Journal of Internet Services and Applications, 2025, 16:1, doi: 10.5753/jisa.2025.5495

Querying large video datasets: a systematic literature review

Clayton Kossoski [**Universidade Tecnológica Federal do Paraná** | ***claytonk@alunos.utfpr.edu.br***]
Heitor Silvério Lopes [**Universidade Tecnológica Federal do Paraná** | ***hslopes@utfpr.edu.br***]
Jean Marcelo Simão [**Universidade Tecnológica Federal do Paraná** | ***jeansimao@utfpr.edu.br***]

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná, Av. Sete de Setembro, 3165 , Curitiba - PR, 80230-901, Brazil.

Received: 30 January 2025 • **Accepted:** 28 July 2025 • **Published:** 02 October 2025

Abstract Querying large-scale video datasets differs from querying short videos due to the inherent challenges in volume, velocity, and variety. In the last decade, this area has emerged thanks to the effectiveness of deep learning methods, new graphics processing units, new video databases, advances in distributed computing, among others. The main goal of querying video streams is to find the best balance between available hardware, software resources, and query latency, taking into account quality goals, constraints, and video configurations. Due to these challenges, many development methods, frameworks, and evaluation metrics have been proposed. As a result, this systematic literature review addresses a gap in the current body of knowledge. It covers ten years, from 2014 to 2024, and 4,248 papers, of which 99 were identified as relevant and used to answer the research questions on (i) processing methods, hardware architecture, and software, (ii) query languages, (iii) evaluation metrics, (iv) and available datasets. In addition, this review shows how this niche is promising and concerned with the rational use of available resources. Among the results, the following are highlighted: cheap detection models are very popular, smart IoT devices are very useful, distributed computing for video query applications is complex, system latency is essential, and there is no standard video query language. Current trends include the development of a standard video query language, in-memory computing, processing where data is produced, low-latency processing, and active learning for labeling objects. This original work shows a domain perspective, identifies problems and opportunities, and provides directions for future studies.

Keywords: Video Query Processing Methods, Video Query Languages, Datasets.

1 Introduction

Querying videos has long been a goal within the computer vision community [Ogle and Stonebraker, 1995; Li *et al*., 1997]. This aspiration has been largely influenced by the database community, which has relied on the versatile tools provided by Structured Query Language (SQL). However, traditional query languages like SQL are designed for structured data, whereas video content is inherently unstructured and frame-based. As a result, processing unstructured data presents a complex challenge, as it requires sophisticated hardware and software algorithms to analyze videos and extract high-level information from raw data, such as relevant details about people, objects, events, and actions [Kang *et al*., 2017, 2019a; Kossoski, 2024].

Recently, the task of querying videos in large-scale datasets or streaming environments has received growing attention, driven by its diverse range of applications. Specifically, Complex Event Detection (CED) involves several challenges, including object detection, tracking, identifying spatiotemporal relationships, and event matching. These tasks are further complicated by factors such as environmental changes, obstructions, and tracking errors [Honarparvar *et al*., 2024].

For instance, in smart cities, traffic must be continuously monitored [Lu *et al*., 2015; Stonebraker *et al*., 2020]. In this context, traffic authorities may need systems that support direct queries to detect congestion, identify vehicles by attributes (e.g., license plate, color), or track coordinated movements such as criminal convoys [Stonebraker *et al*., 2020; Hsieh *et al*., 2018; Hsieh, 2019; Kang *et al*., 2019a].

Due to the high volume, speed and variety of data produced by large scale video querying/retrieval systems, these can be considered big data applications [Zhang *et al*., 2017; Lu *et al*., 2016; Alam *et al*., 2020a]. Unlike traditional data, the term Big Data encompasses large and heterogeneous datasets that demand advanced algorithms and high-performance infrastructure.

