the UDF returns True — would yield a saving of 90%. Our method saves 80.3% - 88.2% of time compared to SCAN, close to that ideal. It saves 44.1% - 79.0% of time compared to ZOMBIE-G. Even when the user desires a small LIMIT size, for example 10% (around 700 items), our method can locate and select good items very quickly (only around 920 selections and 5.0 seconds). In contrast, ZOMBIE-G spends a long time early in execution trying to identify which index groups are valuable, showing most of its benefit only when the user needs larger amounts of data.

ImageNet data is much more complicated than the simple MNIST images, and has much finer-grained categories. But even in this challenging task, VOODOO INDEXING clearly wins on average. It also beats SCAN in 83 out of the 92 individual UDFs. In nine cases the system detects that VOODOO INDEXING is not successful, and so switches over to standard scan. Apart from extremely high-LIMIT cases, VOODOO INDEXING always shows an improvement. It achieves a 18.6% - 38.4% improvement over SCAN and 5.7%

Category @ LIMIT	Query Time (s)		
	No-Switch	Switch	Relative Change
Cuirass @ 30%	3219.9	2739.7	14.9%
Lipstick @ 30%	3965.0	3063.7	22.7%
Schooner @ 60%	3313.2	2598.2	21.6%
Screw @ 30%	3681.0	3216.0	12.6%
Spotlight @ 40%	4517.4	2946.1	34.8%
Corn @ 30%	4474.4	3849.2	14.0%

(a) The best LIMIT case

Category @ LIMIT	Query Time (s)		
	No-Switch	Switch	Relative Change
Cuirass @ 70%	7203.9	7739.1	-7.4%
Lipstick @ 70%	7434.0	8267.7	-11.2%
Schooner @ 80%	5357.9	6146.2	-14.7%
Screw @ 90%	9537.7	9942.6	-4.2%
Spotlight @ 80%	6697.0	8001.6	-19.5%
Corn @ 90%	10491.0	10617.1	-1.2%

(b) The worst LIMIT case

Table 3.3: Compare voodoo indexing with and without the dynamic switch mechanism on ImageNet.

- 31.0% improvement over ZOMBIE-G when LIMIT fraction is less than 80%.

3.6.3 Component Testing

In this section we show that all three extensions to VOODOO INDEXING are effective: dynamic index recovery, scan failover, and batch mode execution. In order to best illustrate their behavior when standard parts of VOODOO INDEXING fail, we ran our experiments on the most challenging dataset: applying the pre-trained ResNeXt-101 model to filter each one of twenty categories from ImageNet dataset.

3.6.3.1 Dynamic Index Recovery

Summary — Compared to naïve VOODOO INDEXING execution, using dynamic index recovery saves up to 34.8% in query time.

Overview — We tested how much dynamic index recovery can improve by trying to obtain better dendrograms midway through query execution. Whether using dynamic index recovery choice or not, we ran with the scan failover mechanism in place. We considered 200 scenarios: all 20 UDFs, at LIMIT levels ranging from 10% to 100%. For each UDF, we identified the LIMIT setting that yielded the biggest benefit from dynamic index recovery, and the worst benefit from dynamic index recovery (which could be negative, i.e., the method worsens the performance).

Results — For six categories, using dynamic index recovery made a meaningful difference to query runtimes. (For the remaining 14, using dynamic index recovery made no meaningful overall impact.) The best differences for these categories are shown in Table 3.3a. Compared to naïve VOODOO INDEXING, dynamic VOODOO INDEXING can save up to 34.8% query time. Note that this table shows the change derived from *an individual dynamic index switch*.

Although dynamic index recovery often helps overall, there may be certain settings where it actually yields worse results, as seen in Table 3.3b. (Remember that the greedy algorithm only examines query performance for a short period of time before deciding on a particular dendrogram for a long period.) But these bad scenarios are rare and small, and on balance dynamic index recovery is worthwhile. We believe this is because even if switching to a new index has an unfortunate impact, it can always switch back to a better dendrogram in the future.

3.6.3.2 Scan Failover

Summary — The scan failover mechanism can successfully detect cases in the ImageNet data where VOODOO INDEXING fails to yield a benefit, and then switch over to SCAN. This yields up to a 25.8% improvement in query time compared to dynamic VOODOO INDEXING without the scan failover, and adds only 6.0% extra runtime on average compared to SCAN.

Overview — As described in Section 3.4.4, there may be situations in which the user's predicate is simply unrelated to the index groups. In this case, the best strategy is simply to fail over to SCAN. We ran dynamic VOODOO INDEXING with and without the scan failover mechanism.

Results — Table 3.4 shows the results of these experiments for LIMIT setting that yields the best runtime improvement for scan failover. For five failed categories, we list the number of selected items used in the scan simulation process described in Section 3.4.4. We also show the runtimes for VOODOO INDEXING, in **no-failover** and **failover** modes. As expected, this mechanism shows obvious speedup on all of these five failure cases, yielding up to 25.8% improvement. The overhead in implementing this method — the processing of the scan simulation items — entails selecting fewer than 20,000 items, or just 6% of the total dataset size. Therefore, after switching to scan, VOODOO INDEXING's runtime of the cases in Table 3.4 is only 6.0% over scan's on average.

3.6.3.3 Batch Mode

Summary — Batch mode can yield 61.7% - 66.2% query time improvements.

Overview — All of the previous experiments are run in *single mode*; that is, the

Category @ LIMIT	Simulation size	Query Time (s)		
		No-Failover	Failover	Relative change
American lobster @ 50%	14898	6104.8	5835.3	4.4%
Dugong @ 40%	16390	5618.0	4624.5	17.7%
Greater Swiss Mountain dog @ 30%	15861	3849.0	2857.3	25.8%
Honeycomb @ 30%	18914	4585.6	3839.5	16.3%
Pop bottle @ 40%	19671	5596.6	4722.5	15.6%

Table 3.4: Comparing dynamic voodoo indexing with and without scan failover on five failure cases from ImageNet

method repeatedly (1) chooses a single data item to process, then (2) predicts which data item to choose next. However, for performance reasons we may want to load multiple data items into the GPU for batch processing, so as to fully utilize GPU RAM and data-parallel processing.

In this experiment, we modified the query time algorithm to choose 40 items at a time for processing. It loads all chosen items into GPU memory and applies the selection predicate to all 40 items. It then applies 40 updates to the bandit model.

Results — Figure 3.6 shows the averaged performance results, comparing dynamic VOODOO INDEXING with scan failover in single mode and in batch mode. In the left figure, we can see that the two approaches process roughly the same number of items. It is not surprising that single mode in most cases processes fewer items than batch mode; the bandit model in single mode should be more accurate.

It is quite surprising that in some cases single mode may process *more* items than batch mode. We do not yet understand why this happens; our current hypothesis is that processing batches of items may have an impact similar to that of changing the α exploration/exploitation parameter, and that this has a positive impact in some situations.

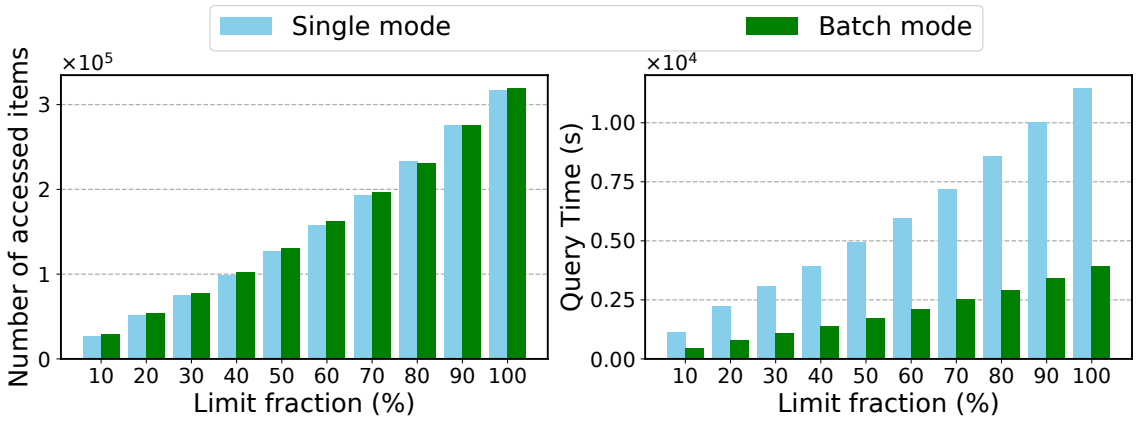


Figure 3.6: Batch vs single mode for voodoo indexing on the ImageNet dataset.

But the right hand figure shows that the runtime advantages of batch mode are definitely worth any small losses encountered via choosing imperfect data items: batch mode can save 61.7% - 66.2% of query time compared to single mode for any LIMIT value.

3.6.4 Varying Deployment Scenarios

In this section, we generate synthetic UDFs, index structures, and datasets with different selectivities to show VOODOO INDEXING's performance in the context of a wide range of deployment scenarios. We started with the same real-world datasets. Unless otherwise stated in the text below, we filtered them for the same labels by using the same UDFs as in Section 3.6.2, and averaged the results. In these experiments we set the LIMIT value to 40% of the total number of satisfying data items, a moderate and representative LIMIT setting. We see that VOODOO INDEXING is effective in most, but not all, plausible scenarios.

3.6.4.1 UDF Execution Time

Summary — VOODOO INDEXING is effective for all UDF runtimes we would expect to see in common processing settings.

Overview — In this section, we tested the lowest bound of UDF times that still make VOODOO INDEXING worthwhile compared with SCAN and ZOMBIE-G. For very short runtimes, the overhead of VOODOO INDEXING might not be worthwhile. We varied the UDF time from 0 to 20ms for each dataset.

Results — Figure 3.7 shows the speedup factor of VOODOO INDEXING over competing methods, with UDF runtime on the x-axis. The black dotted line is the boundary when speedup is 1. From this figure, we can see that our method's expected speedup is always greater than 1, except when the UDF runtime is very fast (under 1ms). At this stage, decreasing the number of predicate UDF invocations is not worth the extra overhead from the selection mechanism. As UDF runtime increases, VOODOO INDEXING shows it is worth its extra overhead. Eventually, the method's speedup reaches a plateau where it is close to the observed ratio of UDF invocations.

3.6.4.2 Index Structure Quality

Summary — Query performance varies with cluster quality; in some datasets, observed performance is close to what the best possible clustering might enable.

Overview — In this experiment, we varied the index group quality to see its impact on VOODOO INDEXING and competing methods. We created synthetic indexes in the following manner: (1) we set the group size to be the same as the clustered structure, (2) created a *random* index structure by placing each data item into a single randomly-chosen

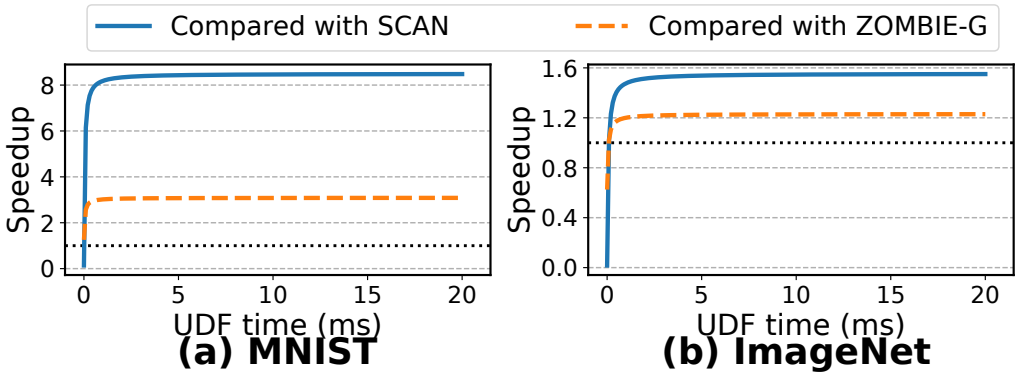


Figure 3.7: Speedup over Scan and Zombie-G, varying the UDF time.

cluster, (3) constructed a *best* index structure assigning data items to clusters according to their true labels.

Results — Perhaps unsurprisingly, Figure 3.8 shows that both VOODOO INDEXING and ZOMBIE-G are more effective when the index structure quality gets better. When the index structure is random, no method can do well, though at least the overhead of our method is minimal compared to SCAN. It is interesting to compare *best* performance to what can actually be obtained with the existing clustering. In MNIST, the observed practical performance comes close to what is enabled by the best possible clustering. For ImageNet, the large gap between observed performance and the *best* suggests that our practical clusterings of ImageNet are worthwhile but far below what might be possible. This analysis gives reason to hope that with better cluster features, VOODOO INDEXING might be able to obtain substantially better performance on ImageNet-linked queries.

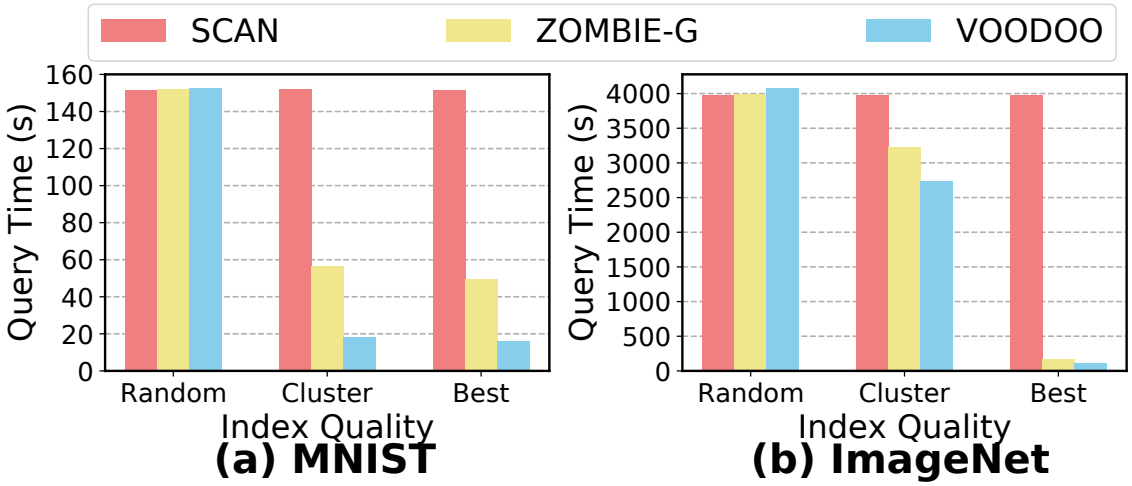


Figure 3.8: Analyze Voodoo's performance under different index structure quality

3.6.4.3 Selectivity

Summary — VOODOO INDEXING is effective under a very wide range of moderate UDF selectivities. It shows no worse performance than competing methods when UDF selectivities are extreme.

Overview — In this experiment, in order to test the range of UDF selectivities for which VOODOO INDEXING is effective, we randomly selected ten binary ResNeXt-based UDFs in ten categories, from among the eighty trained models introduced in Section 3.6.1. We constructed new datasets from ImageNet to yield an effective selectivity of around 0.5%, 10%, 30% and 90% for these UDFs. We averaged the query time improvements over competing methods for each selectivity level.

Results — Figure 3.9 shows that for all tested selectivities, VOODOO INDEXING is better than standard SCAN and ZOMBIE-G. When the selectivity is in the middle range (e.g. 10% and 30% as shown in the figure), our method yields substantial benefits. These

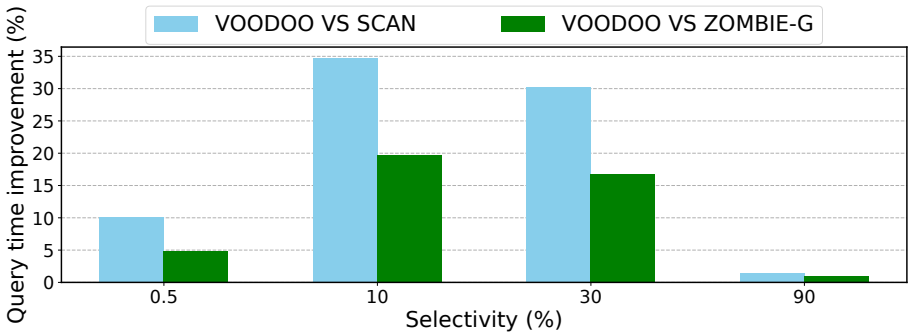


Figure 3.9: Query time improvement over competing methods on ImageNet, varying the selectivity.

results are to be expected. When the selectivity is extremely large, SCAN is an extremely effective algorithm, and when selectivity is extremely low, it can be challenging to find satisfying data records even in a high-quality index.

3.6.5 SparkSQL Experiments

Summary — Our prototype system can work for real SQL query processing and can achieve up to an 86.6% improvement over SparkSQL.

Overview — We tested our SparkSQL-specific system on 70000 images in MNIST dataset by running a SQL query with a synthetic UDF implemented as the predicate to filter digit 0 images. The executed query is:

```
1  SELECT * FROM MNIST
2  WHERE label(image) = 0 LIMIT 700
```

with varying LIMIT value. In order to show our system's performance under different levels of UDF complexity, we made the UDF *label()* returns ground truth and set the UDF time to be 5ms and 10ms. Similar to previous experiments, the dataset is clustered into

LIMIT	# samples		T(s)(UDF=5ms)		T(s)(UDF=10ms)	
	Voodoo	Spark	Voodoo	Spark	Voodoo	Spark
10%	**860**	7248	**7.40**	36.43	**11.70**	72.67
20%	**1585**	14313	**17.02**	72.15	**24.95**	143.72
30%	**2350**	21371	**22.92**	107.53	**34.67**	214.39
40%	**3051**	28589	**23.78**	143.74	**39.04**	286.69
50%	**3779**	35479	**31.16**	178.17	**50.06**	355.57
60%	**4497**	42320	**35.65**	212.46	**58.13**	424.06
70%	**5227**	49306	**40.12**	247.41	**66.25**	493.94
80%	**6307**	56124	**48.01**	281.63	**79.54**	562.25
90%	**8394**	63154	**82.69**	316.71	**124.66**	632.48
100%	**67048**	69984	660.98	**351.06**	996.22	**700.98**

Table 3.5: Results of our system and SparkSQL on MNIST

1000 groups and arranged in a dendrogram, and the balancing parameter α is set to be 1. We compared our system against a standard SparkSQL system that we extended with a LIMIT operator that stops scanning when the user query is satisfied.

Results — Table 3.5 compares our prototype system and SparkSQL in the terms of the number of items selected and the query time, with different kinds of UDFs. From this table, we can see that for different LIMIT levels, our system executes many fewer selections than standard SparkSQL, except the improvement when the LIMIT fraction is 100%. In this case, VOODOO INDEXING encounters the inevitable problem of low-payoff cluster exploration. No matter if UDF execution time is 5ms or 10ms, our system is much faster than SparkSQL when LIMIT is less than 100%, yielding up to 86.6% improvement in query time. The saved UDF time can make up for the extra overhead in our system around item location and selection. And when UDF execution time is large, our system's strength is more obvious.

3.6.6 Discussion

While our system can work for a wide range of opaque filter queries, it still has several limitations. Our mechanism may not be helpful when LIMIT values are extremely small, because of the time needed to identify high-value clusters. When the LIMIT value is extremely large, the system may also be slow, processing even low-payoff clusters. Fortunately, we believe that the broad range of middle-size LIMIT values will be popular and important, representing a good fit to analytical and ML style workloads.

The system may also be ineffective when the UDF returns TRUE extremely frequently or infrequently, because there is almost no difference between clusters' payoff, making cluster selection useless work and degrade to traditional scan. However, opaque filter queries with moderate selectivity are again likely to be a widespread workload: usually users would choose a related massive database consisted of items in more than one classes.

Overall, except for the above unusual situations, our mechanism is beneficial for common opaque filter queries on popular databases.

In addition, although our system is designed for video queries, it is also applicable for opaque filter queries on other unstructured data, such as text. For example, we can use this mechanism to filter sentences with positive sentiments.

3.7 Conclusion

As opaque filter queries in machine learning workloads become ubiquitous, an efficient filter strategy is required. However, due to its opaque semantic and unstable characteristic, real-world systems just execute these queries by simple scan. In summary, we present

a novel two-phase indexing mechanism for opaque filter query optimization, which can select data that satisfies UDF predicate quickly, yielding to query time reduction. In addition, we build standalone and SparkSQL-integrated systems, and verify that both of them can achieve high performance on real-world datasets.

CHAPTER IV

Controlled Intentional Degradation in Analytical Video Systems

4.1 Introduction

Society is experiencing a vast increase in the availability of video data. This data can be used for a range of public good applications, such as traffic monitoring and gathering commerce data. In these applications, administrators attach importance to analytical accuracy, but may also have competing goals. One goal is to meet system requirements. For example, wireless sensor networks, widely applied for building control, environmental monitoring, etc., suffer from low bandwidth and low power constraints [33, 130]. Another goal is to preserve private information (e.g., facial imagery) captured by video. This information can raise public concern due to potential leakage during shipment of video off-camera or execution of malicious queries. Finally, video surveillance is supposed to obey legal regulations [145]. For instance, according to the EU General Data Protection Regulation [180], face blurring is required when any closed-circuit television (CCTV)

footage is shared with a third party.

Generic intentional degradation methods are helpful for these analysis requirements [36, 142, 10, 197, 180, 56, 11, 48]. For example, frame rate reduction can be applied when the storage budget is limited. Frame resolution reduction can ensure legal compliance and is also useful for informal privacy protection. Although video degradation is extremely valuable, it usually does harm to analytical result accuracy, so it has to be done in a careful and controlled way. Unfortunately, no current system reveals how degradation affects analytical accuracy.

In response, we introduce a system for enabling **controlled intentional degradation**. The system has a few basic components:

- A **set of configurable networked cameras** that can collect, modify, and transmit images to a central system for query processing.

- A set of **destructive interventions** available in each camera: decreased resolution, decreased sampling rates, selective image removal, etc. These interventions likely solve system, privacy and legal compliance problems but likely decrease analytical query accuracy.

- A **video query processor** that receives a set of images from the cameras and implements an analytical query. It will be common for this query to include a UDF that embodies a trained neural network.

- A **public administrator** who determines the appropriate degradation/accuracy tradeoff for each query in a workload. This administrator could be an actual individual holding a public office, or a public committee, etc.

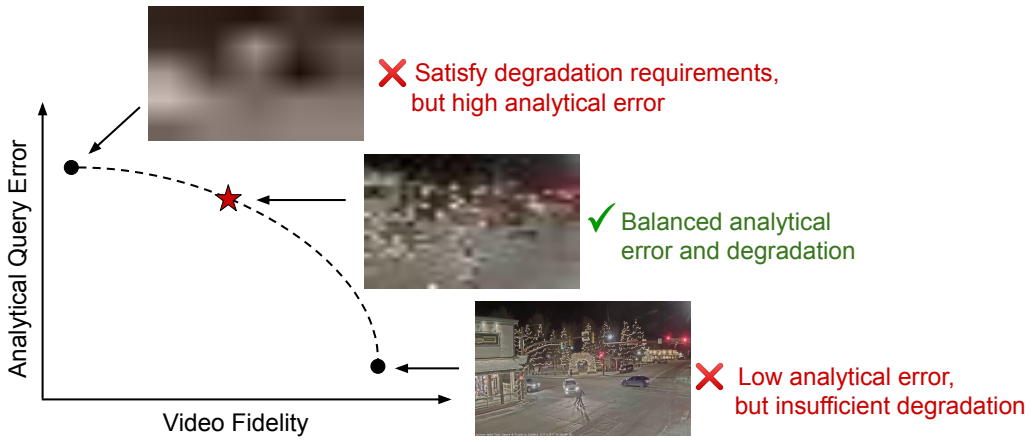


Figure 4.1: The public administrator must make a query-specific tradeoff that balances degradation requirements with the benefits of accurate analytical queries. Our system does not choose a tradeoff. Rather, it makes the tradeoff curve visible to the administrator.

EXAMPLE 1. *Harry is the public administrator for a city that collects surveillance videos of a road. The city wants to compute the average number of cars per frame on weekends so as to extrapolate the average cars per hour in order to schedule construction work. The city wants to maximize individuals' privacy, especially faces, and minimize the energy consumption during video transmission from cameras to the central system, but the maintenance department needs a frame-averaged car count that is within 10% of the correct answer. Harry configures the cameras to lower the frame resolution. However, the extremely low resolution has led to a query result that is badly wrong. Without knowing how the frame resolution affects the accuracy of the query, Harry cannot implement the city's preferences.*

System Goals — A well-informed tradeoff between destructive interventions and aggregate query accuracy is difficult to make, since interventions can interact in unexpected ways with the query. For example, a slightly-reduced frame sampling rate may not impact

a query that counts pedestrians since pedestrians move relatively slowly. But once the sampling rate falls under a particular threshold, the query may become very inaccurate. Therefore, even when the video system operator's degradation goals stay stable, the optimal tradeoff point can change with changes in the query, the destructive interventions, or the video contents. In an ideal world, the video system operator could examine a query-specific *tradeoff curve* (as in Figure 4.1) to determine an appropriate set of interventions.

EXAMPLE 2. *Harry submits the weekend car counting query to the system and receives a customized* degradation accuracy tradeoff curve. *By examining the curve, he finds that 128×128 is the lowest resolution that would not cause more than 10% analytical error. The cameras now collect and transmit only this low-resolution information, greatly improving privacy and saving energy while still giving the city maintenance department what it needs.*

Unfortunately, it is not clear how to generate this tradeoff curve. A simple approach would be: run the query on a representative portion of video, run it again on a degraded version of the video, and then compare the resulting query outputs. However, this naïve method presents serious problems:

- Accessing the original video and lightly-degraded video means we cannot conserve systems resources and preserve private data for the examined portion of video.
- It is computationally expensive, because it may need to be performed on many different degradation "knob settings".

The above problems may be acceptable if this examined portion is small and limited, but as mentioned above, we potentially need to recompute tradeoff curves for every new

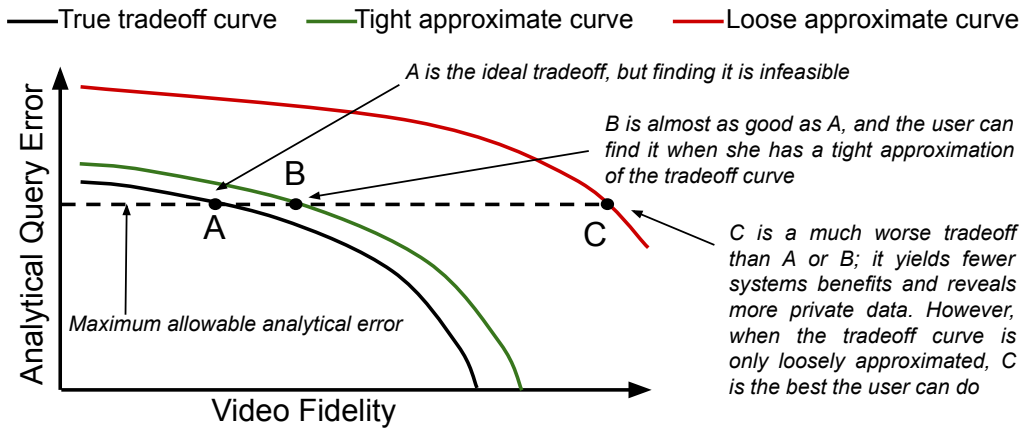


Figure 4.2: Conceptual diagram of approximate curves with a tight upper bound and a loose upper bound.

query, model, or video set. This naïve method may have to be applied almost continuously, violating the goals that motivated intentional degradation in the first place. We sidestep all of these problems by computing tradeoff curves without access to the underlying video. Moreover, we show that it can be done in a computationally efficient manner.

Technical Challenge — The main challenge of producing valuable tradeoff curves is to estimate the accuracy of the approximate query answer when video data is modified by any set of destructive interventions. The interventions transform video in different ways. For example, the *reduced frame sampling intervention* samples frames randomly; while the *image removal intervention* samples frames that do not contain restricted objects so that video features are modified non-randomly. The query analytical accuracy should be estimated under both random and non-random interventions.

This estimation problem is difficult because we can only get access to degraded samples instead of the unmodified video. A common solution is to compute the upper bound of the analytical error. Figure 4.2 shows a conceptual diagram of the true tradeoff curve

and approximate tradeoff curves, one with a tight and one with a loose upper bound. Given an analytical error threshold, if the true tradeoff curve were known, an administrator could choose the tradeoff at point A. A tight approximate curve lets the administrator choose a level of degradation at point B; the video here is less degraded than at point A, but the loss in degradation is not too bad. However, with a loose approximation curve, the administrator has no choice but to accept the worst tradeoff at C. As a consequence, we can see the upper bound needs to be tight. Online aggregation [68], stopping algorithms such as EBGS [127] and holistic aggregation approximation methods [108, 117] can provide error upper bounds for a variety of aggregate queries. However, these methods cannot compute sufficiently tight outputs to enable good degradation decisions, especially when video is substantially degraded. Moreover, they are not able to deal with non-random interventions.

Our Approach — We propose new algorithms to provide tight upper bounds of analytical error, allowing us to create better degradation/accuracy tradeoff curves for aggregate queries with `AVG`, `SUM`, `COUNT`, `MAX` and `MIN` functions. These queries' results are computed at a frame level, then aggregated; such queries have been introduced and investigated in previous work [84, 100]. Deduplicated aggregate query types are beyond the scope of this paper. The novelty of our work for each type of destructive intervention is summarized in Table 4.1.

When the destructive interventions are **random**, for aggregate queries with `AVG`, `SUM` or `COUNT`, we adapt the analytical error estimation method from the empirical Bernstein stopping algorithm [127], and further improve it by relaxing the confidence interval construction requirement and applying the Hoeffding–Serfling inequality [16]. For aggregate queries with `MAX` or `MIN`, we leverage the normal approximation for hypergeometric distri-

Video Scenario	Technical Problem	Our Novelty
Estimate analytical accuracy of video aggregate queries under random destructive interventions, e.g., reduced frame sampling. (Section 4.2.1)	Provide a tight upper bound of the error of the aggregate result estimation under a certain confidence level when the distribution of models' outputs is unchanged. (Section 4.2.4)	AVG, SUM, COUNT: Improve the error bound estimation method adapted from the empirical Bernstein stopping algorithm and apply the Hoeffding–Serfling inequality. (Section 4.3.2)
		MAX, MIN: Leverage the normal approximation for hypergeometric distribution to estimate the error bound of extreme quantiles. (Section 4.3.2)
Estimate analytical accuracy of video aggregate queries under non-random interventions, e.g., reduced frame resolution and image removal. (Section 4.2.1)	Provide a tight upper bound of the error of the aggregate result estimation under a certain confidence level when the distribution of models' outputs may change. (Section 4.2.4)	Profile repair: Use the randomly sampled correction set to correct possibly wrong error bounds and minimize the correction set size according to its own analytical accuracy, or create tradeoff curves from a similar but less sensitive video. (Section 4.3.2)

Table 4.1: The video scenarios, technical problems and novelty in our model.

bution in order to approximate the error of extreme quantiles.

When the destructive interventions are **non-random**, we propose a *profile repair* strategy. We introduce a *correction set* of video that is only modified by random interventions with the aim of correcting our method's analytical accuracy estimation. We minimize the size of this correction set as much as possible. Administrators may construct correction sets by applying random interventions to the query-specified video. When it is not possible to use only random interventions on the query video (perhaps when the video is especially sensitive), it is still possible to obtain a good approximation: administrators can choose

to compute from a separate video set that is similar to the query video, yielding a similar tradeoff curve, and then use this curve to guide non-random interventions applied to the intended query video. Finally, note that the correction set can also improve the accuracy of tradeoff curves for random interventions in some cases.

Contributions — Our contributions are as follows:

- We propose a novel *video degradation-accuracy profiling* model that enables governments to implement well-informed tradeoffs for system, privacy and legal compliance reasons. (Section 4.2)

- We design novel algorithms for random and non-random destructive interventions to compute tight error bounds of query result estimations for tradeoff curve profiling. Our method can obtain a 155% tighter error bound than the previous state-of-the-art method. (Section 4.3)

- We embodied these ideas in a prototype software system, SMOKESCREEN, and evaluated it on a range of video datasets and aggregate query types. We show that SMOKESCREEN enables tradeoffs that are 88% more accurate than a method based on previously-known approaches. (Sections 4.4 and 4.5)

4.2 Problem Formulation

We introduce the types of video degradation in Section 4.2.1, the importance of degradation accuracy tradeoff curves in Section 4.2.2, frequently-used vocabulary in our model in Section 4.2.3, and the technical problems' formal formulation in Section 4.2.4.

4.2.1 Video Degradation

There are often system, privacy and legal compliance requirements in addition to the pure analytical accuracy requirement, so administrators have to balance these competing goals. Intentionally degrading video is a common operation in analytical settings. Here are three ways to do it:

Intervention example 1: Reduced frame sampling — This method reduces the ratio of the randomly sampled frames against the total query-specified frames. With this intervention, time-related privacy (e.g., daily life tracks) will not be revealed [36], and video file size can be reduced to meet system requirements such as a low bandwidth constraint [142, 10] and energy limitations [197].

Intervention example 2: Reduced frame resolution — This method reduces the resolution of processed frames. With this intervention, objects like faces that can be recognized from high-resolution images will not be revealed so as to obey legal regulations [180]; the burden on system resources can also be mitigated [56, 11].

Intervention example 3: Image removal — This method entirely deletes frames that contain restricted objects so as to ensure legal compliance and preserve privacy [48]. Sensitive objects include people, faces, license plates, etc. Any combination of them may be considered to be restricted.

Besides these three examples, there are also other degradation methods, such as noise addition [191], video compression techniques [63], etc. All of these methods can be divided into two categories. **Random** interventions modify video features such that the distribution of models' outputs is unchanged (e.g., reduced frame sampling). **Non-random**

interventions modify underlying videos such that the distribution of models' outputs may change (e.g., reduced frame resolution and image removal).

A single intervention type may not meet all the requirements, such as different legal regulations, and may affect analytical accuracy much more than other interventions. For example, previous experiments show that low video resolution can significantly affect the accuracy of some classification models [95]. As a consequence, we allow the administrator to choose a combination of the above three typical intervention examples, covering both random and non-random intervention types. Administrators can tune these degradation knobs (making sample fraction and resolution up or down, and choosing preferred restricted objects) in order to trade analytical accuracy against degradation goals.

4.2.2 Degradation Accuracy Trade-Off Curves

Two important features about video make the tradeoff problem difficult. First, administrators, perhaps driven by their local government, usually have different preferences about query answer quality and the best degradation level. As a result, it is not feasible to simply fix the intervention settings for all administrators. Second, the shape of the degradation accuracy tradeoff curve changes depending on the query (e.g., calculating the average or the maximum number of cars) and the video content (e.g., collected from the surveillance camera at a downtown intersection or at a narrow road). Figure 4.3 shows two real degradation accuracy tradeoff curves of queries that compute the average number of cars per frame on night-street video [84] and UA-DETRAC video [184]. YOLOv4 [20] is used in these queries to detect cars. The x-axis describes frame resolution, while the y-axis is the relative error of the estimated query result. These two curves are quite different from each

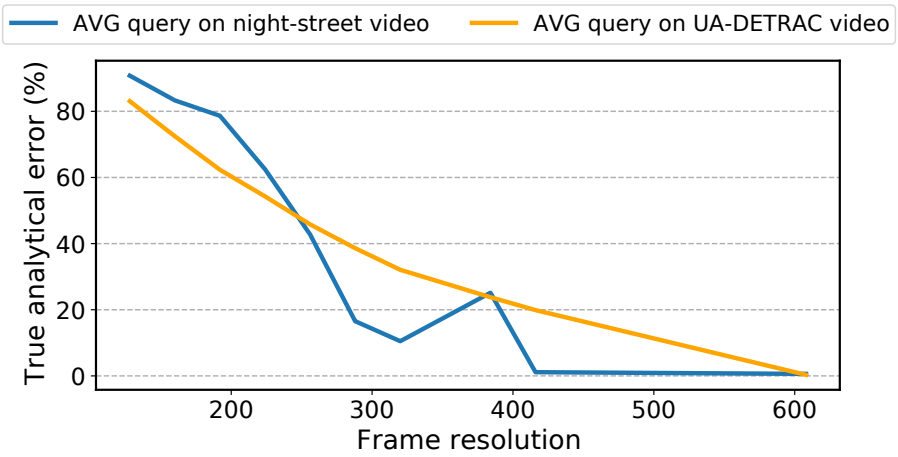


Figure 4.3: Real degradation accuracy tradeoff curves for the AVG query on two different video datasets.

other, illustrating how they are video-dependent.

Therefore, a system that supports administrators in making this crucial degradation/accuracy trade-off must provide video- and query-specific curves.

4.2.3 Usage Model

In our video degradation-accuracy profiling model, frequently used vocabulary is summarized as follows:

- **Original video**: The raw unaltered video collected by the set of networked cameras. This video has not yet been processed by the destructive interventions. It is never processed directly by the video query processor.

- **Degraded video**: Applying a destructive intervention to the original video will yield a set of degraded video. It can then be analyzed by the video query processor.

- **Profile**: A profile describes a tradeoff between a destructive intervention and analytical accuracy for each unique combination of video corpus, query, and intervention. The profile consists of a set of (degradation, error) pairs; missing values should simply be interpolated by the administrator. The profile shows the error caused by video degradation when compared with the query result derived from non-degraded video, so the error values are computed without regard to the absolute accuracy of the video analysis model.

- **Profile generation**: Our system produces a unique profile for a given video corpus, query, and intervention.

- **Choosing a tradeoff**: Administrators use a profile to select a desired level of destructive intervention. We expect that queries contain video analysis models of high accuracy, or at least that administrators know the approximate accuracy of models. Administrators can adjust the analytical accuracy threshold in the selection process by considering models' inherent accuracy. This selected degradation setting is then applied for query result estimation.

- **Public preferences**: Preferences that guide the administrator when choosing a tradeoff. Forms of preference include: the minimum allowable analytical error, the maximum allowable frame resolution, and so on.

Consider Harry using our model:

EXAMPLE 3. *Harry activates our profiling model for his query. During the* ***profile generation*** *stage, the system produces the* ***profiles*** *by degrading a representative portion of* ***original video*** *under multiple sets of interventions and sending the* ***degraded video*** *to the*

Parameter	Description	Example
D	Video data	Surveillance video
F_{model}	Video analytics model	Car detector
F_A	Aggregate function	Average
f	Reduced sampling	0.1
p	Reduced resolution	128×128
c	Restricted object	Person
$1-\delta$	Confidence level	95%
$X_1, ..., X_N$	Model outputs on original frames	# cars in 1000 frames
$x_1, ..., x_n$	Model outputs on degraded frames	# cars in 100 frames (128×128) with no people contained
$v_1, ..., v_m$	Model outputs on correction set	# cars in randomly sampled 200 frames
Y_{true}	True query answer	Average value of $X_1, ..., X_N$
Y_{approx}	Approximate answer	Average value of $x_1, ..., x_n$
err_b	Upper bound of approximation error	Upper bound of the relative error $\lvert\frac{Y_{approx}-Y_{true}}{Y_{true}}\rvert$

Table 4.2: Frequently used notation in SMOKESCREEN

*query processor for analytical error bound estimation, then returns the **profiles** to Harry. During the **choosing a tradeoff** stage, Harry determines a proper set of interventions according to the **public preferences**, so he tunes the knobs and runs the car-counting query on the appropriately degraded video to obtain an approximate query result that is within 10% of the correct result.*

4.2.4 Technical Formulation

With the above design of the degradation-accuracy profiling model, we still face several technical problems. In the *choosing a tradeoff* stage, algorithms are needed to estimate the query result under destructive interventions. In the *profile generation* stage, algorithms are needed to estimate the analytical accuracy under a broad range of degradation settings; this stage should operate on video that is degraded as much as possible while still yielding

a valid profile. These problems can be stated formally, and all of the notation is listed in Table 4.2.

The video analytical query is characterized by a 3-tuple of parameters (D, F_{model}, F_A). The video data D is queried to collect useful information. Two functions, F_{model} and F_A, represent the video analysis model (e.g., car detector) and the aggregate function (e.g., `AVG`) in the query. This analysis model's behavior is our definition of the ground truth. The value N is the number of frames that should be sent to F_{model} and F_A in naïve execution. In addition, the 3-tuple of parameters (f, p, c) represent the destructive interventions, which are reduced frame sampling, reduced frame resolution, and restricted objects respectively. The analytical query answer should be executed under these interventions, that is, only n ($n = N \times f$) frames with resolution p (e.g., 128×128), which do not contain objects c (e.g., person), may be processed by F_{model} and F_A to obtain the approximate query answer Y_{approx}. The value err_b, computed to reflect the analytical accuracy, denotes the upper bound of the relative error of the approximate query result compared with the true result with probability at least $1 - \delta$.

PROBLEM 1: *Given video analytical query (D, F_{model}, F_A), compute the approximate query answer Y_{approx} and a tight upper bound err_b of the approximation error under destructive interventions (f, p, c).*

PROBLEM 2: *In the profile generation stage, compute the profiles while maximizing the interventions.*

4.3 Algorithms

Now we introduce our novel algorithms to solve the above problems. In Section 4.3.1, we describe the administration procedure in the stages of profile generation and choosing a tradeoff. In Section 4.3.2, we propose our query answer and error bound estimation algorithms for frequently used aggregate query types. In Section 4.3.3, we further discuss the details of profile generation in our model.

4.3.1 Administration Procedure

In the *profile generation* stage, provided with a query with an analysis model F_{model} and an aggregate function F_A, a tight upper bound of analytical error is computed when the original video D is degraded by every set of intervention candidates. (Intervention candidate selection will be discussed in Section 4.3.3.) These error bounds can form a degradation hypercube with cube slices as multiple two-dimensional arrays that are returned to the administrators in order to choose an appropriate set of interventions. Initially, administrators are only shown three cube slices — obtained by fixing each unseen dimension to the loosest intervention value — visualized as 2D plots. They choose intervention candidates by considering both public preferences (e.g., images that contain people should be removed) and the interventions' effect on analytical accuracy shown in the curves (e.g., resolution 128×128 makes query results too inaccurate), and then adjust the fixed dimensions for more plots, and fine-tune these knobs according to bounded error values. At last, the query result is estimated by running the query on the video D or upcoming videos processed by the determined degradation operations. The algorithms of estimating analytical results and error bounds are described in the following sections.

4.3.2 Query Answer and Error Bound Estimation

We describe our estimation algorithms for frequently used aggregate functions. We first address the case of reduced frame sampling and then we introduce the profile repair strategy for non-random interventions.

4.3.2.1 AVG Function

The aggregate function `AVG()` is applied to calculate the frame-level average value of a user-defined vision model's outputs on video frames. In EXAMPLE 1, the public administrator, Harry, applies this function to collect the average number of cars per frame in order to learn how busy the road is. Let $X_1, X_2, ..., X_N$ denote the outputs of N frames with mean μ and range R. Due to the reduced frame sampling intervention f, only n frames are randomly sampled, yielding outputs $x_1, x_2, ..., x_n$. The relative error of the approximate query result Y_{approx} compared with the true query result μ, $|\frac{Y_{approx}-\mu}{\mu}|$, is used as the analytical accuracy metric, so we aim to compute the upper bound of this relative error.

Many research efforts have focused on this computation. When the sample size is relatively large, sample mean approximately obeys the normal distribution according to the central limit theorem, so the upper bound of absolute error between sample mean and true mean can be derived [68], and then the upper bound of relative error can be obtained by dividing the lower bound of the query result. However, it is highly probable that the administrator chooses the sample fraction to be a small value. In other words, the central limit theorem will become useless exactly in the scenarios where our system aims to be the most useful. Online aggregation [68] also provides another more conservative bound

from Hoeffding's inequality [72]. Besides these classic approaches, a tighter upper bound can be derived from the Hoeffding-Serfling inequality [16] proposed recently, which assumes sampling without replacement instead of $i.i.d.$ sampling. Moreover, early stopping algorithms — determining a stopping point when the error is within some threshold — can also be adapted for the error bound estimation. The empirical Bernstein stopping algorithm [127] provides a new query result estimation instead of sample mean, yielding a tighter error bound. We further improve this method by relaxing the confidence interval construction requirement and applying the tight Hoeffding-Serfling inequality [16], which is more suitable for a small sample size than the empirical Bernstein bound [14] in the original version. This estimation mechanism is shown in Algorithm 4.

Algorithm 4: AVG()

Input: Aggregate query (D, F_{model}, AVG), Intervention f, δ

`// Sample model outputs`

$x_1, x_2, ..., x_n$ = F_{model}(Sample(D, f));

`// Calculate Hoeffding-Serfling bound` I

Compute sample range R and sample mean $\bar{x}_n$;

$\rho_n = \min\{(1 - \frac{n-1}{N}), (1 - \frac{n}{N})(1 + \frac{1}{n})\}$;

$I = R\sqrt{\frac{\rho_n \log(2/\delta)}{2n}}$;

`// Compute approximate result and error bound`

$UB = |\bar{x}_n| + I$;

$LB = \max(0, |\bar{x}_n| - I)$;

$Y_{approx} = sgn(\bar{x}_n) \cdot \frac{2UB \cdot LB}{UB + LB}$;

$err_b = \frac{UB - LB}{UB + LB}$;

Output: Y_{approx}, err_b

Hoeffding-Serfling inequality states that with probability at least $1 - \delta$, $\bar{x}_n - \mu \leq R\sqrt{\frac{\rho_n \log(1/\delta)}{2n}}$, where $\bar{x}_n$ is the sample mean: $\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\{x_i\}$ is sampled without replacement, and $\rho_n = \min\{(1 - \frac{n-1}{N}), (1 - \frac{n}{N})(1 + \frac{1}{n})\}$. Similarly, with this confidence

level, $\bar{x}_n - \mu \geq -R\sqrt{\frac{\rho_n \log(1/\delta)}{2n}}$. Due to union bound, with probability at least $1-\delta$, $|\bar{x}_n - \mu| \leq R\sqrt{\frac{\rho_n \log(2/\delta)}{2n}}$. We denote this bound as I, so $(\bar{x}_n - I, \bar{x}_n + I)$ is a $1-\delta$ confidence interval for μ. In contrast to the empirical Bernstein stopping algorithm, we do not need to simultaneously construct the intervals for all $n \in \mathbb{N}^+$ but just for the sample size n under $1-\delta$ confidence level. As a result, this confidence interval can be smaller by our construction. Correspondingly, we set LB to $\max(0, |\bar{x}_n| - I)$ and UB to $|\bar{x}_n| + I$ rather than the definitions in the stopping algorithm.