Thus, traditional processing and analysis tools can no longer be efficient in the case of big data applications [Oussous *et al*., 2018]. For instance, most data scientists and experts define Big Data by the following three main characteristics (called 3Vs) [Furht and Villanustre, 2016; Oussous *et al*., 2018]:

- Volume: It refers to large volumes of digital data generated continuously from various devices and applications (e.g. IP cameras, smartphones, social networks, sensors, records, etc.)
- Velocity: refers to the rapid generation of data that necessitates swift processing in order to derive meaningful information and relevant insights.
- Variety: refers to the diverse nature of data generated from multiple distributed sources and in various formats, such as videos, documents, comments, and logs. Large datasets typically encompass both structured and

unstructured data, which may be public or private, local or remote, shared or confidential, and complete or incomplete, among other variations.

Fortunately, in the last ten years at least, there has been a lot of progress in the development of software and hardware that is making it possible to work with large masses of video. Particularly, computer vision has advanced significantly due to deep learning [Goodfellow *et al*., 2016; Rosebroke, 2017; Chollet, 2021], detection models [Farhadi and Redmon, 2018; Bochkovskiy *et al*., 2020], larger datasets [Kang *et al*., 2017; Wen *et al*., 2020; Jodoin *et al*., 2014; MOT2016, 2022; Xipg, 2021; Ristani *et al*., 2016; Kossoski, 2024], graphical processing libraries[1], video description [Aafaq *et al*., 2019], development frameworks [Pouyanfar *et al*., 2018a; Dong *et al*., 2021], and improved hardware acceleration (e.g., GPUs).

With regard to advances in hardware, it is worth pointing out the distributed processing of large volumes of data and, in particular, the use of Graphical Processing Unit (GPU) to speed up model training and object detection in big data context [Pouyanfar *et al*., 2018b; Alam *et al*., 2020b].

However, these advances are insufficient to address key issues in video querying, notably *computational cost and latency* [Kang *et al*., 2017; Yi *et al*., 2017; Hsieh *et al*., 2018; Kang *et al*., 2019a; Yadav *et al*., 2020; Hwang *et al*., 2022].

In these literature, computational cost refers to the processing time of a video querying application and latency refers to the time required between starting a query and retrieving it when the query matches with an event in video. Both these concepts are directly related to the tools and processing steps used in the processing pipeline of a video querying application. It is particularly relevant because video querying at scale is much more challenging than a standalone application on a single node. The complexity involves several hardware, software, network connections, databases, and high-level applications.

Many researchers have studied different aspects of this area and proposed approaches to object detection, data ingestion, content-based video retrieval, video databases, processing pipelines, and query languages. To the best of our knowledge, there is no systematic review that consolidates state-of-the-art approaches to video query systems.

Precisely, this paper aims to fill this gap and present a novel perspective on the domain, including a broad overview of the relevant methods, frameworks, models, query languages, video databases, and evaluation metrics. Therefore, with the aim of providing valuable content to the scientific community, this paper conducted a Systematic Literature Review (SLR) of 99 studies published between 2014 and 2024 and presents the following contributions:

- It highlights relevant approaches for querying videos through both software and hardware solutions, with illustrations that can help readers gain a better understanding;
- A comparison of existing video query languages, including the types of supported operators;
- A list of common or available video datasets for benchmark;
- A summary of issues and trends in the query video and direction for future research.

The remainder of this paper is organized as follows: Section 2 provides background information on computer vision and video codecs. Section 3 presents the related work. Section 4 presents the method and organization of the review following a SLR. Sections 5, 6, 7, and 8 treat each research question formulated in the SLR with descriptions and the answers found in relevant papers. Finally, Section 9 presents the conclusion, trends, and challenges.

2 Background

This section provides a brief overview of computer vision and video codecs, as these topics are commonly addressed in the reviewed literature.

2.1 Computer Vision

Computer Vision (CV) is a sub-field of computer engineering focused on enabling computers to interpret and understand images and videos. In academic literature, CV generally refers to the research and development of algorithms and models designed to automate tasks that require visual perception. These tasks encompass image recognition, object detection, facial recognition, image generation, and more. Computer vision techniques are widely applied in areas such as image and video analysis, autonomous vehicles, medical image analysis, and augmented reality [Rosebrock, 2016].