Theorem 1. *The approximate query answer Y_{approx} and the error bound err_b with probability at least $1-\delta$ are as follows:*

$$Y_{approx} = sgn(\bar{x}_n) \cdot \frac{2UB \cdot LB}{UB + LB}, \tag{4.1}$$

$$err_b = \frac{UB - LB}{UB + LB}. \tag{4.2}$$

Proof. With probability at least $1-\delta$,

$$|Y_{approx}| = \frac{2UB \cdot LB}{UB + LB} = (1 + err_b)LB \leq (1 + err_b)|\mu|, \tag{4.3}$$

$$|Y_{approx}| = \frac{2UB \cdot LB}{UB + LB} = (1 - err_b)UB \geq (1 - err_b)|\mu|. \tag{4.4}$$

When $LB = 0$, it can be derived that $Y_{approx} = 0$ and $err_b = 1$, so err_b is the error bound. When $LB \neq 0$, the inequality $|\bar{x}_n| > I \geq |\bar{x}_n - \mu|$ holds true, so $sgn(Y_{approx}) = sgn(\bar{x}_n) = sgn(\mu)$, where $sgn()$ is the sign function. We can obtain the following inequality:

$$\left|\frac{Y_{approx} - \mu}{\mu}\right| = \frac{||Y_{approx}| - |\mu||}{|\mu|} \leq err_b \tag{4.5}$$

Therefore, the above theorem holds true. □

4.3.2.2 SUM Function

The aggregate function `SUM()` is applied to calculate the sum of the model's outputs in each frame. This function can be used to compute the sum of all cars seen in each frame in a time period. It captures both car number and car speed information, which is valuable for determining the road congestion level. The parameters and the error metric are the same as the above. In this case, $Y_{true} = N\mu$, so we compute the upper bound of the relative error $|\frac{Y_{approx}-N\mu}{N\mu}|$. We assume that the length of video is known before any processing. According to the above conclusion, we define $Y_{approx} = sgn(\bar{x}_n) \cdot \frac{2UB \cdot LB}{UB+LB} \cdot N$ and $err_b = \frac{UB-LB}{UB+LB}$ to make err_b the error bound with probability at least $1 - \delta$.

4.3.2.3 COUNT Function

The aggregate function `COUNT()` is applied to calculate the number of frames that satisfy the query predicate. This function can be used to compute the number of frames (i.e., the length of time) when there are varying levels of cars. It would be helpful to decide when congestion is low enough to close a single lane. Although this seems like a new problem, we can redefine it as the estimation problem for `SUM`. For each frame i, if the predicate model returns TRUE, we assign an associated value 1 to X_i; otherwise, 0 is assigned to X_i. Therefore, the count problem is transformed to calculating the sum of X_i, and the above conclusion can be directly applied here.

4.3.2.4 MAX/MIN Function

The aggregate function `MAX()` or `MIN()` is applied to calculate extreme values in the frame-level outputs. This function can be used to compute the maximum/minimum num-

ber of cars that exist in one frame in order to detect the most/least crowded moment. Unfortunately, it is hard to estimate and analyze extreme values just by sampling, because only the extreme value itself in the samples seems to be related to the true result. Therefore, we use rth-quantile to estimate the result of `MAX()` and `MIN()` (when r is close to 1 or 0). The goal is transformed into estimating the rth-quantile in the outputs, $X_1, X_2, ..., X_N$. There are n frames randomly sampled without replacement for processing, yielding $x_1, x_2, ..., x_n$. For quantiles, BlinkDB [6] uses the same relative error metric as other aggregate query types. However, this metric is substantially affected by the hidden distribution, especially for extreme quantiles. As a result, the ranks rather than the actual values are compared, that is, the relative error between the ranks of Y_{true} and Y_{approx} in the original array, $|\frac{rank(Y_{approx})-rank(Y_{true})}{rank(Y_{true})}|$, is used to reflect the accuracy. This metric is also compatible with the definition of ϵ-approximate quantile [116].

Previous works [108, 117] have designed sampling-based algorithms to estimate quantiles in wireless sensor networks and for business intelligence applications. The classic approach [117] proposed an estimation based on Stein's lemma. A recent work [108] made estimates based on the central limit theorem. However, there are two problems in these algorithms. First, the inequality bound is too loose during the derivation process. Second, they assume random sampling with replacement, which is less reasonable than our non-replacement assumption. Both of them lead to loose upper bounds. We make improvements based on recent work [108] (the novelty is summarized in Table 4.1). We propose the quantile approximation algorithm as follows, shown in Algorithm 5, and compare our algorithm with the better approach [117] between the above two previous works in Section 4.5.2.

Algorithm 5: MAX() or MIN()

Input: Aggregate query (D, F_{model}, F_A), Intervention f, Extreme percentage r, δ

`// Sample model outputs`

$x_1, x_2, ..., x_n$ = F_{model}(Sample(D, f));

`// Compute approximate result and error bound`

$sortList$ = Sort($x_1, x_2, ..., x_n$);

Y_{approx} = $sortList[n \cdot r]$;

$\hat{F}_{\hat{k}}$ = $sortList$.count(Y_{approx}) / n;

if F_A == MAX **then**

  $err_b = (\frac{\phi_{\frac{\delta}{2}}\sqrt{r(1-r)}\sqrt{\frac{N-n}{n(N-1)}}+\hat{F}_{\hat{k}}}{\hat{F}_{\hat{k}}} + 1) \cdot \frac{\hat{F}_{\hat{k}}}{r}$

end

else

  $err_b = (\frac{\phi_{\frac{\delta}{2}}\sqrt{(r+\hat{F}_{\hat{k}})[1-(r+\hat{F}_{\hat{k}})]}\sqrt{\frac{N-n}{n(N-1)}}+\hat{F}_{\hat{k}}}{\hat{F}_{\hat{k}}} + 1) \cdot \frac{\hat{F}_{\hat{k}}}{r}$

end

Output: Y_{approx}, err_b

Let $\{s_1, s_2, ...\}$ be the sorted distinct values in $X_1, X_2, ..., X_N$. Each s_i occurs N_i times in this array and n_i times in the sampled array, the frequency of which is $F_i = \frac{N_i}{N}$ and $\hat{F}_i = \frac{n_i}{n}$ respectively. According to the definition of rth-quantile, $Y_{true} = \min_i\{s_i : \sum_{j=1}^{i} F_j \geq r\}$. Let Y_{true} and Y_{approx} be the kth and $\hat{k}$th distinct value, i.e., $Y_{true} = s_k$ and $Y_{approx} = s_{\hat{k}}$.

Theorem 2. *The approximate quantile Y_{approx} and error bound err_b with probability at least $1 - \delta$ can be constructed as follows.*

$$Y_{approx} = \min_i\{s_i : \sum_{j=1}^{i} \hat{F}_j \geq r\}. \tag{4.6}$$

When the aggregate function is MAX, r is close to 1,

$$err_b = (\frac{\phi_{\frac{\delta}{2}}\sqrt{r(1-r)}\sqrt{\frac{N-n}{n(N-1)}} + F_k}{\min_{\hat{k}+1\leq i\leq k-1 \text{ or } k+1\leq i\leq \hat{k}-1} \hat{F}_i} + 1) \cdot \frac{\max_{\hat{k}+1\leq i\leq k \text{ or } k+1\leq i\leq \hat{k}} F_i}{r}. \tag{4.7}$$

And when the aggregate function is MIN, r is close to 0,

$$err_b = (\frac{\phi_{\frac{\delta}{2}}\sqrt{(r+F_k)[1-(r+F_k)]}\sqrt{\frac{N-n}{n(N-1)}}+F_k}{\min_{\hat{k}+1\le i\le k-1 \text{ or } k+1\le i\le \hat{k}-1}\hat{F}_i}+1)\cdot \frac{\max_{\hat{k}+1\le i\le k \text{ or } k+1\le i\le \hat{k}}F_i}{r}. \tag{4.8}$$

Proof sketch: The error metric satisfies the inequality:

$$error = \frac{|\sum_{i=1}^{k}F_i - \sum_{i=1}^{\hat{k}}F_i|}{\sum_{i=1}^{k}F_i} \le \frac{|k-\hat{k}|\max_{\hat{k}+1\le i\le k \text{ or } k+1\le i\le \hat{k}}F_i}{r}. \tag{4.9}$$

According to the definition, we have $\sum_{i=1}^{\hat{k}-1}\hat{F}_i < r \le \sum_{i=1}^{k}F_i$ and $\sum_{i=1}^{k-1}F_i < r \le \sum_{i=1}^{\hat{k}}\hat{F}_i$. So when $\hat{k} > k$, $\hat{k}-k < \frac{\sum_{i=1}^{k}F_i-\sum_{i=1}^{k}\hat{F}_i}{\min_{k+1\le i\le \hat{k}-1}\hat{F}_i}+1$, and when $k > \hat{k}$, $k-\hat{k} < \frac{\sum_{i=1}^{k}\hat{F}_i-\sum_{i=1}^{k}F_i+F_k}{\min_{\hat{k}+1\le i\le k-1}\hat{F}_i}+1$. Therefore,

$$error < (\frac{|\sum_{i=1}^{k}\hat{F}_i - \sum_{i=1}^{k}F_i| + F_k}{\min_{\hat{k}+1\le i\le k-1 \text{ or } k+1\le i\le \hat{k}-1}\hat{F}_i}+1)\cdot\frac{\max_{\hat{k}+1\le i\le k \text{ or } k+1\le i\le \hat{k}}F_i}{r}. \tag{4.10}$$

Because $\sum_{i=1}^{k}\hat{F}_i = \frac{\sum_{i=1}^{k}n_i}{n}$, and $\sum_{i=1}^{k}n_i$ obeys hypergeometric distribution, we can obtain that $\mathbb{E}[\sum_{i=1}^{k}\hat{F}_i] = \sum_{i=1}^{k}F_i$ and $Var[\sum_{i=1}^{k}\hat{F}_i] = \sum_{i=1}^{k}F_i(1-\sum_{i=1}^{k}F_i)\cdot\frac{N-n}{n(N-1)}$. It has been demonstrated that there is a normal approximation for the hypergeometric distribution when $N, n, \sum_{i=1}^{k}N_i, \sum_{i=1}^{k}n_i$ are large [131, 47], so there exists an asymptotic normal distribution: $\frac{\sum_{i=1}^{k}\hat{F}_i-\sum_{i=1}^{k}F_i}{\sqrt{Var[\sum_{i=1}^{k}\hat{F}_i]}} \sim \mathcal{N}(0,1)$. When r is close to 1, $Var[\sum_{i=1}^{k}\hat{F}_i] \le r(1-r)\cdot\frac{N-n}{n(N-1)}$, so

$$P\left(|\sum_{i=1}^{k}\hat{F}_i - \sum_{i=1}^{k}F_i| \ge \phi_{\frac{\delta}{2}}\sqrt{r(1-r)}\sqrt{\frac{N-n}{n(N-1)}}\right) \le \delta, \tag{4.11}$$

where $\phi_{\frac{\delta}{2}}$ is the Z-score. Therefore $P(error \ge err_b) \le \delta$ is satisfied. When r is close to 0, $Var[\sum_{i=1}^{k}\hat{F}_i] \le (r+F_k)[1-(r+F_k)]\cdot\frac{N-n}{n(N-1)}$. Similarly, the theorem holds true.

In the formula of err_b, F_i (for any $i \in \mathbb{N}^+$), k, and $\hat{k}$ are unknown. Ideally, $\hat{F}_i$ and F_i, k and $\hat{k}$ should be close, so we use $\hat{F}_{\hat{k}}$ to estimate F_k, $\min_{\hat{k}+1\le i\le k-1 \text{ or } k+1\le i\le \hat{k}-1}\hat{F}_i$, and

$\max_{\hat{k}+1 \leq i \leq k \text{ or } k+1 \leq i \leq \hat{k}} F_i$ above. It needs to be noted that a distribution approximation is utilized in the above proof. Although it holds true when sample size is large, the derived error bound is still valid experimentally in Section 4.5 even when sample size is vary small.

4.3.2.5 Managing Combinations of Random and Non-random Interventions through Profile Repair

We have provided the error bound for different aggregate functions for random interventions. However, the algorithms cannot be directly applied when there are non-random interventions because sampled outputs from videos degraded by non-random interventions can be systematically wrong in one direction. Under this circumstance, these sampled outputs are not enough for an accurate error bound. A correction set, $v_1, v_2, ..., v_m$, obtained from processing videos degraded by only random interventions, is required to repair the biased bound. Its construction is elaborated in Section 4.3.3. Once it is constructed, it can be used for correcting error bounds of any combination of interventions. Our algorithm is shown in Algorithm 6, and the proof sketches for the error bounds are presented below. These proofs leverage the error bound conclusion connected with random interventions (see Theorem 1 and 2). That is, the error bound of the correction set under a certain confidence level has been proved, and it is utilized in the inequality derivation below. Further, note there is no distributional assumption of the outputs from videos degraded by non-random interventions.

For the aggregate function `AVG()`, we assume that $v_1, v_2, ..., v_m$ are randomly sampled outputs without replacement. The approximate answer $Y_{approx}(\mathbf{v})$ to estimate μ and the error bound $err_b(\mathbf{v})$ obtained only from the correction set as in Equation 4.1 and 4.2 can

Algorithm 6: Managing a Combination of Random and Non-random Interventions

Input: Aggregate query (D, F_{model}, F_A), Destructive interventions (f, p, c), δ, r, m

`// Compute approximate result and error bound of the degraded video and the correction set`

Y_{approx}, err_b = resultErrorEst(D, F_{model}, F_A, f, p, c, δ, r);

$Y_{approx}(\mathbf{v})$, $err_b(\mathbf{v})$ = resultErrorEst(D, F_{model}, F_A, m/len(D), None, None, δ, r);

`// Correct the error bound of degraded video`

if F_A == AVG or SUM or COUNT **then**

  $err_b = \frac{(1+err_b(\mathbf{v}))|Y_{approx} - Y_{approx}(\mathbf{v})|}{|Y_{approx}(\mathbf{v})|} + err_b(\mathbf{v})$

end

if F_A == MAX or MIN **then**

  $\sum_{i=1}^{\hat{k}} \hat{F}_i$ = Rank of Y_{approx} in correction set / m;

  $\sum_{i=1}^{\hat{k}(\mathbf{v})} \hat{F}_i$ = Rank of $Y_{approx}(\mathbf{v})$ in correction set / m;

  $err_b = \frac{|\sum_{i=1}^{\hat{k}} \hat{F}_i - \sum_{i=1}^{\hat{k}(\mathbf{v})} \hat{F}_i|}{r} + err_b(\mathbf{v})$

end

Output: err_b

satisfy $\frac{|Y_{approx}(\mathbf{v})-\mu|}{|\mu|} \leq err_b(\mathbf{v})$, with probability at least $1-\delta$. So when non-random interventions exist, the error bound for the approximate result Y_{approx} can be derived as follows:

$$\begin{aligned}\frac{|Y_{approx}-\mu|}{|\mu|} &\leq \frac{|Y_{approx}-Y_{approx}(\mathbf{v})|+|Y_{approx}(\mathbf{v})-\mu|}{|\mu|} \\ &\leq \frac{(1+err_b(\mathbf{v}))|Y_{approx}-Y_{approx}(\mathbf{v})|}{|Y_{approx}(\mathbf{v})|}+err_b(\mathbf{v}).\end{aligned} \tag{4.12}$$

Since it is derived from the error bound of the correction set, this error bound also holds true with probability at least $1-\delta$. And for other functions, `SUM()` and `COUNT()`, because the error metric is the same, the corrected error bound can be derived similarly.

For the aggregate function `MIN()` or `MAX()`, the approximate rth-quantile $Y_{approx}(\mathbf{v})$ and the error bound $err_b(\mathbf{v})$ obtained only from the correction set as in Equation 4.6, 4.7, and 4.8 can satisfy $\frac{|\sum_{i=1}^{\hat{k}(\mathbf{v})}F_i-\sum_{i=1}^{k}F_i|}{\sum_{i=1}^{k}F_i} \leq err_b(\mathbf{v})$ with probability at least $1-\delta$, where $Y_{approx}(\mathbf{v})$ is the $\hat{k}(\mathbf{v})$th distinct value. So

$$\begin{aligned}\frac{|\sum_{i=1}^{\hat{k}}F_i-\sum_{i=1}^{k}F_i|}{\sum_{i=1}^{k}F_i} &\leq \frac{|\sum_{i=1}^{\hat{k}}F_i-\sum_{i=1}^{\hat{k}(\mathbf{v})}F_i|+|\sum_{i=1}^{\hat{k}(\mathbf{v})}F_i-\sum_{i=1}^{k}F_i|}{\sum_{i=1}^{k}F_i} \\ &\leq \frac{|\sum_{i=1}^{\hat{k}}F_i-\sum_{i=1}^{\hat{k}(\mathbf{v})}F_i|}{r}+err_b(\mathbf{v}),\end{aligned} \tag{4.13}$$

with probability at least $1-\delta$. In this formula, the true rank difference $|\sum_{i=1}^{\hat{k}}F_i-\sum_{i=1}^{\hat{k}(\mathbf{v})}F_i|$ is unknown, so we use the rank difference $|\sum_{i=1}^{\hat{k}}\hat{F}_i-\sum_{i=1}^{\hat{k}(\mathbf{v})}\hat{F}_i|$ between Y_{approx} and $Y_{approx}(\mathbf{v})$ in the correction set to estimate it.

4.3.3 Discussion

In this section, we further discuss details of the profile generation stage.

4.3.3.1 Correction Set Construction

As introduced in Section 4.3.1, the correction set is necessary for estimating the analytical accuracy of non-random interventions. It can also improve the error bound of random interventions when the correction set can provide substantially more information than the degraded video, as shown in Section 4.5.2. Constructing the correction set requires access to videos with random interventions alone; non-random interventions are not permissible. However, using only random interventions is feasible in many cases. Instead of using a non-random intervention, the administrator might be willing to apply a random one at a very high degradation level. (For example, they may choose a lower sampling rate instead of lowering frame resolution.) In addition, since the correction set is only required in the profile generation stage, it may be acceptable to permit a lower level of degradation for just a limited amount of time.

The correction set should still be degraded by the random interventions as much as possible; in the context of reduced frame sampling, that means minimizing the set's size. However, there is a limit to how much degradation can be introduced, since we want to ensure a tight error bound for the downstream process. According to the definition of the corrected bound in Equation 4.12 and 4.13, when $err_b(\mathbf{v})$ is smaller, the corrected error bound is tighter. Therefore, we need to achieve low $err_b(\mathbf{v})$ while using as few frames in the correction set as possible, i.e., picking the elbow of the curve of $err_b(\mathbf{v})$ against m, the size of the correction set. In our design, we use a simple heuristic to determine the size: the correction set's size is increased gradually by 1% of the total size of the original video to output $err_b(\mathbf{v})$. Once the difference between the current and the previous output is less than 2%, which means the value $err_b(\mathbf{v})$ does not change much with the correction set's

size (i.e., the elbow), or the current size reaches the size limit defined by the administrator, we stop growing the correction set.

If pure random interventions are not allowed or only substantial random interventions are allowed, it may be that no correction set or only a small correction set is possible. In that case, an alternative method to approximate the tradeoff curve may be generating profiles on less privacy-sensitive video at another time. Videos at different times might interact with analytical models in different ways, but they are expected to be visually similar and will yield roughly similar profiles. Experiments in Section 4.5.3 demonstrate that similar profiles arise from visually-similar video collections.

4.3.3.2 Intervention Candidate Design and Time Complexity

Our system first considers many possible sets of destructive interventions (f, p, c). For the reduced frame sampling f, similar to the correction set design, we consider sample fractions at 1% intervals. For reduced frame resolution p and image removal c, we uniformly generate ten frame resolutions and all combinations of possibly sensitive classes. Then administrators filter out the intervention candidates that cannot satisfy degradation goals.

The total time of profile generation includes time for the neural network model to process frames plus the analytical error estimation time. Estimation time is usually negligible compared with the network's image processing time (discussed in Section 4.5.3). Model processing time is $O(N_{model} \cdot T_{model})$, where N_{model} is the total number of model invocations, and T_{model} is the averaged processing time on each frame, including loading, transformation and inference. An early stopping and reuse strategy can be applied

to decrease N_{model}. For each resolution candidate, the error bound is estimated for frame sampling rate candidates in ascending order. In this way, model outputs for frames sampled at a low rate can be reused for the outputs at a high rate, and the estimation process can stop early when the error bound decreases slowly. As a result, the profile generation overhead is modest.

4.4 System Prototype

We implemented a prototype system, SMOKESCREEN, in Python. This system ran on a 64-core (2.10GHz) Intel Xeon Gold 6130 server with 512 GB RAM and 4 GeForce GTX 1080 Ti GPUs. It embodies our novel algorithms and contains three main components: 1) video frame processor, 2) analytical result and error bound estimator, and 3) correction set and intervention candidate design.

Video frame processor — This component processes video frames by calling the UDFs in queries. We use YOLOv4 [20] and Mask R-CNN [65] as two built-in models for detection UDFs. YOLOv4 has been implemented based on a neural network framework, Darknet [147], written in C and CUDA. This model is invoked through a Python interface in this component. And we directly apply a Mask R-CNN implementation based on Keras and Tensorflow [3]. The processed video frames are from decoded videos which are stored on a disk for downstream processing. Only one frame can be loaded, resized, and processed at a time (i.e., no batch computation), and all the model inference procedures run on a GPU.

Analytical result and error bound estimator — This component consists of our estimation algorithms in Section 4.3. The cost of the estimation itself is relatively small.

Correction set and intervention candidate design — This component determines the correction set size and the sets of intervention candidates, working in the profile generation stage. By calling the above estimation component to process video frames of different sizes, the error bound differences are computed and the size of the correction set is determined as stated in Section 4.3.3. This component also interacts with administrators to collect intervention candidates, which will then be sent to the error bound estimator.

4.5 Experiments

We evaluate three core claims about SMOKESCREEN:

1. For random destructive interventions, our algorithm can provide a tighter analytical error bound than competing methods. This holds true for every aggregate query type. (Section 4.5.2)

2. For both random and non-random destructive interventions, the correction set can improve the performance of error bound estimation. And our technique can efficiently determine an appropriate correction set size. (Section 4.5.2)

3. The discussion of profile generation time and profile similarity between similar videos is demonstrated. (Section 4.5.3)

4.5.1 Experimental Setting

We describe our workloads, baselines, and accuracy metrics.

Workloads — We evaluated our system on multiple workloads. Each workload consists of a video dataset, a trained neural network to process video frames, an aggregate

function to collect useful information, and a set of destructive interventions. Every workload was run 100 times, and the experimental results below are the averaged results of 100 trials of the following workloads unless stated otherwise.

- **Video dataset** — The video set is one of either night-street video or UA-DETRAC video. The night-street video is surveillance video of a street in Jackson Hole at night, which is released by the BlazeIt project [84]. It contains 973k frames in total and the frame rate is 30 FPS. We selected one out of every fifty frames (19463 frames) to construct our dataset. The UA-DETRAC video [184] is recorded at Beijing and Tianjin in China. It contains 40 sequences (56k frames) in its test dataset and the frame rate is 25 FPS. We selected 12 sequences (15210 frames) for our experiments.

- **Neural network model** — We used Mask R-CNN [65] for night-street video and YOLOv4 [20] for UA-DETRAC video to detect cars. The detection threshold was set to be 0.7 for both of the models. Although the confidence output associated with each detected object can further improve the detection accuracy when averaged over frames, we just utilized the object output alone in each frame for simplicity because we assume the model output as the ground truth and our work does not try to improve the model's standalone accuracy.

- **Aggregate function** — The aggregate function is one of `AVG`, `SUM`, `MAX`, or `COUNT`. In our experiments, they were used to compute the average, sum, maximum of the number of cars in frames, and count the number of frames that contain cars respectively. For `MAX`, our system estimates 0.99 quantile as an approximation of the

maximum value.

- **Destructive intervention** — A set of destructive interventions is composed of reduced frame sampling fraction, reduced frame resolution, and the restricted class for image removal. We assumed the video with the original length and the highest resolution as the original video. We set the highest resolution to be 640×640 for Mask R-CNN and 608×608 for YOLOv4. In our experiments, the sample fraction can be any value less than one, the frame resolution should be lower than the highest value and meet models' requirements (e.g., the default structure of Mask R-CNN can only handle the resolution in multiples of 64), and restricted classes include "person" and "face". We detected "person" by applying YOLOv4 with detection threshold 0.7, and detected "face" by applying MTCNN [206] with threshold 0.8. Restricting "person" is usually a more strict intervention because people can appear in cameras with unclear faces. According to the detector, 2761 frames (14.18%) contain "person" and 782 frames (4.02%) contain "face" in night-street data; 10018 frames (65.86%) contain "person" and 377 frames (2.48%) contain "face" in UA-DETRAC data. These contained classes for each frame were stored as prior information.

Baselines — We evaluated against the first four baselines for `AVG`, `SUM`, and `COUNT`, and evaluated against the last baseline for `MAX`.

- **EBGS** — The EBGS algorithm [127] is widely used for early stopping when estimated error is within some small number. We directly used it to estimate the query result and error bound instead of using the stopping mechanism.

- **Hoeffding-Serfling** — The upper bound of absolute error can be derived from the

Hoeffding-Serfling inequality [16]. We divided it by the lower bound of the query result in order to obtain the upper bound of relative error for comparison.

- **Hoeffding** — Online aggregation [68] provides the upper bound of absolute error from Hoeffding's inequality. Then we processed it in the same way as above.

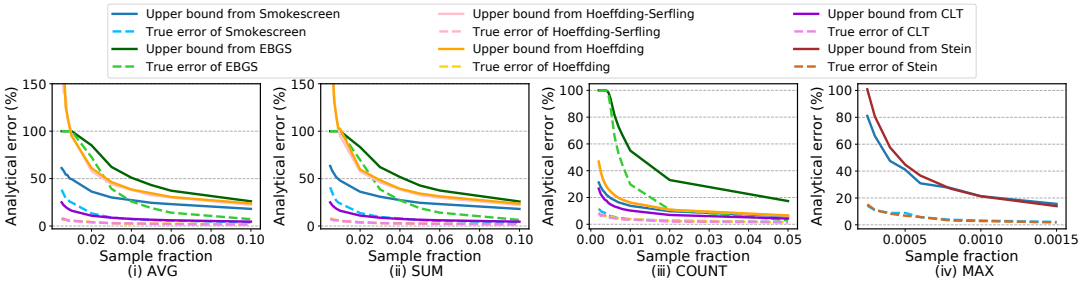
- **CLT** — Online aggregation [68] also provides the upper bound of absolute error from the central limit theorem. Then we processed it in the same way as above.

- **Stein** — [117] minimizes the sample size that can ensure the ϵ-approximate extreme quantile based on Stein's lemma. We directly used it to derive the error bound.

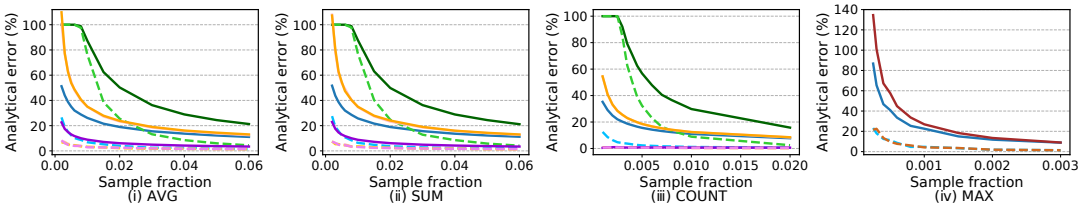
Accuracy Metrics — As introduced in Section 4.3.2, the relative error of the approximate query result compared with the true result was used as the accuracy metric when querying with aggregate functions `AVG`, `SUM` or `COUNT`, and the relative error of the approximate result's true rank compared with the true result's true rank was used when querying with `MAX` in our experiments. We treated the query result without destructive interventions as the true result. Our algorithms computed upper bounds of these relative errors and we compared them with the true relative errors.

4.5.2 Analytical Result and Accuracy Estimation

We show that our query result and accuracy estimation algorithms are effective across a range of video data, models and aggregate query types for both random and non-random interventions.



(a) night-street



(b) UA-DETRAC

Figure 4.4: The true relative error of estimated query result (dashed lines) and error bound (solid lines) computed from Smokescreen and baselines for each aggregate query type on two datasets

4.5.2.1 Managing Random Interventions

Summary — For random destructive interventions, our basic algorithms can provide good estimated analytical results and tighter upper bounds of relative errors compared with reliable competing methods for every aggregate query type. Our error bound can be up to 154.70% tighter than baselines, and the tight bound can enable tradeoffs that are 88% more accurate.

Overview — We evaluated our query result and error estimation algorithms against competing methods on four aggregate query types, two video datasets and two models. When we varied the sample fraction, we did not tune other destructive interventions. And when we varied the frame resolution or restricted class, the sample fraction was set to be

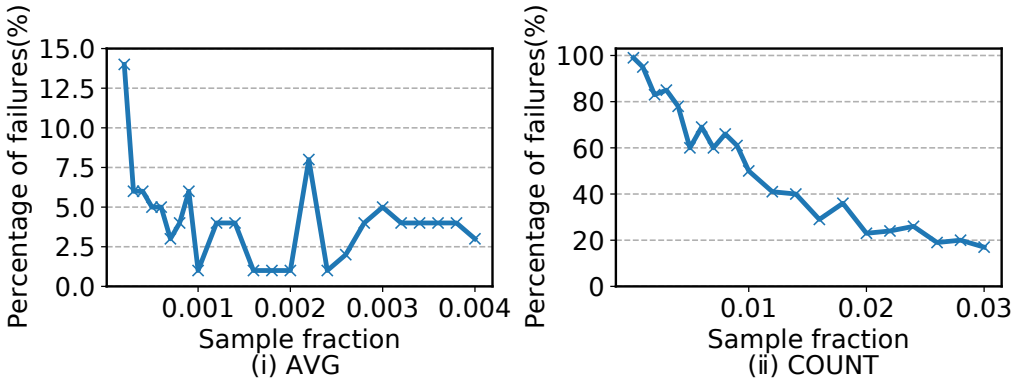
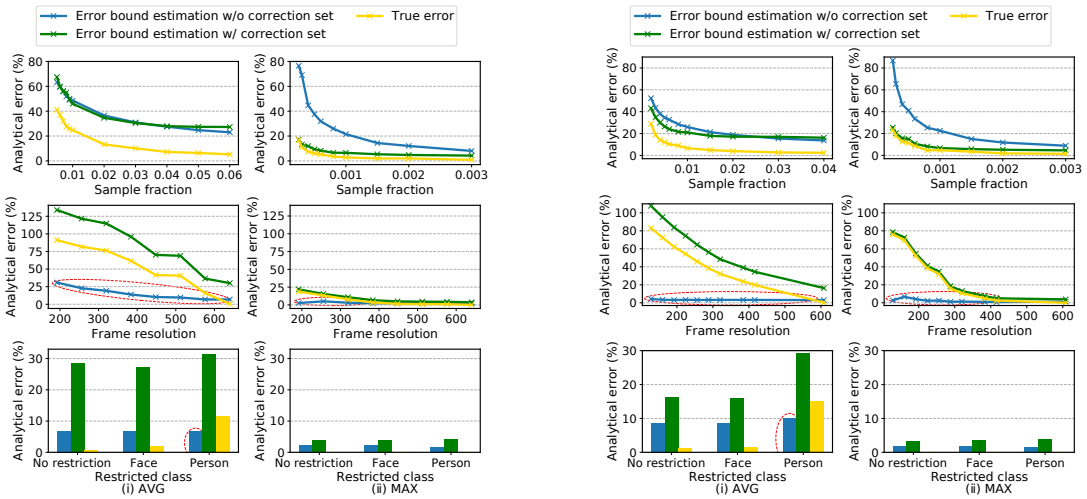


Figure 4.5: The percentage of the situation when the error bound from CLT is smaller than the true error in 100 trials

0.5. For better comparison with baselines, no correction set was used in this experiment.

Results — We show the true relative error of estimated query result and the error bound computed by each method in Figure 4.4. It shows the results varying with the reduced frame sampling intervention. From the true analytical error of SMOKESCREEN (blue dashed lines), we can find that for every aggregate function, the sample fraction increases, the true estimation error goes down and approaches zero. Since the curves have flattened, for the four query types, we end them when the fraction is 0.1, 0.1, 0.05, 0.0015 for night-street video, and 0.06, 0.06, 0.02, 0.003 for UA-DETRAC video. The curves indicate that our algorithm can collect useful information from the samples and show good performance even when the sample fraction is relatively small. When looking at the upper bound from SMOKESCREEN (blue solid lines), we can find that they are always higher than the true error curves, which means that our error estimation algorithm for random interventions truly provides an upper bound of the error.

For `AVG`, `SUM` and `COUNT`, the query result and error estimation of Smokescreen are al-



(a) night-street (b) UA-DETRAC

Figure 4.6: Compare the estimated error bound w/ and w/o correction set with the true error under random and non-random destructive interventions on two video datasets

ways better than EBGS. When compared with Hoeffding and Hoeffding-Serfling, although our result estimation is less precise, a tight error bound is the more important goal. Our error bound can be up to 154.70% tighter (Due to the range of the y-axis, it is not shown in the figure). All of these algorithms, SMOKESCREEN, EBGS, Hoeffding and Hoeffding-Serfling, can ensure these upper bound estimations are greater than the true errors with at least 95% probability. It seems that CLT can provide an even tighter bound than ours. However, CLT can be brittle and unreliable: it cannot always obtain a bound at the 95% confidence level especially when the sample size is small. Figure 4.5 shows the percentage of situations when CLT's error bound is smaller than the true error on UA-DETRAC video in 100 trials. These upper bound estimations would provide misleading information for administrators to determine a set of interventions that yield large error beyond expectations. For `MAX`, our query result estimation is the same as Stein's, but our error bound is

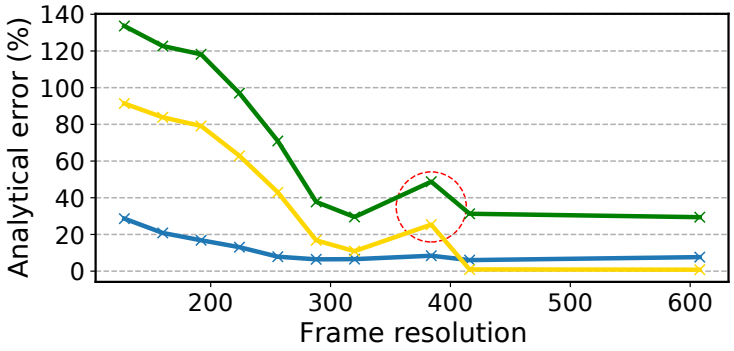


Figure 4.7: Apply YOLOv4 to compute the average number of cars in night-street video. The relative error is abnormally large when resolution is 384×384. The legend is the same as that in Figure 4.6.

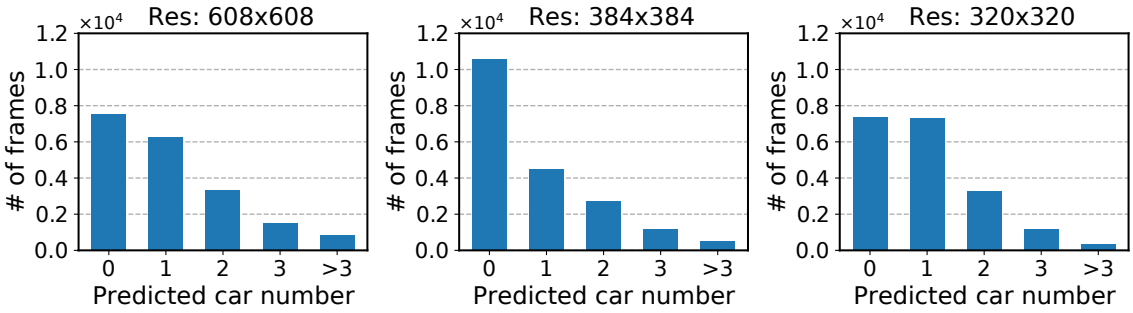


Figure 4.8: Car number distribution predicted by YOLOv4 in night-street video data

tighter when the sample fraction is small.

When non-random interventions are applied, none of the above techniques can provide correct upper bounds, that is, the error estimation cannot be guaranteed to be greater than the true error. These destructive interventions will be handled in the next section.

4.5.2.2 Managing Combinations of Random and Non-random Interventions

Summary — Our error correction algorithm can provide a true error bound when non-random interventions exist, and can further improve basic algorithms' bound estimations

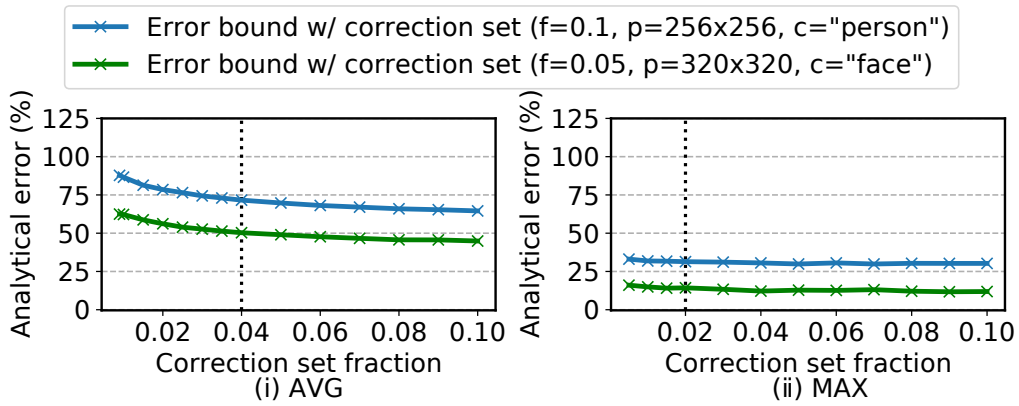


Figure 4.9: The error bound estimation with different correction set sizes for two sets of destructive interventions on UA-DETRAC video

for random interventions in some cases.

Overview — In order to test our error correction algorithm, we compared error bound estimation computed with and without the correction set with the true error under each set of interventions. Because the algorithms for `SUM` and `COUNT` are almost the same as that for `AVG`, we only tested `AVG` and `MAX` functions. We set the sizes of the correction sets according to the correction set construction strategy in Section 4.3.3: 6% of the original frames for function `AVG` and 2% for function `MAX` for night-street video, and 4% for `AVG` and 2% for `MAX` for UA-DETRAC video. Only one kind of intervention was tuned at a time and the other two were fixed. When testing the combination situation when both random and non-random interventions exist, we set the sample fraction to be 0.5 while varying non-random interventions. The only exception was that we set the sample fraction to be 0.1 when changing the restricted class for UA-DETRAC video, because the number of frames that do not contain "person" is less than half of the total number.

Results — Figure 4.6 shows the error bounds with and without the correction set un-

der each set of interventions for AVG and MAX functions. In the second and third rows of Figure 4.6, when the frame resolution is low or the restricted class is "person", the error bound without correction set (blue curve or blue bar), circled in red, can be lower than the true error (yellow curve or yellow bar), so they are wrong and will mislead administrators. It happens because low-resolution objects are hard to be detected by neural network models and the existences of "person" and "car" are very likely to be correlated, both yielding systematic error in samples. Fortunately, the error correction algorithm can solve this problem: the error bound with correction set (green curve or green bar) is always higher than the true error. From the first row, it shows that the correction set is also helpful for random interventions when the size of the set is much larger than the size of the degraded video (that is, it provides more information). When there is only the random intervention, the tighter of the error bounds with and without the correction set is used as the error estimation.

Besides Mask R-CNN, we also applied YOLOv4 to detect cars in night-street video, and we noticed an abnormal situation when querying the average number of cars with frame resolution interventions, shown in Figure 4.7. The estimation error under resolution 384×384, marked in the red circle, is even larger than that under lower resolutions. To find out the reason, we show the predicted car number distribution, i.e., the number of frames that are predicted to contain certain number of cars, in resolution 608×608 (ground truth), 384×384, and 320×320, in Figure 4.8. It shows that the distribution under resolution 320×320 is similar to the true distribution, while that under resolution 384×384 deviates substantially from the truth. Therefore, the neural network's large prediction error causes the inaccurate result estimation. If not provided with degradation profiles,

administrators might unknowingly select this bad intervention that keeps video's good fidelity while yielding a high estimation error. Fortunately, our algorithms can detect this counter-intuitive situation to help administrators make a reasonable tradeoff.

4.5.2.3 Correction Set Size

Summary — Our algorithm, which determines an appropriate correction set size through the change of its error bound with its size, is effective in real cases so that checking the correction set's performance under every set of interventions can be avoided.

Overview — In this experiment, in order to verify that an appropriate correction set size can be directly obtained from its error bound without considering multiple destructive interventions, we tested two sets of interventions and all four aggregation functions on two datasets. These representative sets of interventions were randomly selected: (1) sample fraction 0.1, frame resolution 256×256 and restricted class "person"; and (2) sample fraction 0.05, frame resolution 320×320 and restricted class "face".

Results — The curves of error bound estimation that varies with correction set fraction for `AVG` and `MAX` on UA-DETRAC video are shown in Figure 4.9, and other cases are similar. In this figure, the x-axis, the correction set fraction, is the proportion of the correction set size to the length of the original video. When it is larger, error bounds become smaller and approach true errors. When the fraction is large enough, the slopes of these curves are close to zero, which means more correction data would not make the estimation more precise, so we should stop increasing the size. According to the mechanism in Section 4.3.3, the determined fractions are shown as dotted vertical lines. We can find that even though two sets of destructive interventions' curves are different, the determined

fractions are appropriate choices for both of them because their slopes have dropped down at the intersections with the dotted lines.

4.5.3 Other Experiments

In this section, we discuss the profile generation time and profile similarity between similar videos.

4.5.3.1 Profile Generation Time

Summary — The total time of profile generation is dominated by network model processing time, which is determined by the model and intervention candidates.

Overview — We evaluated the total time of profile generation for the analytical query that employs YOLOv4 to compute the average number of cars in UA-DETRAC video. In the profiles, we set the highest resolution to be 608×608, and ten resolutions were selected as the intervention candidates. And we set the loosest image removal intervention to be no restricted class. As shown in Section 4.5.2, the determined correction set fraction is 0.04 in this case. We also set this value as the highest sample fraction.

Results — YOLOv4 needs to be invoked 6084 times to process 4% of the total frames under every resolution setting, and the total time is around three minutes. Compared with the model processing time, our estimation stage takes only tens of milliseconds for each set of degradation interventions, so the profile generation time is dominated by the former. When the neural network model, the video content, or the intervention candidate settings are different, the profiling time would vary.

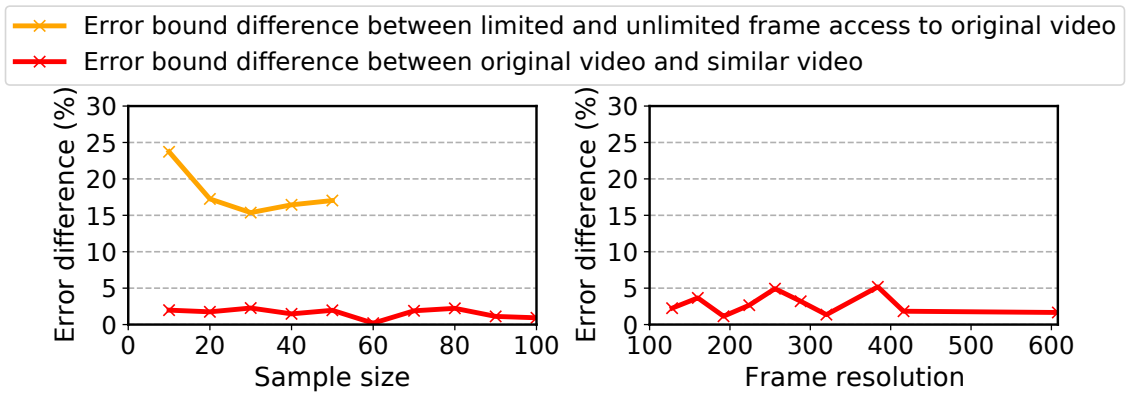


Figure 4.10: Compare two error bound differences when using or not using a similar video.

4.5.3.2 Profile Similarity between Similar Videos

Summary — Similar profiles can be generated from a similar video to guide the trade-off in the original video.

Overview — We computed the profiles of the `AVG` analytical query with YOLOv4 on two video sequences selected from UA-DETRAC dataset. One video (MVI_40771), denoted as video A and set as the original video, is from a traffic monitoring camera at a busy intersection. Another video (MVI_40775), denoted as video B, is captured by the same camera at a different time, and is visually similar to the original video. They contain 1720 frames and 975 frames respectively. We tested reduced frame sampling and reduced resolution interventions and set the correction set size as 500 for both video A and B. We also tested multiple degradation settings for video A when at most 50 randomly sampled frames can be accessed, which may happen due to a high degradation requirement. We compared the target profiles of video A when 500 frames are sampled as the correction set with other profiles by computing the absolute differences.

Results — Figure 4.10 shows the profile differences. In the left figure, the reduced frame sampling intervention is applied with the fixed resolution 608×608. The total number of frames are different in the two video sequences, so we use the sample size instead of the sample fraction as the x-axis for better comparison. And we only show the results when the size is less than 100 because the error bound differences only slightly change beyond this area. The limited frame access (up to 50 frames) to video A causes an incomplete and loose error bound estimation, yielding the substantial difference (orange line) compared with the target profile. Fortunately, when enough frames (500 frames) in video B are accessed, the error bound differences between this similar video and video A (red line) are close to zero. In the right figure, the resolution is varied with the fixed sample size 500. Similarly, the error bound differences between video A and B are very small and always within 5%.

4.6 Conclusion

As video data of public locations is increasingly collected and analyzed, how to balance the analytical query accuracy and other competing goals becomes a problem. In summary, we present a novel video degradation-accuracy profiling model which is able to produce accuracy/degradation tradeoff curves so that administrators can determine a set of appropriate destructive interventions. In addition, we implemented our prototype system, SMOKESCREEN, and verified its good performance on real-world video datasets.

For the analytical accuracy estimation problem in this work, we modeled random and non-random interventions as shown in Table 4.1. This modeling is not restricted to videos — if other scenarios for other data types can be modeled as the same technical problems in

Table 4.1, our algorithms are also applicable. If videos' unique properties are exploited — for example, a sequence of frames are so similar that part of frames can be skipped from processing — the quality of the estimated error bound can be further improved.

CHAPTER V

Optimizing Video Selection Queries With Commonsense Knowledge

5.1 Introduction

With the increased availability and popularity of video databases [99, 4, 89], video selection queries have emerged as a growing area of research interest [8, 37, 115, 85, 84]. 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

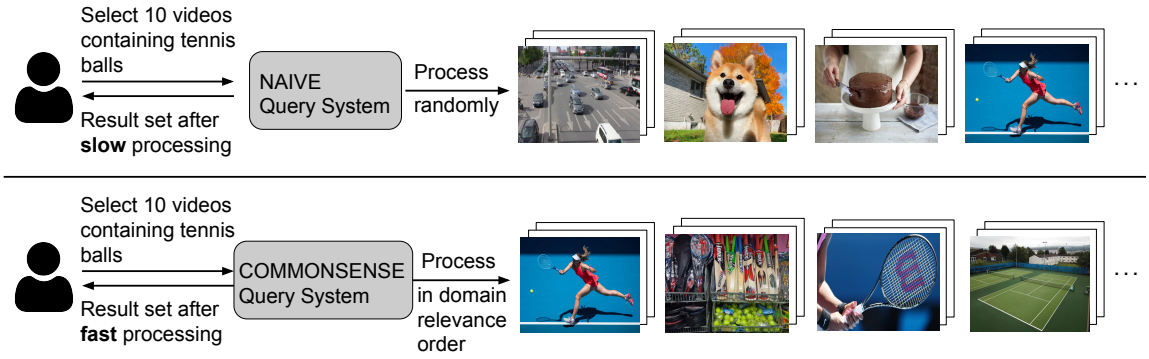


Figure 5.1: A comparison between naïve query system and our query system based on commonsense knowledge

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 [84, 86] 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 [118]). 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. As a consequence of the detector's long inference time [171] and long preprocessing time (e.g., decoding) [87], 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:*

```
1 SELECT * FROM videoCorpus
```

```
2 WHERE DetectedObject = ''Tennis Ball''
3 LIMIT 10
```

She utilizes a naïve query system that simply scans the corpus as shown in the upper part of Figure 5.1. 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 [79, 35]. 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 [85] only works when the video is very static, and the specialized neural network model method [85, 84, 8] 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 [73] 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 [133]), 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 — 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 [85] 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.