For a computer, images are represented simply as large numeric matrices. In contrast, humans interpret images in terms of distinct components and meaningful concepts. This discrepancy is known as the semantic gap, which refers to the difference between human perception of an image's contents and the way those contents are represented for computer understanding [da Silveira, 2023; Rosebroke, 2017]. While image algorithms process low-level data, such as pixels and colors, humans comprehend higher-level data, such as objects and events. The challenge lies in bridging the gap between basic visual elements and the meaningful concepts that humans perceive [Alam *et al*., 2020b].

Significant advancements have been made in computer vision (CV), as well as in other fields such as natural language processing and speech recognition, largely due to major improvements in Artificial Intelligence (AI) research. These improvements are driven by increased computing power and the availability of large datasets for training detection models. Two important and related areas of AI are Deep Learning (DL) and Machine Learning (ML).

While DL is a subset of Neural Networks (NN), which in turn is a subfield of ML, DL distinguishes itself from ML by its ability to automatically extract features from unstructured data. This capability reduces the need for human intervention, which is a common requirement in traditional ML methods. Neural networks, fundamental to DL, consist of layers of interconnected nodes, each with weights and thresholds. The term "deep" refers to the multiple layers within

[1] https://docs.opencv.org/4.x/index.html

these networks, which allow for more complex processing and representation of data [Rosebroke, 2017; Chollet, 2021].

In particular, Deep Learning architectures typically involve neural networks with multiple stacked layers, each containing one or more hidden layers [Nielsen, 2015; Chollet, 2021; Goodfellow *et al*., 2016]. This approach enables the network to extract features at increasingly finer granularities. However, it also results in higher processing costs. These layered or hierarchical representations form a mathematical framework for learning from data, where each successive layer captures progressively more meaningful representations of the data. The term deep in deep learning reflects this concept of hierarchical, layer-by-layer abstraction. Deep Learning has led to significant breakthroughs in several traditionally challenging areas of computing, such as image classification at near-human levels, speech recognition, handwriting transcription, autonomous driving, advanced machine translation, text-to-speech synthesis, and web search optimization, among others.

Convolutional Neural Networks (CNNs), a class of deep learning models, have become the state-of-the-art approach for a wide range of computer vision tasks, including object detection and classification [Krizhevsky *et al*., 2012; Bochkovskiy *et al*., 2020; Sandler *et al*., 2018; He *et al*., 2017, 2016]. In CNNs, hidden layers perform convolution operations—such as multiplications or other scalar products—typically followed by a Rectified Linear Unit (ReLU) activation function [Nair and Hinton, 2010]. These convolutional layers are often complemented by additional components, including pooling layers, fully connected layers, and normalization layers, to enhance feature extraction and learning capacity [Goodfellow *et al*., 2016].

2.2 Video capture CODECs

A CODEC is a software, hardware, or combined solution that can digitize, compress, and decompress audio or video signals. It converts raw data into digital form, allowing it to be transmitted, received, stored, and compressed to reduce storage space, increase transmission bit rate, or both [Punchihewa and Bailey, 2020; Sun *et al*., 2024]. According to the author, some of the most common video encoding standards include:

- H.264/AVC (Advanced Video Coding): One of the most widely used video compression standards, H.264 offers high-quality compression and is supported by a broad range of devices and platforms.
- H.265/HEVC (High-Efficiency Video Coding): A newer standard that provides greater compression efficiency compared to H.264/AVC. It offers the same video quality at smaller file sizes, maintaining the same bit rate.
- VP9: An open-source video compression standard developed by Google, VP9 offers compression efficiency similar to H.265/HEVC and is primarily used for streaming applications like YouTube.
- AV1: Another open-source video compression standard developed by the Alliance for Open Media, AV1 provides even better compression efficiency than VP9 and H.265/HEVC. Due to its advantages, AV1 is expected to see widespread adoption for streaming video.
- MPEG-2: An older video compression standard still used in some applications, such as DVDs and broadcast television. While it provides good quality, MPEG-2 is less efficient than more recent standards like H.264/AVC and H.265/HEVC.