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 [83] and Fréchet inequalities [51] 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 [40] 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 5.1. 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 5.2)
- 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 5.3)
- 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 5.4 and 5.5)

5.2 Problem Formulation

In this section, we define the video selection query optimization problem in Section 5.2.1, introduce the query optimization strategy with commonsense knowledge in Section 5.2.2, discuss our design considerations in Section 5.2.3, and elaborate the domain assumptions in Section 5.2.4. All of the notations are listed in Table 5.1.

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 $\rightarrow$ video)	{[tennis racket, person] $\rightarrow V_1$}
$P(\mathcal{O}\|\mathcal{L}_i)$	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 5.1: Frequently used notation

5.2.1 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, topics, etc.). The object detector D, e.g., a neural network such as YOLO9000 [148], is applied to this corpus to identify videos of interest where the target objects $\mathcal{O}$ exist; the target objects $\mathcal{O}$ are defined in the query's predicate, e.g., searching for videos with a "tennis ball". The LIMIT clause with the number k restricts videos that should be selected and returned to the user. 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 (i.e., n is smaller), the total query processing time can be reduced. This leads to the following optimization problem:

QUERY OPTIMIZATION PROBLEM: *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.*

5.2.2 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.*

5.2.3 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 video information may also be utilized as the index. It may be effective to predict objects' existence based on other textual information from videos, such as descriptions generated by video captioning models [192] or video topics identified through video classification [173]. 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. 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 [2, 88, 160]. 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. Moreover, when such features are directly used for object detection at query time, we can expect that our commonsense knowledge models with object lists will still facilitate faster video selection due to the compactness of the object lists and models. Given these considerations, we choose object lists as the index.

5.2.4 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[1], psychedelic films[2], animation instructional videos[3], etc.; (2) videos in which common objects do not exist, for example, typography videos with just text rather than visuals[4], nonrepresentational art videos[5], etc.; (3) videos in which frames do not change much across time, for example, surveillance video in an elevator at night, dash cam video of a car on a congested road, etc. The difference detector method [85] is already designed to handle this static video type. Fortunately, the above video types constitute only a small proportion of existing videos. For other video types, even unusual genres like animation, our method would be beneficial as long as there are semantic relations among video frames.

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

[1]Movie example: Avatar
[2]Video example: `https://www.youtube.com/watch?v=JJkPLYmUyzg`
[3]Video example: `https://www.youtube.com/watch?v=NZbrdCAsYqU`
[4]Video example: `https://www.youtube.com/watch?v=qZEPs3vmYB4`
[5]Video example: `https://www.youtube.com/watch?v=q2Ffbo5fpEQ`

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. Even humans have difficulty reliably identifying these vague objects in videos. Moreover, 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.

5.3 Algorithms

Now, we introduce our video selection query optimization algorithms. In Section 5.3.1, we introduce the overall procedure — our system's three-stage architecture. In Section 5.3.2, we focus on the model preparation stage and elaborate on the construction of commonsense knowledge models. In Section 5.3.3, we further improve the model based on the ground truth collected online.

5.3.1 Overall Procedure

We design a three-stage architecture in our system as shown in Figure 5.2. 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

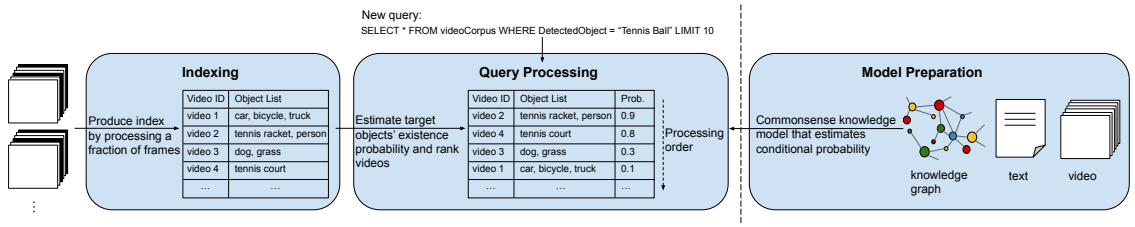


Figure 5.2: The architecture of the overall procedure. In the model preparation stage, an offline stage, commonsense knowledge models that estimate objects' conditional existence probability are constructed from knowledge graphs, text or videos. The whole video corpus goes through the indexing stage; only a fraction of frames (black frames) are processed to create the observed but incomplete object lists as the index. When new queries arrive, videos are ranked based on the index and the probabilistic model in the query processing stage.

incomplete object list $\mathcal{L}_i$. These object lists mapped to videos as key-value pairs compose the index $\mathcal{I}$. The indexing frame rate is adjustable according to varying index time budgets (e.g., sampling one out of thirty frames). 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 7 describes this procedure. In lines 1-7, videos in which target objects $\mathcal{O}$ are observed at index time are added to the result set directly, and the remaining videos will be taken into consideration in the following steps. In line 8, the existence probability of target objects $\mathcal{O}$ conditioned on the imperfect index $\mathcal{I}$ is predicted by the prepared probabilistic model M. A high-quality model will assign high probabilities to predicate-related videos. In line 9, 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 10-15.

Algorithm 7: Query processing

Input: Video corpus $\mathcal{V}$, object detector D, target objects $\mathcal{O}$, LIMIT number k, probabilistic model M, Index $\mathcal{I}$

1 **for** V_i in $\mathcal{V}$ **do**
2 **if** $\mathcal{O} \subseteq \mathcal{L}_i$ **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 $\mathcal{P}$ = M($\mathcal{I}$, $\mathcal{O}$);
9 $\mathcal{V}$ = $\mathcal{V}$[$\mathcal{P}$.argsort()[::-1]];
10 **repeat**
11 V_{select} = $\mathcal{V}$.getNext();
12 **if** $\mathcal{O} \subseteq D(V_{select})$ **then**
13 $resultSet$.append(V_{select});
14 **end**
15 **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 5.3.2.

5.3.2 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 $\mathcal{O}$, this video is handled in lines 1-7 in Algorithm 7 before applying the prob-

abilistic model. In the subsequent parts of this section, we will only consider the index that does not contain target objects. In Section 5.3.2.1, we introduce a conditional probability formula-based model when there are only off-the-shelf knowledge graphs and text as commonsense knowledge sources. In Section 5.3.2.2, we propose a neural network-based model when videos are also available to provide commonsense knowledge.

5.3.2.1 Model Built Without Videos

A wealth of commonsense knowledge is embodied in various existing sources, such as text corpora [165], knowledge graphs [125], 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 [125], ConceptNet [166], Wikidata [181], 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 [166], 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 [183], 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 of the objects. Since object names are often unambiguous, and knowledge graphs like ConceptNet can be sufficiently comprehensive for handling synonyms, we directly match the detected object names with knowledge graph nodes.

First, we estimate the pairwise conditional probability $P(\mathcal{O}|\mathcal{L}_i)$ in which both the target object list $\mathcal{O}$ and observed list $\mathcal{L}_i$ have only one object. The closeness between two node embeddings can be measured by cosine similarity [45], 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 $\mathcal{O}$ and $\mathcal{L}_i$'s word embeddings to estimate the following probability ratio:

$$\frac{P(\mathcal{O} \cap \mathcal{L}_i)}{P(\mathcal{O} \cup \mathcal{L}_i)} := \max\left(\frac{embedding(\mathcal{O}) \cdot embedding(\mathcal{L}_i)}{\|embedding(\mathcal{O})\| \cdot \|embedding(\mathcal{L}_i)\|}, \epsilon\right). \tag{5.1}$$

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 $\mathcal{O}$ 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 [26]. We estimate the

probability by the relative value of the average daily pageview:

$$P(\mathcal{O}) \coloneqq \sqrt{\frac{\text{Avg daily pageview of } \mathcal{O}}{\max\{\text{Avg daily pageview of detectable objects}\}}}. \tag{5.2}$$

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:

$$P(\mathcal{O}|\mathcal{L}_i) = \frac{(P(\mathcal{L}_i) + P(\mathcal{O}))\frac{P(\mathcal{O}\cap\mathcal{L}_i)}{P(\mathcal{O}\cup\mathcal{L}_i)}}{(1 + \frac{P(\mathcal{O}\cap\mathcal{L}_i)}{P(\mathcal{O}\cup\mathcal{L}_i)})P(\mathcal{L}_i)}. \tag{5.3}$$

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 $P(\mathcal{O}|\mathcal{L}_i)$ 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 $\mathcal{O}$ are denoted as $O_1, O_2, ..., O_r$. Similar to the "naïve" conditional independence assumptions in naïve Bayes classifiers [151], 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 $\mathcal{O}$. According to Bayes' theorem [83],

$$P(\mathcal{O}|\mathcal{L}_i) = \frac{P(\mathcal{O})P(\mathcal{L}_i|\mathcal{O})}{P(\mathcal{L}_i)} = \frac{P(\mathcal{O})\prod_{j=1}^{m_i} P(L_{i,j}|\mathcal{O})}{P(\mathcal{L}_i)}. \tag{5.4}$$

In this formula, $P(L_{i,j}|\mathcal{O})$ is calculated by Equation (5.3) if $r = 1$ or calculated by Equation (5.4) if $r > 1$.

Our next step involves estimating $P(\mathcal{L}_i)$ and $P(\mathcal{O})$ in Equation (5.4). According to Fréchet inequalities [51]:

$$\max(\sum_{j=1}^{m_i} P(L_{i,j}) - (m_i - 1), 0) \leq P(\mathcal{L}_i) \leq \min_j P(L_{i,j}). \tag{5.5}$$

Since we can compute the probability of two objects existing in the same video by plugging in Equation (5.1) and (5.2), i.e., $\forall j, k \in [1, m_i], j, k \in \mathbb{N}$,

$$P(L_{i,j} \cap L_{i,k}) = \frac{(P(L_{i,j}) + P(L_{i,k})) \frac{P(L_{i,j} \cap L_{i,k})}{P(L_{i,j} \cup L_{i,k})}}{1 + \frac{P(L_{i,j} \cap L_{i,k})}{P(L_{i,j} \cup L_{i,k})}}, \tag{5.6}$$

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

$$P_{LB}(\mathcal{L}_i) = \max(\frac{1}{m_i - 1} \sum_{1 \leq j < k \leq m_i} P(L_{i,j} \cap L_{i,k}) - (\frac{m_i}{2} - 1), 0), \tag{5.7}$$

$$P_{UB}(\mathcal{L}_i) = \min_{1 \leq j < k \leq m_i} P(L_{i,j} \cap L_{i,k}). \tag{5.8}$$

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 (5.5) for all the combinations, we can obtain Equation (5.7). $P(\mathcal{L}_i)$ is estimated by the mean value of $P_{LB}(\mathcal{L}_i)$ and $P_{UB}(\mathcal{L}_i)$ in our model, and $P(\mathcal{O})$ will be derived similarly. As a result, $P(\mathcal{O}|\mathcal{L}_i)$ in Equation (5.4) can be computed.

5.3.2.2 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 [172]. The distribution of objects portrayed in videos can provide video- and domain-specific knowledge, that is likely more valuable for our system. For example, "tennis racket" and "tennis ball" are probably highly correlated in the commonsense

knowledge from both conventional text sources and video sources. In contrast, ”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 5.3.2.1 is designated as the default model.

In this section, we aim to estimate the probability $P(\mathcal{O}|\mathcal{L}_i)$ 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., $\frac{\text{\# of videos containing } \mathcal{O}}{\text{\# of videos containing } \mathcal{O} \text{ and } \mathcal{L}_i}$ for any possible $\mathcal{O}$ 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 5.3.2.1. However, this formula only integrates pairwise relationships and does

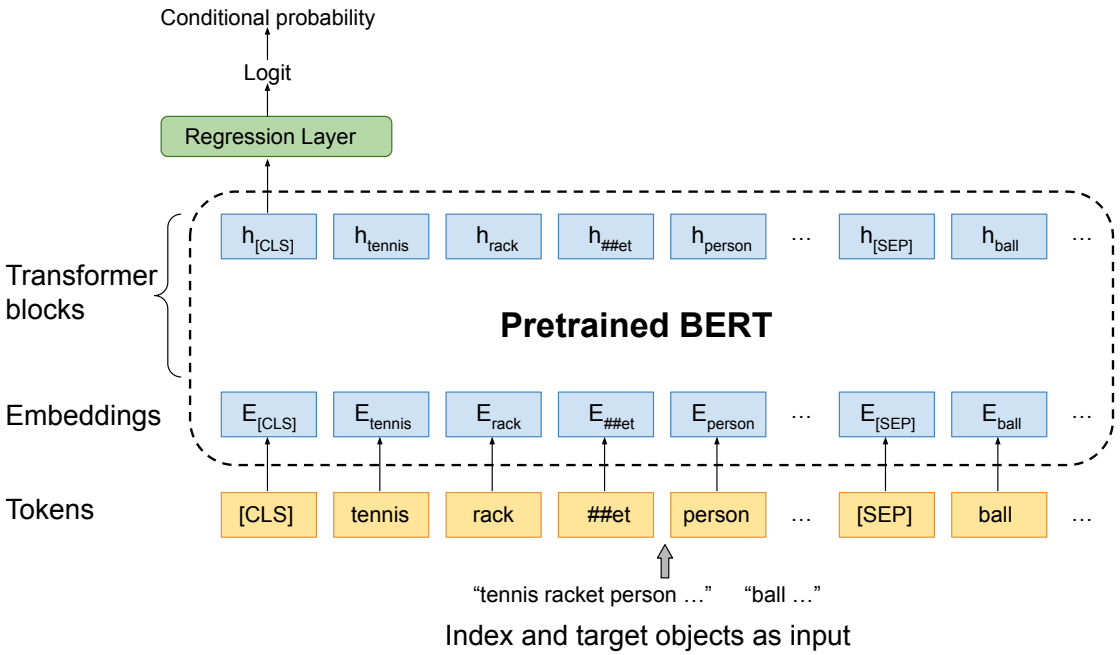


Figure 5.3: The structure of the model built with videos. The input is a sequence of the observed objects in index and target objects; the output is conditional probability prediction.

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. With the development of deep learning, neural network models have emerged as effective tools for learning extremely large and complex probability distributions [104]. 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) [40], 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 [207] 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 [176]. 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 [71].

We start with the case when there is only one target object in a query's predicate. Figure 5.3 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.

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.

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 $\mathcal{O}$ 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 5.5.1.

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.

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.

It needs to be noted that our data generation and training 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.

5.3.3 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 7), 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.

5.4 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 [148] model based on the neural network framework Darknet [147], 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 [187] and PyTorch [136]. 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.

5.5 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 5.5.2).

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 5.5.3).

5.5.1 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 [4] and HowTo100M [124]. These YouTube-style videos comprise important workloads for applications introduced in Section 5.1. 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, simplifying video content representation. 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. When training the BERT regression model, the splitting ratio for training and validation is 3:1. 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 [148] was applied for gener-

ating object statistics for commonsense knowledge and building indexes both at the rate of one frame per second. It was also used for obtaining the ground truth. In our experiments, we considered the object categories in the training data, yielding 1041 distinct objects in YouTube-8M and 1096 objects in HowTo100M. We selected target objects that were in both the above classes 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 [166], 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 5.3.2.1 is set to 0.01. For the model built with videos, we made the amount of training data with labels 1 or 0 balanced. In the training process, the batch size is 128, the number of epochs is 4, the optimizer is Adam algorithm with weight decay [90], 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 [12].

Difference Detector — This method is from NoScope [85]. 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 [94] and Bayesian network [167] can also be employed. However, they are less effective than our neural network model in our tests.

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 5.5.2.

5.5.2 Overall Performance

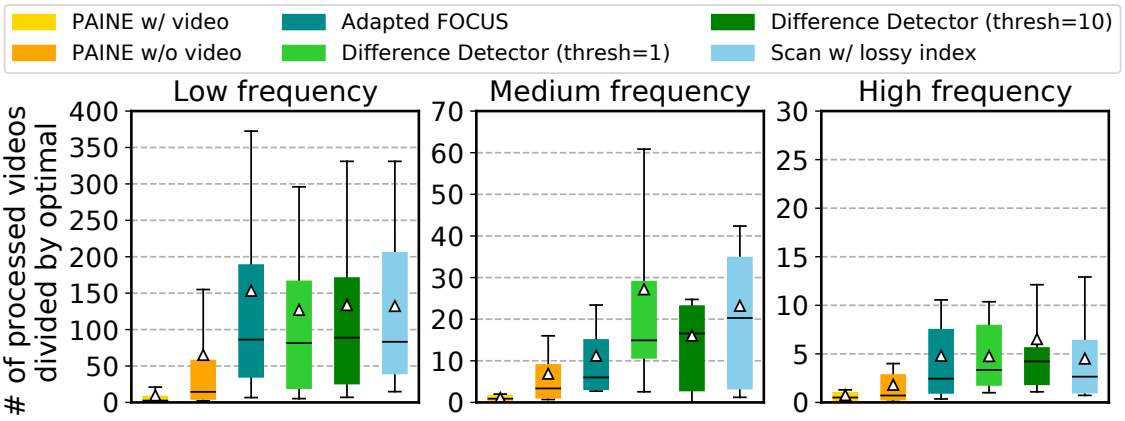
This section shows the overall comparison results between PAINE and the baselines. We evaluated our two models after online learning and tested them on single-target-object and multiple-target-object workloads, as detailed in Section 5.5.2.1 and 5.5.2.2.

5.5.2.1 Single Target Object

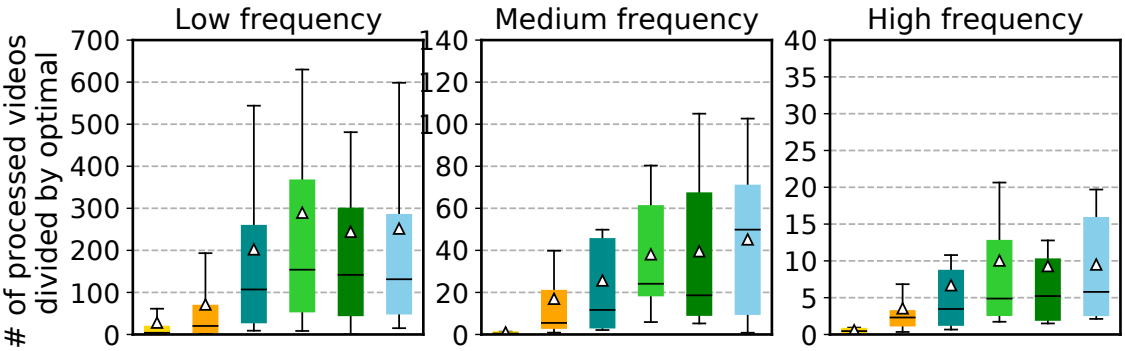
Summary — PAINE can beat all the baselines when there is one single target object in each query. When comparing the model built with videos to baselines, we observed a significant improvement of up to 97.79% over Scan with lossy index while ensuring the same index cost (around 3% of the video collection). In some cases, there may not be available object statistics from videos. 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 representative LIMIT number, 20% of the total number of satisfying video clips for each query. We removed target objects that have satisfying videos of the required size observed at index time. For a fair comparison, all three baselines have the same index budget as ours. For the Scan and Adapted FOCUS methods, the observed object lists in the index are exactly the same as ours. For the Difference Detector method, we set the difference threshold to 1 and 10, and indexed 60 frames for each video.

To infer unseen objects from the lossy index, leveraging large language models seems like a promising and easy approach. Hence, we compared our two models with this approach. The quality of LLM's output depends on the prompt. If we just wrote the prompt ourselves, the results might primarily reflect the prompt rather than LLMs in general. Therefore, we collected various prompts from 10 students (these prompts are shown with



(a) YouTube-8M



(b) HowTo100M

Figure 5.4: 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.

the source code on Github), spanning three solution types: direct object list ranking, video index number ranking, and video scoring. We chose the most popular LLM from OpenAI [22], namely GPT. We evaluated the version 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.

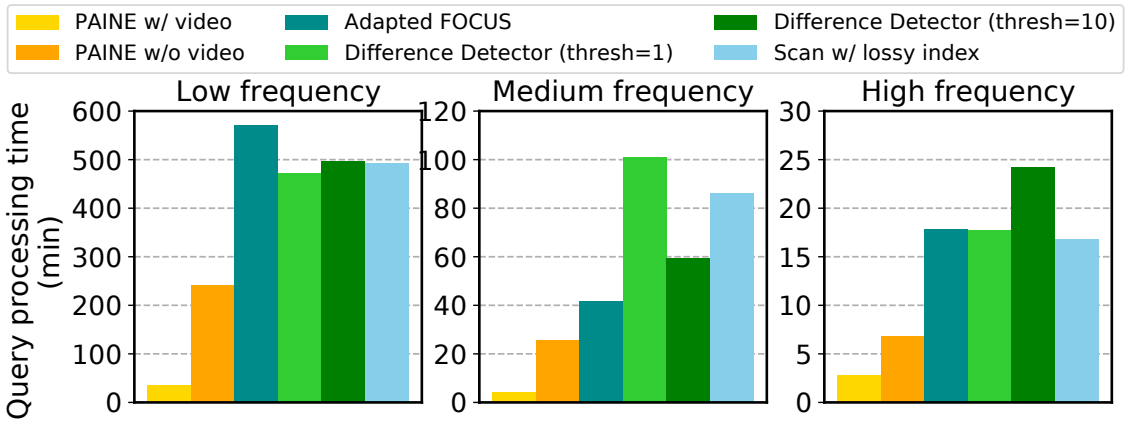
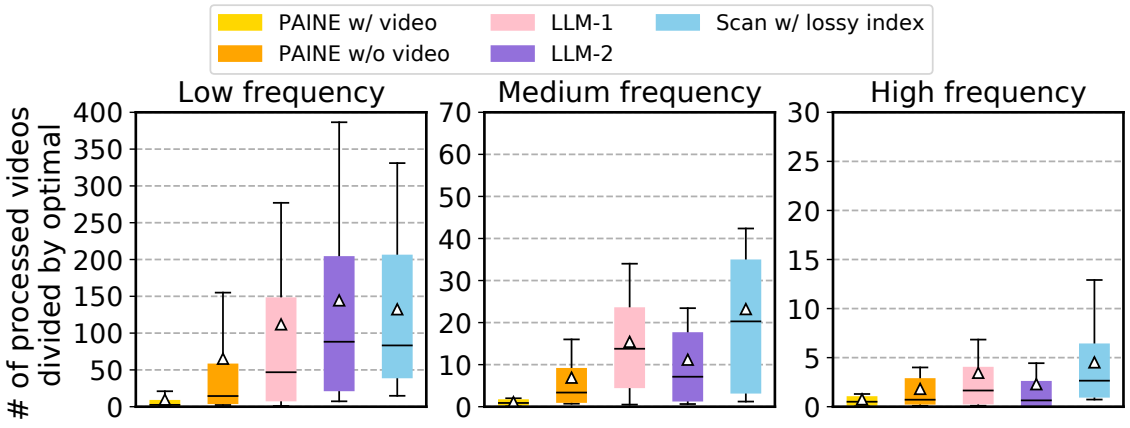


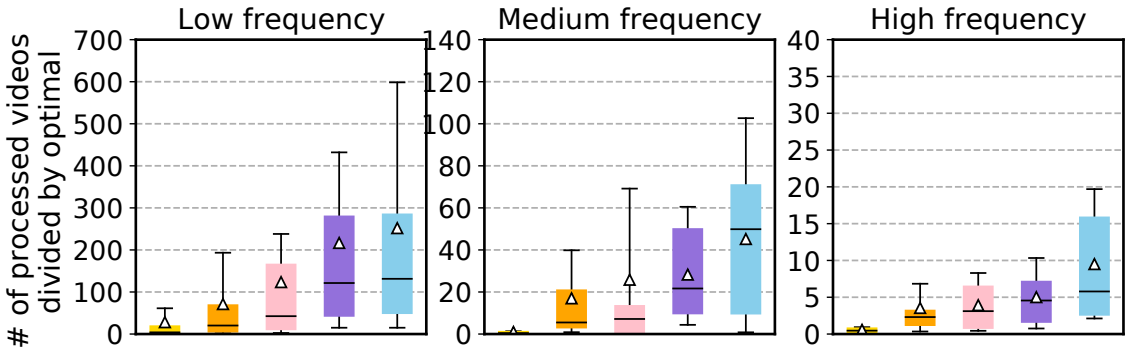
Figure 5.5: Compare the average query processing time of PAINE and baselines on YouTube-8M dataset.

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 5.4. 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.

From left figures to right figures, 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



(a) YouTube-8M



(b) HowTo100M

Figure 5.6: Comparison of our commonsense knowledge models against a large language model (GPT-3.5) using the optimal two prompts from a selection of ten.

capture visual differences. Adapted FOCUS (dark cyan box) is better than other baselines as it can capture similarities between video clips at index time. 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. When it is hard to obtain commonsense knowledge from appropriate videos in advance, the model built without videos would be a good choice. 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 query processing time. Our commonsense knowledge models bring extra overhead due to the assignment of probabilities and video ranking, but this overhead is minor. For example, it takes our BERT regression model around 5.3 seconds to predict probabilities and rank videos 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 5.5 shows the average query processing time of PAINE against baselines on YouTube-8M dataset. The results are aligned with Figure 5.4a.

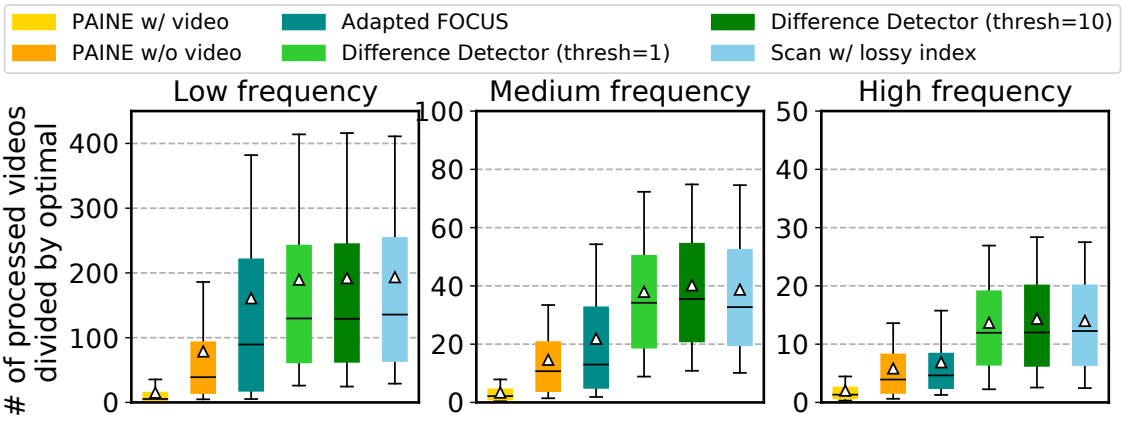
Besides the above baselines from previous research, we compared our commonsense knowledge models with the LLM (GPT-3.5), as depicted in Figure 5.6. We evaluated ten diverse prompts and showed the best two in this figure. Even though the LLM is pow-

erful in addressing various natural language inquiries and it indeed exhibits performance improvement over a simple scan, it is not as useful as our models in this video selection task. By analyzing the LLM's responses, we find that it can return a few "obvious" answers, i.e., videos containing objects that are closely associated with target objects, but then return bad results. This 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 (e.g., simply repeating). Furthermore, good performance cannot be achieved even when processing the prompt for video scoring. Although the factors mentioned above are not concerns in this scenario, LLM still struggles to provide precise scoring.

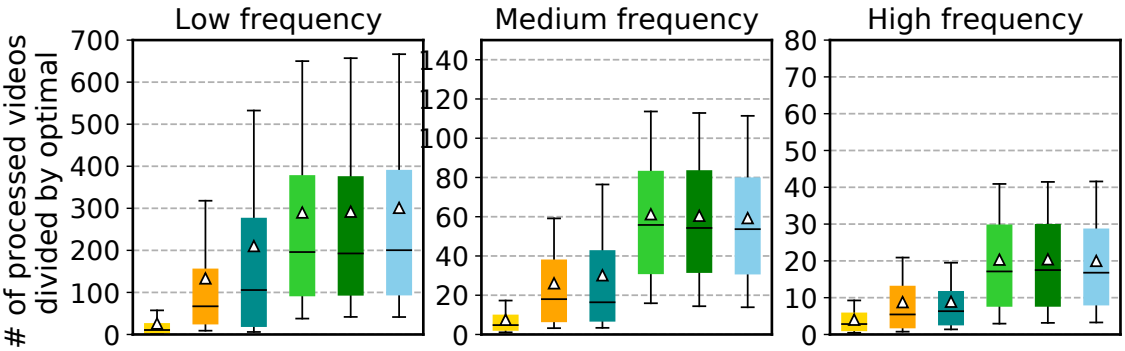
5.5.2.2 Multiple Target Objects

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.

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, and it is categorized either as a member of the low-frequency, medium-frequency, or high-frequency group; (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



(a) YouTube-8M



(b) HowTo100M

Figure 5.7: Compare PAINE with baselines when there are two target objects in each query.

arriving videos. Overall, we constructed 8051 and 10004 queries containing two target objects and 13298 and 17541 queries containing three target objects for each dataset respectively.

Results — Figure 5.7 and Figure 5.8 show the comparison results for two-target-object and three-target-object workloads. In Figure 5.7, 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

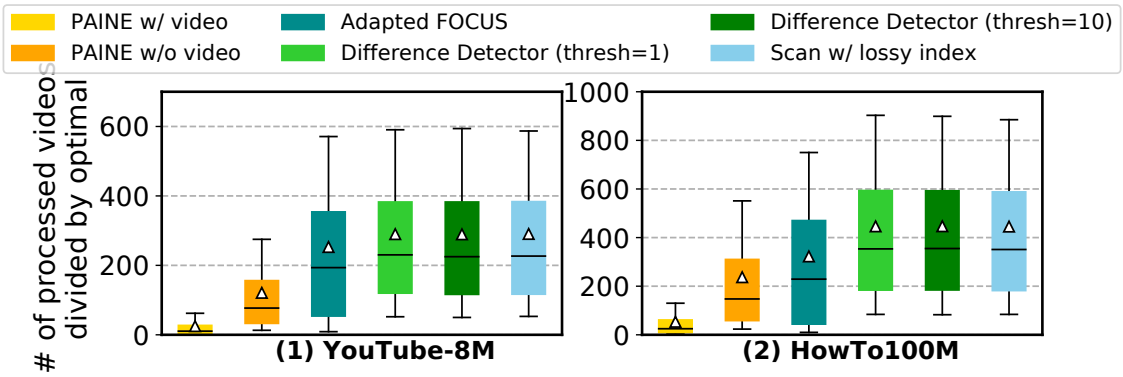


Figure 5.8: Compare PAINE with baselines when there are three target objects in each query.

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 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 5.8. 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, and the model without videos can be effective for target objects with low frequency.

5.5.3 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 5.5.3.1, different amounts of object statistics for model training in Section 5.5.3.2, the effect of the online learning strategy in Section 5.5.3.3, and different LIMIT numbers in Section 5.5.3.4. We tested single-target-object queries in the low-frequency group for all the following experiments, and we set the LIMIT fraction to 20% in Section 5.5.3.1-5.5.3.3.

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 5.5.2 because all three baselines rely directly on the index and they would reduce to a simple scan method in the hard mode.

5.5.3.1 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. We compared our two models in PAINE with a simple scan.

Results — Figure 5.9 shows the performance of PAINE and Scan under different index

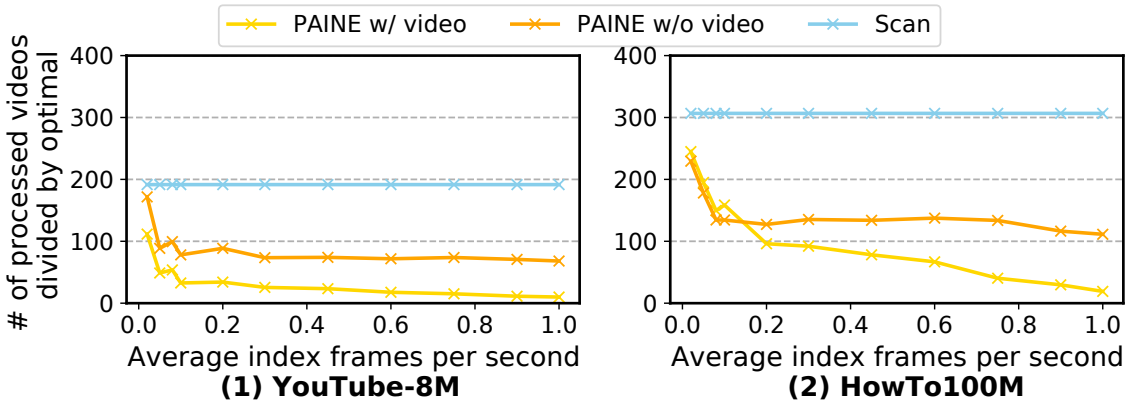


Figure 5.9: The performance of optimization methods when varying the index size.

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.

5.5.3.2 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 the default setting, we assigned a large portion of the video corpus to

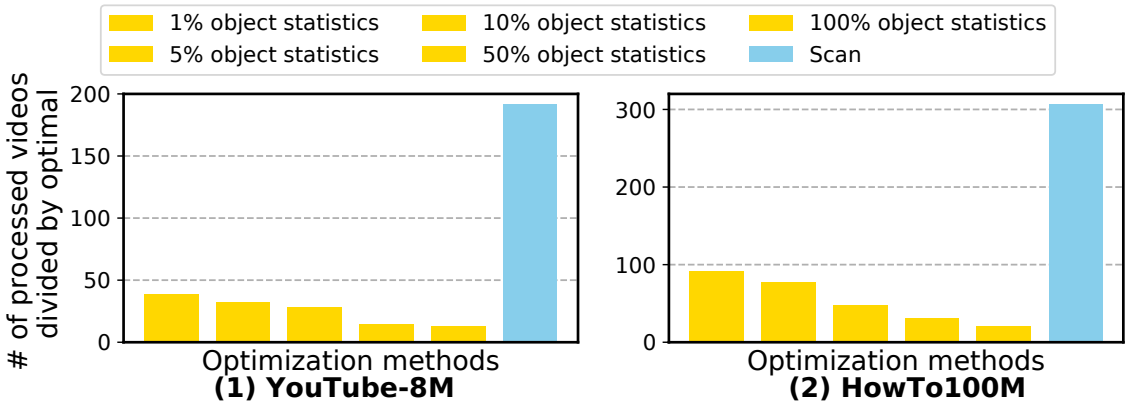


Figure 5.10: The performance of optimization methods trained with different amounts of videos

generate object statistics from videos for model training (11766 video clips in YouTube-8M and 19183 video clips in HowTo100M) and validation (3922 video clips in YouTube-8M and 6395 video clips in HowTo100M). In practice, in the model preparation stage, it might not be feasible to collect this 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 5.10 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.

5.5.3.3 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, compared with no online learning, 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 in the query processing stage.

5.5.3.4 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

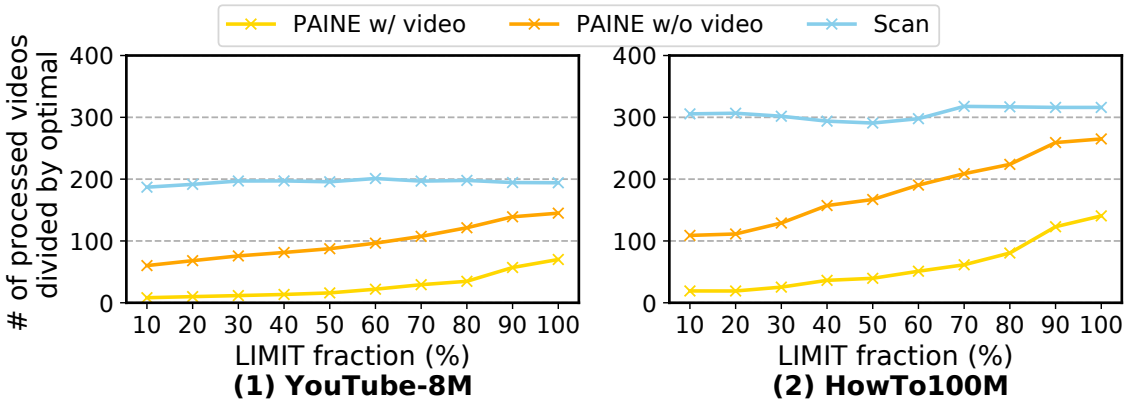


Figure 5.11: The performance of optimization methods when varying the LIMIT value.

fractions ranging from 10% to 100%, with intervals of 10%.

Results — Figure 5.11 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.

5.5.4 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.

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 commonsense knowledge-based 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 regarding target objects, 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 5.5.2.2 demonstrate that PAINE can effectively retrieve rare videos that contain atypical combinations of target objects.

It is worth noting that the query processing procedure can be further improved during the invocation of the object detector, for example, batch processing can be allowed as in voodoo indexing [67], or specialized NNs and the difference detector can be applied if a small error can be tolerated. 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.

5.6 Conclusion

As deep learning models are becoming increasingly popular in object detection, it is common for video selection queries to include these models in the predicates. Due to the

long inference time of detection models, it is crucial to develop optimization techniques to accelerate the query processing. 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.

CHAPTER VI

Uncovering Confounders from Images for Causal Inference

6.1 Introduction

Causal inference is the process of identifying the causal relationships between attributes in data, helping analysts determine whether and how much one factor independently influences the outcome. It is widely used across various domains, such as medicine [141], public policy [49], education [58], etc. Because randomized controlled trials are not always feasible due to constraints such as budget limitations and ethical considerations, it becomes more appealing to estimate causal effect using observational data. However, such works suffer from confounding bias [139], one of the major concerns in the field of causality — confounding variables that are associated with both the factors being studied and the outcome make it challenging to figure out the genuine causal relationship. To address the confounding bias, one primary obstacle is that the confounders may be absent from the data [196].

When the data is incomplete for causal inference, it needs to be augmented with attributes selected from external sources. It has been explored to mine potential confounders and integrate data with tables [157, 162], knowledge graphs [195], and text [202]. In addition to these sources, there might be images associated with the studied attributes, such as patient record tables with medical images. It is important to note that images could potentially contain confounding variables, yet they have not been investigated.

EXAMPLE. *Sansa, a manager of a chain of markets, devised a novel pricing strategy. In order to figure out its effectiveness, she applied the policy across all the markets for a few days, recording the sale revenue. Subsequently, she input both the old and new revenue data into a causal inference system, which analyzed policy's impact on sales. Additional information from the market database, such as the market location and product categories, was incorporated for identifying external confounding variables. The causal inference system indicated that the policy had a positive influence on sales, leading Sansa to continue this strategy.*

However, after a full season, Sansa noticed that the sales revenue had actually declined compared to previous values. It turned out that the weather condition played an important role as a confounding variable — the new pricing policy had only been tested during rainy days by coincidence, and the rainy weather boosted the revenue due to increased umbrella prices. Unfortunately, this weather variable had not been included into the causal inference analysis, leading to distorted results. If video frames from surveillance cameras outside the markets are provided for the causal inference system to extract the weather information, the causal effect of the new policy could be accurately estimated, avoiding the financial losses in this season.

To prevent the confounding bias in the above example, we aim to uncover confounders from images for causal inference. This is not a trivial task due to the unstructured nature of image data. Unlike tables and knowledge graphs in which attributes are explicitly listed, image attributes are hidden in pixels and required to be extracted before integration. Regarding the feature extraction, we face two main challenges. First, it is unclear what specific attributes should be extracted. While one might intuitively consider extracting objects present in the images, the task remains ambiguous due to the various subcategories associated with objects. Additionally, there exist other relevant features to consider, such as weather conditions. The attributes can be highly domain-specific, depending on the images and the studied causal relationships. Attempting to pre-define all the features for a comprehensive solution is unrealistic. Second, feature extraction from images is difficult even when the attribute names are known. A variety of deep learning models have been designed to improve extraction efficiency compared with manual extraction. However, most models are only pretrained for specific tasks, for example, YOLOv5 [81] can detect 80 object categories in MS COCO dataset [112] but cannot distinguish the color of traffic lights. It is impractical to collect all the specific models for various attributes.

In this paper, we tackle the above challenges and propose a novel causal inference pipeline that extracts potential confounding variables from images. Our system accepts an incomplete causal table with specified treatment and outcome names, along with a set of images associated with each tuple, as the input. We design a user interface that collects the names of potential confounders from users. This interface covers a wide range of image features, including object presence, object status, etc., enabling this pipeline to function effectively across diverse domains. In order to extract such features from images,

we utilize the state-of-the-art visual language model, GPT-4 with vision (GPT-4V) [134], which can process both textual and visual information. We design the appropriate prompts to achieve accurate feature extraction. When high accuracy cannot be guaranteed by using the visual language model, our system will employ a lightweight classifier trained on visual embeddings from CLIP [144] to enhance accuracy. Furthermore, we leverage GPT-4V to recommend domain-specific features that complement the confounding information from the user interface. Our pipeline integrates the original table with the extracted visual features, and it outputs the average treatment effect (ATE) from the complete causal table using an existing causal graph generation tool.

In our evaluation process, we collected real-life images for testing across three distinct causal graphs. These causal graphs contain diverse visual features as the confounding variables, including various object statuses and variables with different value ranges. To challenge the robustness of our pipeline, we devised varying levels of difficulty when it came to extracting the confounding variables and gathered images accordingly. Our results demonstrate that our system can successfully identify confounders within images and accurately compute the independent causal effect between the treatment and outcome.

6.2 Approach

In this section, we first introduce our overall procedure about uncovering potential confounding variables from images to correct causal inference in Section 6.2.1. Then, we describe each component of the workflow in detail in Section 6.2.2 - 6.2.4.

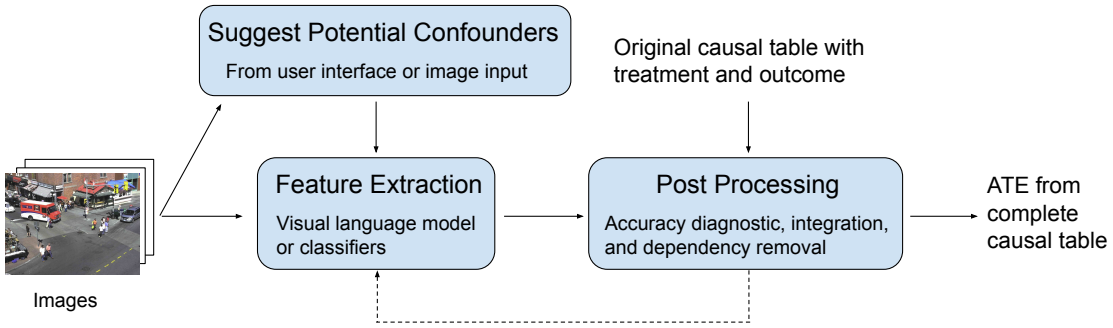


Figure 6.1: The architecture of our system to uncover potential confounders from images.

6.2.1 Overall Procedure

Our end-to-end system architecture that supports the extraction of potential confounders from images is shown in Figure 6.1. Users can submit a relational table, serving as the original causal table, and designate two specific table attributes as the treatment and outcome variables. Additionally, they incorporate relevant images as external sources of confounding factors. Our system outputs the important causal parameter, average treatment effect (ATE), which quantifies the average differences in outcomes between the treatment group and the control group, for example, the revenue difference resulting from applying the new pricing policy or not.

In this workflow, our system initially acquires potential confounder names directly from users or through image analysis, such as "weather" (as detailed in Section 6.2.2). Subsequently, it extracts values of these variables from the image set, such as "rainy" and "not rainy" for the variable "weather" (as elaborated in Section 6.2.3). These identified attribute values are then forwarded to the post-processing component (as outlined in Section 6.2.4) for extraction accuracy diagnostic, integration with the original causal table, and functional dependency removal to yield a complete causal table. Much like other

works in the field of causal inference, we assume this table adheres to the fundamental principles of causal inference — consistency, positivity, and exchangeability. We employ an existing causal discovery tool [1] that is built upon DoWhy [161, 18] to identify the confounders from the attributes. With these confounders accounted for, the tool computes ATE and the associated p-value. If the p-value is greater than the threshold 0.05, users have the option to expand the dataset by collecting more tuples and images and rerun the entire system.

6.2.2 Suggest Potential Confounders

The potential confounding variable names can be obtained directly from the users when they possess domain knowledge. Our user interface, as shown in Figure 6.2, enables users to input feature names related to objects and the environment. In the first category, users have the flexibility to specify the object's name and its attributes (e.g., color, shape, location, etc.) in text, and choose the feature type from options like the existence condition ("existence"), the quantity of this object ("count"), or its status ("status"). When "status" is selected, users can provide a concrete description in the right-most text boxes. In the second category, users can pick the type of environment-related feature from "weather", "outdoor/indoor", and "brightness" and define the status of these features, such as "rainy" and "not rainy".

Due to the rich information contained in images and the extensive background knowledge involved, it is often challenging for users to provide a comprehensive list of the potential confounder names. Besides the above user interface, it would be great if our system can further assist users to address this issue. One possible approach is to propose poten-

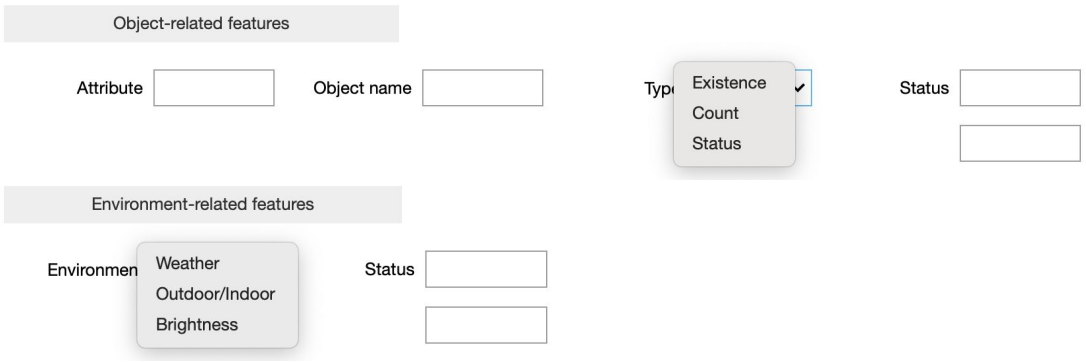


Figure 6.2: Our user interface for users to provide the names of potential confounders

tial confounding variable names from knowledge graphs. However, this approach has two weaknesses. First, it can lead to an excess of potential confounders, some of which may not even be relevant to the images. Consequently, processing images for these variables in the feature extraction stage (Section 6.2.3) would be a waste of computational resources and time. Second, knowledge graphs are limited; there may exist new relationships in images that have not been incorporated into knowledge graphs. To address these issues, our system adopts a more direct approach by generating potential confounder names from the image data. Visual language models can recognize image content and are enriched with general knowledge because they have been trained on vast amounts of text and visual data. They can provide general insights including the status of objects, for example, the traffic light can be red or green. Our system generates prompts to process various images using GPT-4V(ision) and records essential information, including the identified objects, their numbers, locations, common statuses, and current statuses within the image. We filter out objects that always exhibit the identical statuses across images, distinguishing different objects based on their category and location. Such objects just serve as background elements and would not contribute information to causal inference. We present these objects

and statuses to users, allowing them to make informed selections.