3 Related work

The present Systematic Literature Review (SLR) differs from other areas of research, such as video description [Aafaq *et al*., 2019], content-based video retrieval [Spolaôr *et al*., 2020; Latif *et al*., 2019], action recognition [Tran *et al*., 2018; Inacio *et al*., 2021; Gutoski *et al*., 2021], and cross-modal retrieval [Zhen *et al*., 2019]. These studies, however, do not specifically address the video events query processing, reduction of computing costs and latency in large video datasets. Nonetheless, there are related aspects in the existing literature, and in the following, we contrast the similarities and differences between this body of work and the present study.

The authors in Olatunji and Cheng [2019] conducted a review of video analytics methods and techniques applied to video surveillance, providing an in-depth discussion of the theoretical foundations behind the current state-of-the-art approaches. In contrast, our analysis examines video analytics from a different perspective, encompassing common datasets used for training and testing machine learning models, video query languages, and various hardware architectures.

In Zhang *et al*. [2019], the authors reviewed the applications, algorithms, and platforms for edge video analytics, specifically within the context of public safety, with a focus on both domain-specific and general-purpose platforms. In comparison, our review offers a broader overview of diverse hardware computing architectures for video processing, including the edge-first computing architecture.

The authors in Usman *et al*. [2019] analyzed various machine learning platforms that can process and manage multimedia data generated by different smart city applications, exploring the concept of smart cities within the context of Big Data. In contrast, while our review also addresses applications in smart cities and Big Data environments, it specifically concentrates on video query approaches and associated tasks.

The authors in Xu *et al*. [2023] examine the fundamentals of edge computing and provide an overview of video analytics. In comparison, our research offers a comprehensive overview of video analytics, with a focus on three distinct hardware architectures: server-driven, edge computing, and fully distributed systems.

4 Method and organization of the review

This section presents a systematic literature review (SLR) covering the period from 2014 to 2024, adhering to the guidelines established by Pagani *et al*. [2015] for the portfolio se-

lection methodology known as ”Methodi Ordinatio.” This methodology employs an equation referred to as the Index Ordinatio (or InOrdinatio) to classify papers based on their scientific relevance, considering factors such as the journal's impact factor, citation count, and year of publication. As an example, the Methodi Ordinatio framework divides the SLR process into nine phases, presented in Figure 1. The first five phases focus on the selection of a bibliographic portfolio, while the final four phases are dedicated to classifying publications, acquiring papers, and performing a systematic analysis.

In phase 1, we defined the research questions based on the objectives of our SLR: to summarize and clarify the primary approaches, video querying languages, evaluation metrics, datasets, and research gaps within the field.

In phases 2, 3, and 4, we developed a search strategy to identify relevant papers addressing our research questions. This involved defining a search query and selecting target databases. As a result, a set of candidate studies was retrieved from the query search in each database.

Subsequently, in phases 5, 6, and 7, these candidate studies were filtered according to predefined inclusion and exclusion criteria. During this process, the InOrdinatio equation was applied to rank the papers based on their scientific relevance.

Finally, phases 8 and 9 focused on acquiring the full papers, reading them, and performing a systematic analysis of their contents. Each of these phases is described in further detail below.

Phase 1: Establishing the purpose of the research

The aim of this systematic literature review is to identify and examine existing approaches for querying large video databases. To guide this review, four research questions (RQ) have been defined:

- RQ1: What models, methods, and frameworks are available for video querying?
- RQ2: What video query languages exist, and what operations do they support?
- RQ3: How is the quality of existing approaches assessed, both quantitatively and qualitatively?
- RQ4: What datasets are available for training and evaluating models and frameworks?

Phase 2 – Preliminary exploratory search of keywords in databases

To identify key repositories related to video querying, machine learning, big data, and computer vision, a search was conducted on Google Scholar[2] using the initial search term ”querying large video datasets.” The following databases were selected for the search: ACM Digital Library[3], IEEE Xplore[4], SpringerLink[5], and ScienceDirect[6]. Arxiv[7] and similar preprint repositories were deliberately excluded from this SLR, as they do not undergo peer review.