6.2.3 Feature Extraction

After the system has acquired the potential confounding variable names, the next step involves extracting their corresponding values from each image. If there exist specialized pre-trained computer vision models tailored to detect these objects, this would simplify the task. However, due to the domain variance, a more general approach is required. Visual language models, such as Google Bard, GPT-4V(ision), and others, which are designed to understand visual content, offer a fitting solution. These models take natural language input that describes the task and they are able to address visual challenges in various fields.

Our system employs GPT-4V API for the extraction of the potential confounders. We generate prompts to communicate with the model, prompting it to produce concise answers post-parsing. For example, a typical prompt may take the form of "Is the status of this traffic light red or green? Answer in one word." In addition to these basic prompts, we implement the following prompt techniques to further improve the extraction performance. First, we leverage the power of few-shot learning by integrating a few images, each annotated with user-defined labels, into the prompt. This technique encourages the model to learn from these examples. We ensure that the examples are consist of images with various labels. Second, we include domain knowledge into the prompt to provide additional context and background information. For example, we may include the information like, "The top light corresponds to red. The bottom light corresponds to green." Third, we employ other prompt practices to make the model "think", for example, adding the content "give me the reason in the second line" to encourage the model to think about

the underlying rationale.

6.2.4 Attribute Post-processing

The extracted potential confounding variables are then fed into the post-processing component to generate a high-quality causal table. The accuracy of confounder values can affect the causal inference process. If the values deviate significantly from the ground truth, their capacity to correct the Average Treatment Effect (ATE) between the treatment and outcome diminishes. To tackle this issue, we have implemented a diagnostic mechanism. For each variable, users are required to annotate at least 1% of the total images. If the extraction results fall below the accuracy threshold δ_c (for categorical data) or exceed the difference threshold δ_n (for numerical data), we consider the extraction quality to be inadequate and our system will revert to the feature extraction component. In this situation, a general-purpose visual language model struggles to perform effectively, leading our system to rely on specialized classifiers for support. We train a lightweight classifier, a logistic regression model, using visual embeddings from CLIP based on images and labels provided by users.

The precisely extracted attributes are then incorporated into the original causal table. We align the sequence of images with the row numbers of the original causal table, enabling the attributes to be integrated as new columns directly. Adhering to the positivity assumption in causal inference, we remove attributes exhibiting functional dependencies or high intercorrelations.

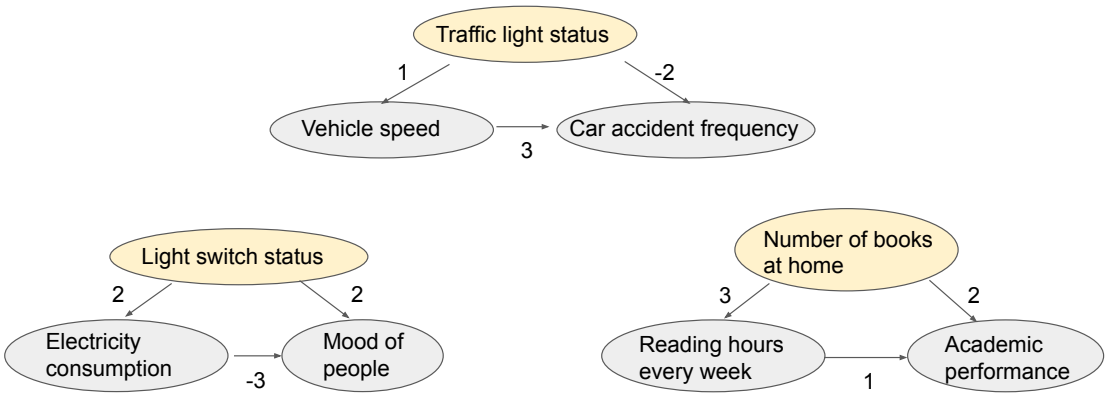


Figure 6.3: Three causal directed acyclic graphs (DAGs) tested in our evaluation

6.3 Evaluation

In this section, we introduce our datasets and experimental settings in Section 6.3.1. We show our results sequentially — we compare the accuracy of ATE from our system with baselines in Section 6.3.2, demonstrate how feature extraction quality affects ATE's accuracy in Section 6.3.3, vary the size of causal table in Section 6.3.4, and evaluate causal inference cost in Section 6.3.5. All experiments were conducted on a 40-core Intel Xeon Gold 6248 server with 384GB RAM and 2 Nvidia Volta V100 GPUs, and a Macbook Pro with the Apple M1 chip and 16GB memory.

6.3.1 Datasets and Experimental Settings

When testing the effectiveness of our system, we confronted the absence of available relational data associated with images prepared for causal inference research. To address this limitation, we created three new datasets which comprised three distinct causal DAGs as shown in Figure 6.3, and each DAG is characterized by a unique confounding variable

type:

- **Traffic light case:** The treatment, high vehicle speed, can independently cause higher outcome, the frequency of car accidents. Traffic light status is a confounding variable — green light causes an increase in vehicle speed while decreasing the frequency of car accidents, as drivers are not required to alter their driving mode. This confounder can be represented as two distinct values: "green" denoted by 1 and "red" denoted by 0, with these states distinguishable through color.

- **Light switch case:** The treatment, high electricity consumption, can diminish people's mood. The light switch status is a confounding variable — when the light is turned on, it consumes more electricity but elevates people's mood due to increased brightness. This confounder can be represented as two values: "on" denoted by 1 and "off" denoted by 0, with these states distinguishable based on the position of the light switches.

- **Education case:** The treatment, more reading hours every week, can boost academic performance. The number of books at home is a confounding variable — a greater number of books at home, which indicates more favorable educational environment, usually results in longer reading hours and better academic performance. This confounder can take a range of non-negative integer values.

Because the causal discovery tool that we employed operates under the common assumption of linear causal relationships, the attributes in our designed DAGs have linear interconnections. The linear coefficients are denoted by the edge weights in Figure 6.3.

To align our data with these DAGs, we generated tables and collected a diverse set of

images captured from various perspectives and locations to reflect the real causal relationships, incorporating small errors that follow a normal distribution. In addition, we ensured that camera-related information, including image resolution, remained constant in each dataset, treating them as unmeasured confounding variables. Furthermore, for the easy scenario, traffic light case, we gathered images categorized into three tiers of complexity, ranging from simple scenarios with a single traffic light, to situations featuring other objects alongside a traffic light, and even cases with two traffic lights, one of which exerted a causal influence.

In the feature extraction component, we invoked GPT-4 API to utilize the model "gpt-4-vision-preview", setting the image resolution as high, the maximum number of tokens as 300, and the temperature as 0. In the post-processing component, we established the accuracy threshold δ_c to be 0.8 and the difference threshold δ_n to be 1. As for the existing causal discovery tool, we used the default back-door adjustment setting.

6.3.2 ATE Accuracy

To test the effectiveness of our system, we compare the ATE output from the following methods:

- **No image:** There is no image input — the ATE is computed based on the original causal table.
- **YOLOv5**: In the feature extraction stage, we utilized an object detection model, YOLOv5 [81], to analyze each image. We generated a table where each column corresponds to a distinct object category, with each cell recording the count of each

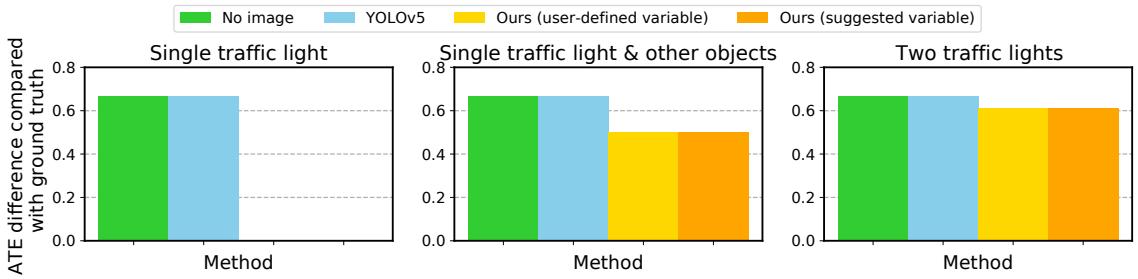


Figure 6.4: We compare the relative ATE difference of our system (with user-defined variable or with suggested variable) with baselines (no image or YOLOv5) in the traffic light case. From left to right, we increase the complexity level of feature extraction — image input contains (1) single traffic light, (2) single traffic light and other irrelevant objects, or (3) two traffic lights with the left one acting as a confounder.

object in each image. This table was then concatenated with the original causal table.

- **Ours (user-defined variable)** We run our system based on user-specified confounding variables. We assume that all the confounders are provided by users through our user interface (Figure 6.2). For example, in the traffic light case, users input "traffic light" as the object name, select the type "status", and proceed to specify the statuses as "red" and "green".

- **Ours (suggested variable)** We run our system based on the potential confounding variables suggested by our system as introduced in Section 6.2.2, independent of users' domain expertise.

We show the comparison results of the above four methods in the traffic light case in Figure 6.4. The y-axis represents the relative difference between ATE and the ground truth with a lower value indicating a better outcome. From left to right, the original causal table is the same, but the inputed images are becoming more complex — they contain single

traffic light, single traffic light with other irrelevant objects, and two traffic lights with the left one affecting the treatment and outcome, separately in each subgraph, making the feature extraction stage more difficult. When there is no image input, the causal discovery tool cannot find any confounding variable, computing the total effect of the treatment on the outcome, which is 66.7% smaller than the true ATE. When there exists images as the external resource and YOLOv5 is used in the feature extraction stage, new object variables are introduced but none of them are considered as the confounder. The reason is that YOLOv5 cannot distinguish green traffic lights and red traffic lights and the detectable objects are not causally related with the color status of traffic lights. Therefore, YOLOv5 cannot improve the no image baseline. In the left figure, our method shows superior performance — the computed ATE is almost the same as the ground truth, because our prompt enables GPT-4V to 100% distinguish the color of traffic lights. Even when users do not provide this confounding variable name, our system can figure out the common status of traffic light is red or green, suggesting a useful potential confounder. When the images become more complex, the accuracy of our methods decreases but our results are still closer to the ground truth compared with baselines. In the method where our system suggests the potential confounders, irrelevant variables are also suggested, such as the existence of people. Fortunately, they do not impact the accuracy of ATE because they are eliminated due to no variance across images or they are not selected as the confounder candidates by the causal discovery tool. However, due to the increased number of variables, it would take more time to extract features. The runtime will be discussed in Section 6.3.5.

The results of the other two harder cases, the light switch case and the education case, are presented in Figure 6.5. In the light switch case, YOLOv5 cannot distinguish the

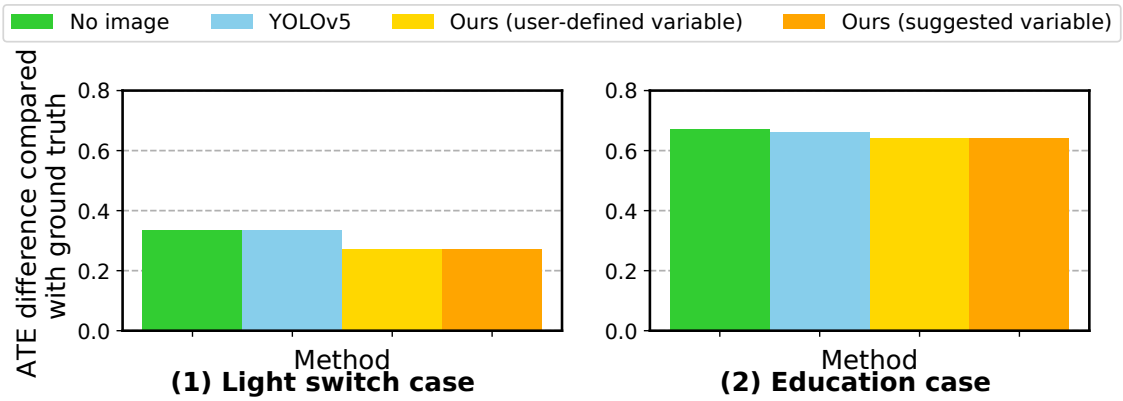


Figure 6.5: The ATE comparison results of the light switch case and the education case.

on or off status of light switches, showing the same results as the no image baseline. When running our system, the detection accuracy of GPT-4V is below the threshold of 0.8 as it struggles to accurately determine the switch's direction. This triggers the sytem to return to the feature extraction stage to train a logistic regression model based on the CLIP embeddings. This model can enhance ATE results compared to baselines. In the education case, YOLOv5 can detect "book" but it is hard for YOLOv5 to output the precise book number. Therefore, its performance falls between the no image baseline and our system. Due to the difficulty of computing the exact number of books, our system is only slightly better than baselines. However, it can reveal the confounders and discern the direction of ATE changes, underscoring the importance of obtaining high-quality confounder values for precise causal inference analysis.

6.3.3 Effect of Feature Extraction Accuracy

In Seciton 6.3.2, our analysis revealed that the computed Average Treatment Effect (ATE) displayed varying levels of accuracy across different methods and datasets. This

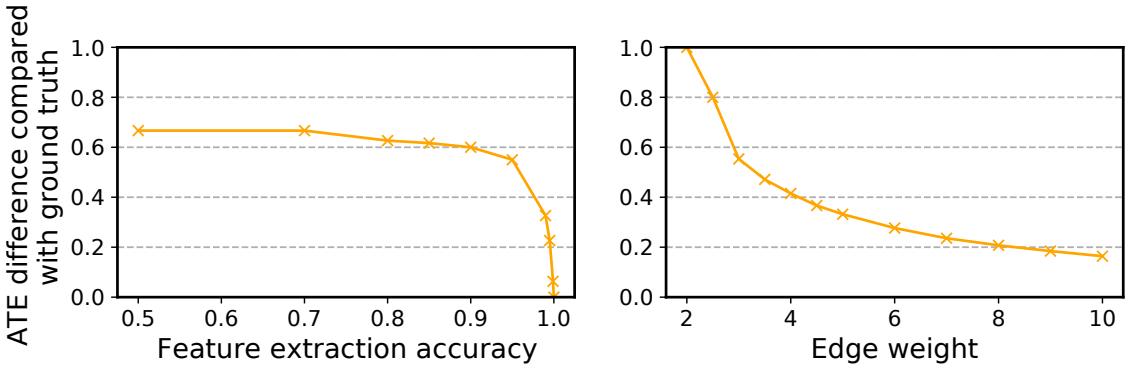


Figure 6.6: The relative ATE difference of our system when varying the feature extraction accuracy and the causal DAG edge weight (the ground truth of causal effect) in the traffic light case.

variability can be attributed to differences in feature extraction precision. When the causal table contains inaccuracies, the causal inference task becomes challenging, as these errors must be discerned and mitigated. In this section, we use the traffic light case to investigate how the accuracy of feature extraction impacts the accuracy of ATE calculations. Specifically, we varied the accuracy of the binary classification in traffic light color and the edge weights within the causal DAG.

We present the results of ATE difference compared with the ground truth of causal effects in Figure 6.6. The left figure illustrates the impact of varying feature extraction accuracy, ranging from 0.5 to 1, achieved by randomly assigning traffic light colors based on the specified accuracy levels. As expected, as feature extraction accuracy increases, the ATE difference decreases, signifying more precise causal inference outcomes. When the extracted variables are 100% accurate, there is no difference between the computed ATE and the ground truth, underscoring the effectiveness of the adopted causal discovery tool. Notably, ATE accuracy exhibits minimal fluctuation when feature extraction accu-

racy is low, but it experiences rapid changes when feature extraction accuracy is large. This indicates the importance of enhancing feature extraction accuracy, as even marginal improvements yield significant benefits when accuracy is already high.

In the right figure, we maintain a constant feature extraction accuracy of 0.95 while varying the edge weight between vehicle speed and the frequency of car accidents in Figure 6.3 from 2 to 10, thereby altering the ground truth of ATE. This figure reflects that the effect of the feature extraction depends on the causal DAG. With consistent feature extraction accuracy, higher edge weights correspond to greater ATE accuracy. This phenomenon arises because, when the edge weight between the treatment and the outcome is substantial, the indirect causal effect becomes relatively minor, thereby reducing the impact of confounding value errors on the computed ATE's accuracy.

6.3.4 Causal Table Size

We have conducted tests of the causal inference task on large causal tables which contain 100,000 rows. When causal tables are too small to yield robust ATE calculations, our system takes measures to ensure the user is alerted and prompted to supply additional data for a more reliable outcome. In such scenarios, our system provides users with the p-value from the causal discovery tool, the probability of obtaining results that are at least as extreme as the computed results under the null hypothesis. To evaluate how the size of causal tables affect the ATE, we varied the number of rows in the traffic light case when providing images containing single traffic light and other irrelevant objects.

Figure 6.7 illustrates the results under different sizes of causal tables. The y-axis on the left side (depicted in red) is ATE difference compared with ground truth, and the y-axis

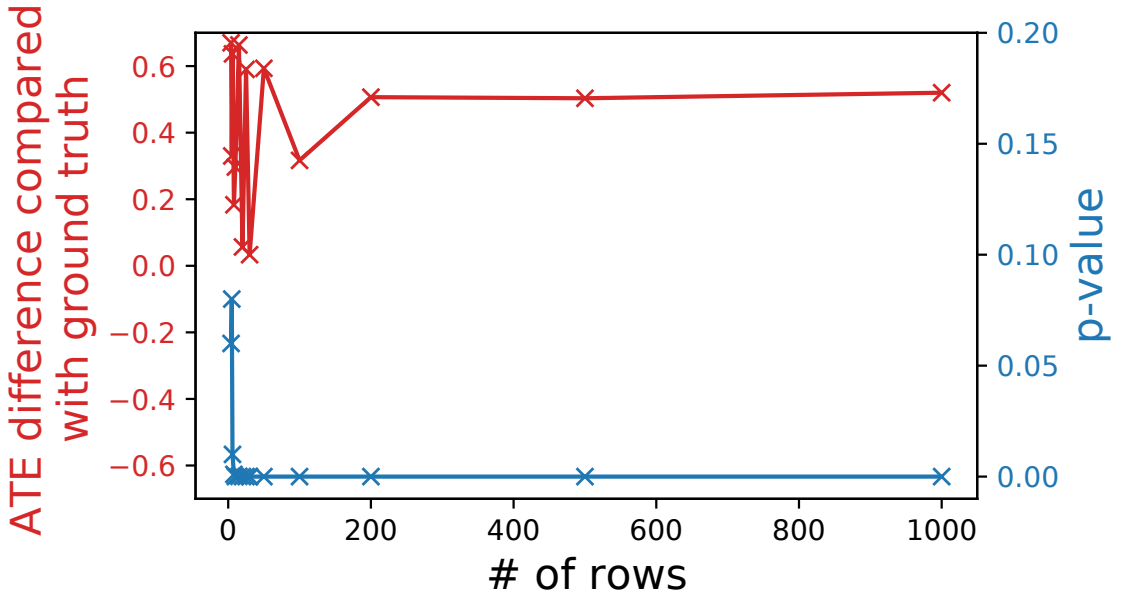


Figure 6.7: The results of ATE accuracy and p-value when varying the size of causal tables in the traffic light case.

on the right side (depicted in blue) represents the p-values. When the number of rows is minimal, the p-value deviates from zero, and the computed ATE exhibits substantial fluctuations. This behavior is due to insufficient evidence to establish statistical significance. As the dataset size increases, the p-value is close to 0 but ATE fluctuations persist. A possible reason is the presence of non-deterministic relationships between variables, characterized by errors following a normal distribution. These errors were introduced during data generation. When the number of rows becomes sufficiently large, such as exceeding 200, the p-value approaches zero, and the computed ATE stabilizes. This observation suggests that increasing the causal table size mitigates uncertainty but does not eliminate the effects of feature extraction errors.

6.3.5 Causal Inference Cost

In this section, we studied the computational costs of causal inference when using our system. To illustrate, let's consider the scenario of the traffic light case, where we work with images containing a single traffic light. The post-processing stage and the application of the causal discovery tool to propose confounding variable candidates and compute the ATE are highly efficient, requiring only approximately 0.3 seconds. The time consumed during the potential confounder suggestion stage and the feature extraction stage dominates the total causal inference duration. This primarily results from the time-intensive invocation of GPT-4V. Its inference time depends on the token length, which is approximately six seconds for each image in the traffic light case. Therefore, the total time required for causal inference is nearly linear with both image size and the number of extracted variables. If the system suggests a substantial number of potential confounders, the overall process would experience slower execution.

6.4 Conclusion

In this work, we designed a new approach to mitigate confounding bias in causal inference by uncovering potential confounders from images. Our system leverages the state-of-the-art visual language model and classifiers to extract user-defined and domain-specific features from images. We demonstrated its ability to find out potential confounders for causal table integration and compute independent causal effects in our evaluation. This research reveals the importance of incorporating visual information into causal inference.

CHAPTER VII

Conclusion and Future Work

7.1 Conclusion

My dissertation is dedicated to addressing the challenges in query processing within video database management systems. In order to mitigate the slow processing issues caused by the large size of video databases and the prolonged inference times required for information extraction, we propose two novel indexing mechanisms to accelerate query processing and facilitate the selection of desired videos and video frames. In our system, VOODOO INDEXING, we introduce item indexes that are organized based on similarity so as to intelligently oversample from groups that are more likely to satisfy query predicates. In addition, within our system PAINE, we construct a cost-effective object index by processing sampled frames and augmenting the index using commonsense knowledge to suggest potential video candidates.

Furthermore, we address the challenging trade-offs between accuracy and competing goals in video analytics, which include system requirements driven by the immense size of video databases, as well as unique privacy considerations for video data. To assist admin-

istrators in balancing these goals, we compute approximate tradeoff curves that quantifies the relationship between video degradation and analytical accuracy.

Lastly, to support causal inference queries in datasets that include images, a challenge task due to the unstructured nature of image data, we develop a comprehensive pipeline. This pipeline is designed to extract confounding variables from images, thereby enhancing the accuracy of causal effect estimation.

7.2 Future Work

In this dissertation, we have presented four works, each offering unique perspectives on improving query processing within video database management systems. In the near future, there are promising research opportunities in each of these areas.

In VOODOO INDEXING, we focus on basic filter queries with one UDF predicate. We view this as the initial step towards a family of methods for opaque query optimization. We believe it may be possible to apply our method to filter queries with multiple predicates by adjusting the reward function. Applying our method to aggregation queries that include a selection is likely to be straightforward, but using a UDF in the HAVING clause of an aggregation query is a serious challenge. This latter situation might appear in large machine learning workloads that use database queries to feed training data to a vast number of model production pipelines; we believe that examining the larger context in which opaque filter queries operate is a promising direction for future projects. Finally, we would like to address the optimization challenge for more elaborate queries such as JOIN queries. A possible way is to adapt our mechanism for cardinality estimation.

In SMOKESCREEN, we focus on the video aggregate queries with frame-level detection

models. Even though they can cover a variety of cases, there exists another type of model, which processes frame sequences, e.g., a RNN model for action recognition and detection. Because reducing the sampling rate likely decreases the accuracy of the model's outputs, simply considering it as a random intervention seems inappropriate. In this situation, both of our algorithms for random and non-random interventions cannot be directly applied. In addition, besides the four commonly used aggregate functions, `AVG`, `SUM`, `MAX`, and `COUNT` in our work, more aggregate types can be explored, such as `VAR`. We believe examining the degradation-accuracy profiling problem for more neural network model types and aggregate functions, as well as exploiting videos' unique properties, are promising future projects.

PAINE opens a new direction to optimizing video selection queries 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. 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 actions detected from multi-frame sequences. It is promising to infer correlations between the above elements from commonsense knowledge.

When it comes to causal inference, videos open up more opportunities, as it contains richer information, especially in terms of temporal characteristics. Within videos, exclusive and valuable temporal information can be extracted, including but not limited to object speed, trajectories, human actions, and event durations. These temporal features may also act as confounding variables in causal relationships. Besides, the temporal nature of videos can help determine causal directions due to its inherent asymmetry. It is

promising to extend the exploration of confounding variables to video data. Additionally, in causal graphs, visual attributes can serve not only as confounders but also as other causal variable types, such as mediators and colliders. A compelling future direction can be constructing complex causal graphs that incorporate both traditional table attributes and visual attributes. Such causal DAGs could provide a more comprehensive depiction of the underlying causal relationships in the world. Moreover, this work has the potential to be adapted and tested in diverse domains, such as medical research, opening up possibilities for interdisciplinary collaboration.

In addition to these directions closely related to our works, there exist other promising avenues for future exploration within the topic of query processing in video database management systems. Firstly, there is a growing number of video streams that require near real-time query processing. The indexing mechanisms are inadequate in this context, prompting the need for new approaches to query optimization. Secondly, it is important to investigate effective storage methods for videos, which can be viewed as high-dimensional vectors. Such methods can significantly impact scalability, performance, and privacy considerations, etc. Lastly, there is high value to develop a video database management system supporting more intricate queries. While current visual language models, like GPT-4V, have the capacity to handle complex queries in natural language, it is challenging for these models to achieve a higher level of accuracy.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] From logs to causal analysis: Extracting data to diagnose large systems. Manuscript under review.

[2] S. H. Abdulhussain, B. M. Mahmmod, M. I. Saripan, S. Al-Haddad, T. Baker, W. N. Flayyih, W. A. Jassim, et al. A fast feature extraction algorithm for image and video processing. In *2019 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2019.

[3] W. Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. `https://github.com/matterport/Mask_RCNN`, 2017.

[4] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[5] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. Jordan, S. Madden, B. Mozafari, and I. Stoica. Knowing when you're wrong: building fast and reliable approximate query processing systems. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 481–492, 2014.

[6] 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. ACM, 2013.

[7] M. R. Anderson and M. Cafarella. Input selection for fast feature engineering. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 577–588. IEEE, 2016.

[8] M. R. Anderson, M. Cafarella, G. Ros, and T. F. Wenisch. Physical representation-based predicate optimization for a visual analytics database. In *2019 IEEE 35th*

International Conference on Data Engineering (ICDE), pages 1466–1477. IEEE, 2019.

[9] E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris, and I. Patras. Video summarization using deep neural networks: A survey. *Proceedings of the IEEE*, 109(11):1838–1863, 2021.

[10] R. T. Apteker, J. A. Fisher, V. S. Kisimov, and H. Neishlos. Video acceptability and frame rate. *IEEE multimedia*, 2(3):32–40, 1995.

[11] A. Aqil, A. O. Atya, S. V. Krishnamurthy, and G. Papageorgiou. Streaming lower quality video over lte: How much energy can you save? In *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*, pages 156–167. IEEE, 2015.

[12] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, et al. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1383–1394, 2015.

[13] S. Athey, R. Chetty, and G. Imbens. Combining experimental and observational data to estimate treatment effects on long term outcomes. *arXiv preprint arXiv:2006.09676*, 2020.

[14] J.-Y. Audibert, R. Munos, and C. Szepesvári. Tuning bandit algorithms in stochastic environments. In *International conference on algorithmic learning theory*, pages 150–165. Springer, 2007.

[15] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[16] R. Bardenet, O.-A. Maillard, et al. Concentration inequalities for sampling without replacement. *Bernoulli*, 21(3):1361–1385, 2015.

[17] F. Bastani, S. He, A. Balasingam, K. Gopalakrishnan, M. Alizadeh, H. Balakrishnan, M. Cafarella, T. Kraska, and S. Madden. Miris: Fast object track queries in video. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1907–1921, 2020.

[18] P. Blöbaum, P. Götz, K. Budhathoki, A. A. Mastakouri, and D. Janzing. Dowhy-gcm: An extension of dowhy for causal inference in graphical causal models. *arXiv preprint arXiv:2206.06821*, 2022.

[19] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, 2005.

[20] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[21] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[22] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

[24] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(May):1655–1695, 2011.

[25] A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[26] Y. Cao, H. Mehta, A. E. Norcross, M. Taniguchi, and J. S. Lindsey. Analysis of wikipedia pageviews to identify popular chemicals. In *Reporters, Markers, Dyes, Nanoparticles, and Molecular Probes for Biomedical Applications XII*, volume 11256, pages 24–41. SPIE, 2020.

[27] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*, 2010.

[28] B. Chandramouli, J. Goldstein, and A. Quamar. Scalable progressive analytics on big data in the cloud. *Proceedings of the VLDB Endowment*, 6(14):1726–1737, 2013.

[29] 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.

[30] S. Chaudhuri, V. Narasayya, and S. Sarawagi. Efficient evaluation of queries with mining predicates. In *Proceedings 18th International Conference on Data Engineering*, pages 529–540. IEEE, 2002.

[31] Z. Che, G. Li, T. Li, B. Jiang, X. Shi, X. Zhang, Y. Lu, G. Wu, Y. Liu, and J. Ye. D ̃2-city: A large-scale dashcam video dataset of diverse traffic scenarios. *arXiv preprint arXiv:1904.01975*, 2019.

[32] R. Chen, N. Mohammed, B. C. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, 4(11):1087–1098, 2011.

[33] M. X. Cheng, L. Ruan, and W. Wu. Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 4, pages 2638–2645. IEEE, 2005.

[34] D. Comer. Ubiquitous b-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137, 1979.

[35] B. F. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A fast index for semistructured data. In *VLDB*, volume 1, pages 341–350, 2001.

[36] J. Dai, J. Wu, B. Saghafi, J. Konrad, and P. Ishwar. Towards privacy-preserving activity recognition using extremely low temporal and spatial resolution cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 68–76, 2015.

[37] M. Daum, B. Haynes, D. He, A. Mazumdar, and M. Balazinska. Tasm: A tile-based storage manager for video analytics. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1775–1786. IEEE, 2021.

[38] M. Daum, B. Haynes, D. He, A. Mazumdar, M. Balazinska, and A. Cheung. Tasm: A tile-based storage manager for video analytics. *arXiv preprint arXiv:2006.02958*, 2020.

[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[41] K. Douglas and S. Douglas. *PostgreSQL: a comprehensive guide to building, programming, and administering PostgresSQL databases*. SAMS publishing, 2003.

[42] P. DuBois. *MySQL*. Pearson Education, 2008.

[43] C. Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[44] Y. Fang, K. Kuan, J. Lin, C. Tan, and V. Chandrasekhar. Object detection meets knowledge graphs. International Joint Conferences on Artificial Intelligence, 2017.

[45] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*, 2016.

[46] A. Fedorov, K. Nikolskaia, S. Ivanov, V. Shepelev, and A. Minbaleev. Traffic flow estimation with data from a video surveillance camera. *Journal of Big Data*, 6:1–15, 2019.

[47] W. Feller. *An introduction to probability theory and its applications, vol 2*. John Wiley & Sons, 2008.

[48] D. A. Fidaleo, H.-A. Nguyen, and M. Trivedi. The networked sensor tapestry (nest) a privacy enhanced software architecture for interactive analysis of data in video-sensor networks. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 46–53, 2004.

[49] A. Finkelstein and N. Hendren. Welfare analysis meets causal inference. *Journal of Economic Perspectives*, 34(4):146–167, 2020.

[50] A. Fire and S.-C. Zhu. Using causal induction in humans to learn and infer causality from video. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 35, 2013.

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

[52] B. C. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (Csur)*, 42(4):1–53, 2010.

[53] E. Gan, J. Ding, K. S. Tai, V. Sharan, and P. Bailis. Moment-based quantile sketches for efficient high cardinality aggregation queries. *arXiv preprint arXiv:1803.01969*, 2018.

[54] J. Gao, T. Zhang, and C. Xu. 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*, volume 33, pages 8303–8311, 2019.

[55] M. N. Garofalakis and P. B. Gibbons. Approximate query processing: Taming the terabytes. In *VLDB*, pages 343–352, 2001.

[56] W. S. Geisler and J. S. Perry. Real-time foveated multiresolution system for low-bandwidth video communication. In *Human vision and electronic imaging III*, volume 3299, pages 294–305. International Society for Optics and Photonics, 1998.

[57] J. Goldstein, R. Ramakrishnan, and U. Shaft. Compressing relations and indexes. In *Proceedings 14th International Conference on Data Engineering*, pages 370–379. IEEE, 1998.

[58] M. Gopalan, K. Rosinger, and J. B. Ahn. Use of quasi-experimental research designs in education research: Growth, promise, and challenges. *Review of Research in Education*, 44(1):218–243, 2020.

[59] B. S. Graham, C. C. d. X. Pinto, and D. Egel. Efficient estimation of data combination models by the method of auxiliary-to-study tilting (ast). *Journal of Business & Economic Statistics*, 34(2):288–301, 2016.

[60] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66, 2001.

[61] M. B. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 275–285, 2004.

[62] M. Grgic, K. Delac, and S. Grgic. Scface–surveillance cameras face database. *Multimedia tools and applications*, 51:863–879, 2011.

[63] A. D. Gupte, B. Amrutur, M. M. Mehendale, A. V. Rao, and M. Budagavi. Memory bandwidth and power reduction using lossy reference frame compression in video encoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(2):225–230, 2011.

[64] B. Haynes, A. Mazumdar, M. Balazinska, L. Ceze, and A. Cheung. Lightdb: A dbms for virtual reality video. *Proceedings of the VLDB Endowment*, 11(10), 2018.

[65] 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.

[66] 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.

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

[68] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 171–182, 1997.

[69] J. M. Hellerstein and M. Stonebraker. *Predicate migration: Optimizing queries with expensive predicates*, volume 22. ACM, 1993.

[70] S. Herath, M. Harandi, and F. Porikli. Going deeper into action recognition: A survey. *Image and vision computing*, 60:4–21, 2017.

[71] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[72] W. Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[73] 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.

[74] F. Hueske, M. Peters, A. Krettek, M. Ringwald, K. Tzoumas, V. Markl, and J.-C. Freytag. Peeking into the optimization of data flow programs with mapreduce-style udfs. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 1292–1295. IEEE, 2013.

[75] F. Hueske, M. Peters, M. J. Sax, A. Rheinländer, R. Bergmann, A. Krettek, and K. Tzoumas. Opening the black boxes in data flow optimization. *Proceedings of the VLDB Endowment*, 5(11):1256–1267, 2012.

[76] S. Idreos, M. L. Kersten, S. Manegold, et al. Database cracking. In *CIDR*, volume 7, pages 68–78, 2007.

[77] F. Ilievski, P. Szekely, and B. Zhang. Cskg: The commonsense knowledge graph. In *European Semantic Web Conference*, pages 680–696. Springer, 2021.

[78] E. Jahani, M. J. Cafarella, and C. Ré. Automatic optimization for mapreduce programs. *arXiv preprint arXiv:1104.3217*, 2011.

[79] M. Jarke and J. Koch. Query optimization in database systems. *ACM Computing surveys (CsUR)*, 16(2):111–152, 1984.

[80] 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.

[81] G. Jocher. Yolov5 by ultralytics, 2020.

[82] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015.

[83] J. Joyce. Bayes' theorem. 2003.

[84] D. Kang, P. Bailis, and M. Zaharia. Blazeit: optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv preprint arXiv:1805.01046*, 2018.

[85] 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.

[86] D. Kang, J. Guibas, P. Bailis, T. Hashimoto, and M. Zaharia. Task-agnostic indexes for deep learning-based queries over unstructured data. *arXiv preprint arXiv:2009.04540*, 2020.

[87] D. Kang, A. Mathur, T. Veeramacheneni, P. Bailis, and M. Zaharia. Jointly optimizing preprocessing and inference for dnn-based visual analytics. *arXiv preprint arXiv:2007.13005*, 2020.

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

[89] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[90] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[91] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper. Learned cardinalities: Estimating correlated joins with deep learning. *arXiv preprint arXiv:1809.00677*, 2018.

[92] T. Ko. A survey on behavior analysis in video surveillance for homeland security applications. In *2008 37th IEEE Applied Imagery Pattern Recognition Workshop*, pages 1–8. IEEE, 2008.

[93] V. Koltchinskii and K. Lounici. Concentration inequalities and moment bounds for sample covariance operators. *Bernoulli*, 23(1):110–133, 2017.

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

[95] M. Koziarski and B. Cyganek. Impact of low resolution on image recognition with deep neural networks: An experimental study. *International Journal of Applied Mathematics and Computer Science*, 28(4):735–744, 2018.

[96] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, pages 489–504. ACM, 2018.

[97] S. Krishnan, Z. Yang, K. Goldberg, J. Hellerstein, and I. Stoica. Learning to optimize join queries with deep reinforcement learning. *arXiv preprint arXiv:1808.03196*, 2018.

[98] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[99] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011.

[100] T. C. Kuo and A. L. Chen. Content-based query processing for video databases. *IEEE Transactions on Multimedia*, 2(1):1–13, 2000.

[101] M. La Cascia and E. Ardizzone. Jacob: Just a content-based query system for video databases. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 2, pages 1216–1219. IEEE, 1996.

[102] S. Lakshmi and S. Zhou. Selectivity estimation in extensible databases-a neural network approach. In *VLDB*, pages 623–627, 1998.

[103] L. Le Cam. The central limit theorem around 1935. *Statistical science*, pages 78–91, 1986.

[104] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[105] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[106] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *22nd International conference on data engineering (ICDE'06)*, pages 25–25. IEEE, 2006.

[107] Y. Leng, C.-C. Chen, Q. Sun, J. Huang, and Y. Zhu. Energy-efficient video processing for virtual reality. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 91–103, 2019.

[108] J. Li, S. Cheng, Z. Cai, J. Yu, C. Wang, and Y. Li. Approximate holistic aggregation in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 13(2):1–24, 2017.

[109] J. Li, L. Niu, and L. Zhang. From representation to reasoning: Towards both evidence and commonsense reasoning for video question-answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21273–21282, 2022.

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

[111] Y. Li, A. Torralba, A. Anandkumar, D. Fox, and A. Garg. Causal discovery in physical systems from videos. *Advances in Neural Information Processing Systems*, 33:9180–9192, 2020.

[112] 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 *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[113] R. Liu, T. Wu, and B. Mozafari. A bandit approach to maximum inner product search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4376–4383, 2019.

[114] 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.

[115] 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. ACM, 2018.

[116] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. *ACM SIGMOD Record*, 27(2):426–435, 1998.

[117] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. *ACM SIGMOD Record*, 28(2):251–262, 1999.

[118] T. Marcoux, N. Agarwal, R. Erol, A. Obadimu, and M. N. Hussain. Analyzing cyber influence campaigns on youtube using youtubetracker. *Big Data and Social Media Analytics: Trending Applications*, pages 101–111, 2021.

[119] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. *ACM SIGMOD Record*, 51(1):6–13, 2022.

[120] K. Marino, R. Salakhutdinov, and A. Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016.

[121] W. D. Maurer and T. G. Lewis. Hash table methods. *ACM Computing Surveys (CSUR)*, 7(1):5–19, 1975.

[122] A. Mazumdar, B. Haynes, M. Balazinska, L. Ceze, A. Cheung, and M. Oskin. Vignette: Perceptual compression for video storage and processing systems. *arXiv preprint arXiv:1902.01372*, 2019.

[123] F. Merlevède, M. Peligrad, and E. Rio. Bernstein inequality and moderate deviations under strong mixing conditions. In *High dimensional probability V: the Luminy volume*, pages 273–292. Institute of Mathematical Statistics, 2009.

[124] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2630–2640, 2019.

[125] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[126] R. Mistry and S. Misner. *Introducing Microsoft SQL Server 2014*. Microsoft Press, 2014.

[127] V. Mnih, C. Szepesvári, and J.-Y. Audibert. Empirical bernstein stopping. In *Proceedings of the 25th international conference on Machine learning*, pages 672–679, 2008.

[128] B. Momjian. *PostgreSQL: introduction and concepts*, volume 192. Addison-Wesley New York, 2001.

[129] A. MySQL. Mysql, 2001.

[130] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao, et al. A variegated look at 5g in the wild: performance, power, and qoe implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 610–625, 2021.

[131] W. Nicholson et al. On the normal approximation to the hypergeometric distribution. *The annals of mathematical statistics*, 27(2):471–483, 1956.

[132] F. Olken. *Random sampling from databases*. PhD thesis, University of California, Berkeley, 1993.

[133] OpenAI. Gpt-4 technical report, 2023.

[134] OpenAI. Gpt-4v(ision) system card. 2023.

[135] J. Ortiz, M. Balazinska, J. Gehrke, and S. S. Keerthi. Learning state representations for query optimization with deep reinforcement learning. *arXiv preprint arXiv:1803.08604*, 2018.

[136] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[137] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7753–7762, 2019.

[138] J. Pearl et al. Models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 19(2):3, 2000.

[139] J. Pearl and D. Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

[140] 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.

[141] M. Prosperi, Y. Guo, M. Sperrin, J. S. Koopman, J. S. Min, X. He, S. Rich, M. Wang, I. E. Buchan, and J. Bian. Causal inference and counterfactual prediction in machine learning for actionable healthcare. *Nature Machine Intelligence*, 2(7):369–375, 2020.

[142] X. Qi, Q. Yang, D. T. Nguyen, G. Zhou, and G. Peng. Lbvc: towards low-bandwidth video chat on smartphones. In *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 1–12, 2015.

[143] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[144] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[145] Q. M. Rajpoot and C. D. Jensen. Video surveillance: Privacy issues and legal compliance. In *Promoting Social Change and Democracy Through Information Technology*, pages 69–92. IGI global, 2015.

[146] K. Ramachandra, K. Park, K. V. Emani, A. Halverson, C. Galindo-Legaria, and C. Cunningham. Froid: Optimization of imperative programs in a relational database. *Proceedings of the VLDB Endowment*, 11(4):432–444, 2017.

[147] J. Redmon. Darknet: Open source neural networks in c, 2013.

[148] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[149] A. Rheinländer, A. Heise, F. Hueske, U. Leser, and F. Naumann. Sofa: An extensible logical optimizer for udf-heavy data flows. *Information Systems*, 52:96–125, 2015.

[150] M. Riedewald, D. Agrawal, et al. pcube: Update-efficient online aggregation with progressive feedback and error bounds. In *Proceedings. 12th International Conference on Scientific and Statistica Database Management*, pages 95–108. IEEE, 2000.

[151] I. Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[152] M. A. Roth and S. J. Van Horn. Database compression. *ACM Sigmod Record*, 22(3):31–39, 1993.

[153] D. B. Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

[154] M. Rudelson and R. Vershynin. Hanson-wright inequality and sub-gaussian concentration. *Electronic Communications in Probability*, 18:1–9, 2013.

[155] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[156] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. 1998.

[157] A. Santos, A. Bessa, F. Chirigati, C. Musco, and J. Freire. Correlation sketches for approximate join-correlation queries. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1531–1544, 2021.

[158] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[159] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[160] J. Shao, H. T. Shen, and X. Zhou. Challenges and techniques for effective and efficient similarity search in large video databases. *Proceedings of the VLDB Endowment*, 1(2):1598–1603, 2008.

[161] A. Sharma and E. Kiciman. Dowhy: An end-to-end library for causal inference. *arXiv preprint arXiv:2011.04216*, 2020.

[162] X. Shi, Z. Pan, and W. Miao. Data integration in causal inference. *Wiley Interdisciplinary Reviews: Computational Statistics*, 15(1):e1581, 2023.

[163] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 239–249, 2004.

[164] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[165] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237. Springer, 2002.

[166] R. Speer, J. Chin, and C. Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[167] T. A. Stephenson. An introduction to bayesian network theory and usage. Technical report, Idiap, 2000.

[168] M. Stonebraker and L. A. Rowe. The design of postgres. *ACM Sigmod Record*, 15(2):340–355, 1986.

[169] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[170] M. Szummer and R. W. Picard. Indoor-outdoor image classification. In *Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database*, pages 42–51. IEEE, 1998.

[171] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[172] N. Tandon, A. S. Varde, and G. de Melo. Commonsense knowledge in machine intelligence. *ACM SIGMOD Record*, 46(4):49–52, 2018.

[173] Y. Tang, D. Ding, Y. Rao, Y. Zheng, D. Zhang, L. Zhao, J. Lu, and J. Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1207–1216, 2019.

[174] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640, 2016.

[175] B. Thuraisingham. Privacy constraint processing in a privacy-enhanced database management system. *Data & Knowledge Engineering*, 55(2):159–188, 2005.

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

[177] I. Trummer, S. Moseley, D. Maram, S. Jo, and J. Antonakakis. Skinnerdb: regret-bounded query evaluation via reinforcement learning. *Proceedings of the VLDB Endowment*, 11(12):2074–2077, 2018.

[178] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the energy efficiency of a database server. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 231–242, 2010.

[179] V. Veitch, D. Sridhar, and D. Blei. Adapting text embeddings for causal inference. In *Conference on Uncertainty in Artificial Intelligence*, pages 919–928. PMLR, 2020.

[180] P. Voigt and A. Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

[181] D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[182] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[183] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[184] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 2020.

[185] T. White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[186] M. Wloka. Batch, batch, batch: What does it really mean. In *Presentation at game developers conference*, 2003.

[187] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics.

[188] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[189] Y.-S. Xu, T.-J. Fu, H.-K. Yang, and C.-Y. Lee. Dynamic video segmentation network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6556–6565, 2018.

[190] Z. Xu, G. T. Kakkar, J. Arulraj, and U. Ramachandran. Eva: A symbolic approach to accelerating exploratory video analytics with materialized views. In *Proceedings of the 2022 International Conference on Management of Data*, pages 602–616, 2022.

[191] T. Yamada, S. Gohshi, and I. Echizen. Use of invisible noise signals to prevent privacy invasion through face recognition from camera images. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1315–1316, 2012.

[192] A. Yang, A. Nagrani, P. H. Seo, A. Miech, J. Pont-Tuset, I. Laptev, J. Sivic, and C. Schmid. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. *arXiv preprint arXiv:2302.14115*, 2023.

[193] Y. Yang, Y. Li, C. Fermuller, and Y. Aloimonos. Robot learning manipulation action plans by" watching" unconstrained videos from the world wide web. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[194] K. Yi, C. Gan, Y. Li, P. Kohli, J. Wu, A. Torralba, and J. B. Tenenbaum. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

[195] B. Youngmann, M. Cafarella, Y. Moskovitch, and B. Salimi. On explaining confounding bias. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 1846–1859. IEEE, 2023.

[196] B. Youngmann, M. Cafarella, B. Salimi, and A. Zeng. Causal data integration. *arXiv preprint arXiv:2305.08741*, 2023.

[197] J. Yu, H. Han, H. Zhu, Y. Chen, J. Yang, Y. Zhu, G. Xue, and M. Li. Sensing human-screen interaction for energy-efficient frame rate adaptation on smartphones. *IEEE Transactions on Mobile Computing*, 14(8):1698–1711, 2014.

[198] Y. Yue, S. A. Hong, and C. Guestrin. Hierarchical exploration for accelerating contextual bandits. *arXiv preprint arXiv:1206.6454*, 2012.

[199] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, et al. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[200] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6720–6731, 2019.

[201] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*, 2018.

[202] J. Zeng, M. F. Gensheimer, D. L. Rubin, S. Athey, and R. D. Shachter. Uncovering interpretable potential confounders in electronic medical records. *Nature Communications*, 13(1):1014, 2022.

[203] K. Zeng, S. Gao, B. Mozafari, and C. Zaniolo. The analytical bootstrap: a new method for fast error estimation in approximate query processing. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 277–288, 2014.

[204] 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.

[205] H.-B. Zhang, Y.-X. Zhang, B. Zhong, Q. Lei, L. Yang, J.-X. Du, and D.-S. Chen. A comprehensive survey of vision-based human action recognition methods. *Sensors*, 19(5):1005, 2019.

[206] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[207] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. 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*, pages 19–27, 2015.

[208] J. Zou, Q. Zhao, W. Yang, and F. Wang. Occupancy detection in the office by analyzing surveillance videos and its application to building energy conservation. *Energy and Buildings*, 152:385–398, 2017.

[209] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023.