[2]https://scholar.google.com/
[3]https://dl.acm.org/
[4]https://ieeexplore.ieee.org/Xplore/home.jsp
[5]https://link.springer.com/
[6]https://www.sciencedirect.com/
[7]https://arxiv.org/

Phase 3 – Definition and combination of keywords and databases

The following keywords were defined for searching the selected databases: “querying video stream”, “query video”, “search events in video”, “query video database”, and “query video language”. The search was restricted to the period from 01/01/2014 to 31/12/2024, with the aim of achieving a comprehensive coverage of relevant papers spanning nearly a decade. We focused on the past decade, as the rise of deep learning since 2012 has significantly advanced video processing [Krizhevsky *et al*., 2012].

Phase 4 – Search in the databases

The broad search returned a total of 4,248 results, distributed as follows: IEEE Xplore (313), ACM Digital Library (362), SpringerLink (3,320), and ScienceDirect (253).

Phase 5 – Filtering procedures

All the papers retrieved in the broad search were carefully checked and filtered. The Mendeley reference manager tool was used to load and remove duplicates. Subsequently, each remaining paper was quickly assessed for relevance by reviewing its title, abstract, and keywords. Only those papers directly related to the domain were retained to form the initial portfolio. As a result of this filtering process, many papers were excluded, with the aim of retaining only the most pertinent studies. Ultimately, a total of 99 papers remained.

Phase 6 – Collecting information about paper's impact

In addition to the year of publication, each selected paper was also evaluated on the basis of its impact factor and the number of citations. These widely recognized metrics are indicative of the scientific relevance of a paper published in scholarly journals. Pagani *et al*. [2015] proposed the InOrdinatio equation (1) to rank the papers.

In this equation, IF represents the impact factor or another impact index of journals, which is divided by 1000 to normalize its value, α is a weighting factor ranging from 1 to 10, assigned by the researcher; $ResearchYear$ refers to the year the research was conducted; $PublishYear$ is the year the paper was published; and Ci denotes the number of citations the paper has received. As suggested by Pagani *et al*. [2015], the value of α should be set based on the user's experience with the topic. To maintain objectivity and avoid bias, we set $\alpha = 5$ for all the papers under review.

$$InOrdinatio = (IF/1000) + \alpha * [10 - (ResearchYear - PublicationYear)] + \sum Ci \quad (1)$$

Among the results obtained in Phase 5, many papers were published in conferences or as book chapters and, therefore, did not have an Impact Factor. According to Pagani *et al*. [2015], when the Impact Factor is unavailable, the researcher may choose an alternative metric. In this case, we decided to include these publications due to their academic relevance and high citation count. Consequently, we used the H-index Hirsch and Buela-Casal [2014] of the first author, as it is readily available via Google Scholar, in place of the Impact Factor.

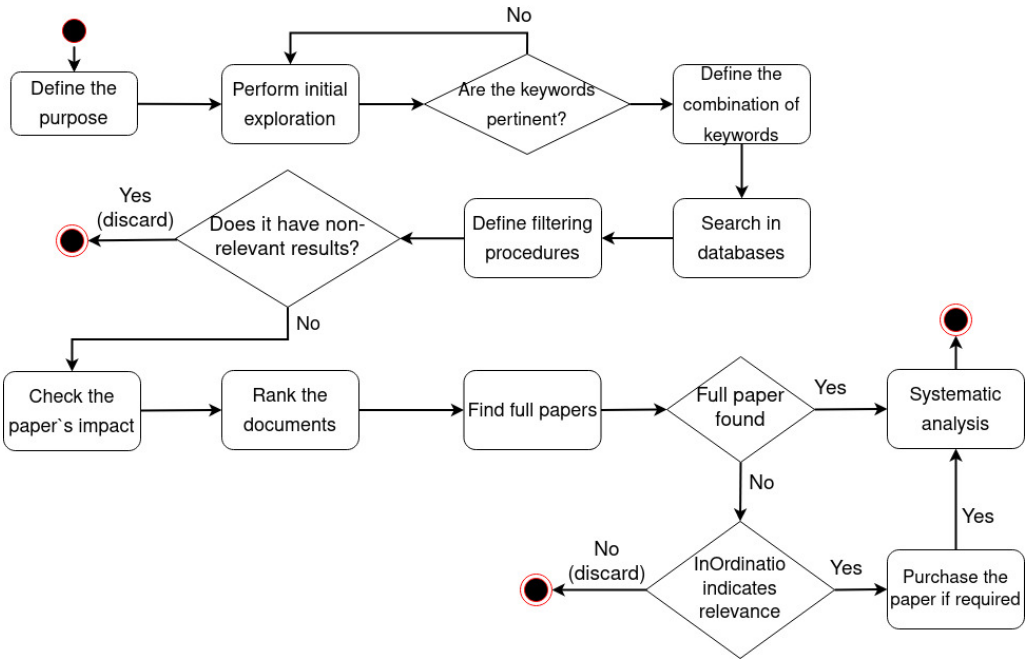

Figure 1. Phases of the review

Phase 7 – Ranking the papers using the InOrdinatio

Table 1 presents the ranking of the papers according to the InOrdinatio approach.

Phase 8 – Retrieving the full papers

According to Pagani *et al*. [2015], the researcher determines the number of papers to be read based on their InOrdinatio rank. In this study, all remaining papers were thoroughly read and included in the composition of this review.

Phase 9 – Final reading and systematic analysis of the papers

In the following four sections, the research questions formulated in phase 1 are addressed based on the analysis of the papers surveyed.

5 RQ1: Which models, methods, frameworks, and systems exist for video querying?

This section presents the main architectures and software components used in video query systems.

Subsection 5.1 discusses the main approaches for handling large-scale video querying applications focusing on hardware configuration. Subsection 5.2 describes the implementation of a complete video query pipeline, encompassing the stages of ingestion, storage, retrieval, and consumption.

5.1 Overview of the main approaches for handling large-scale video querying applications

The systematic literature review identified hardware configuration as a critical driver behind the development of the reviewed methods and frameworks. The need to query large-scale video datasets demands high-performance computing resources, often relying on distributed architectures. Consequently, virtually all approaches take hardware availability into account as a fundamental aspect of their design.

Hardware configurations refer to the physical infrastructure that interconnects video producers, consumers, and the entire processing pipeline. In general, there are three computing architectures: a) server-driven, computing everything on a powerful server; b) edge-first, computing everything where the data are produced; and c) fully distributed, distributing computing across hierarchical nodes. Each of these approaches has advantages and disadvantages. It is worth noting that many solutions employ hybrid architectures to some degree. Nevertheless, for the sake of clarity and analytical coherence, the hardware architectures are categorized based on the primary problem addressed by the respective authors. Hence, certain methods do not explicitly specify the architecture they employ; consequently, this section focuses on articles where the architecture is clearly defined. To further enhance readability, the name of each solution method or framework (when available) is presented in bold throughout the text.

5.1.1 Server-driven

The most common hardware architecture is “server-driven”, also known as client-cloud configuration and cloud computing. These architectures benefit from high computational resources and facilitate the deployment of large and complex models. However, they introduce high network usage, increase transmission latency, and may become bottlenecks under heavy workloads. Due to its simplicity, this architecture is the most widely used over time for many different applications and query processing. Figure 2 shows an overview of the server-driven hardware architecture.

Table 1. Papers in chronological order and the computed InOrdinatio ranking

Paper	Type	H-index	Year of publication	Number of citations	InOrdinatio
Shen *et al*. [2014]	R	20	2014	20	35
Jodoin *et al*. [2014]	R	5	2014	148	163
Lu *et al*. [2015]	A	0	2015	29	49
Chen *et al*. [2015]	C	8	2015	586	606
Lu *et al*. [2016]	S	14	2016	118	143
Han *et al*. [2016]	C	20	2016	476	501
Patino *et al*. [2016]	D	15	2016	122	147
Ristani *et al*. [2016]	D	5	2016	3298	3323
de Boer *et al*. [2017]	C	12	2017	3	33
Kang *et al*. [2017]	C	26	2017	468	498
Yi *et al*. [2017]	S	16	2017	379	409
Zhang *et al*. [2017]	S	9	2017	533	563
Shen *et al*. [2017]	C	18	2017	87	117
Hung *et al*. [2018]	S	9	2018	333	368
Pakha *et al*. [2018]	R	0	2018	57	92
Poms *et al*. [2018a]	A	7	2018	92	127
Jiang *et al*. [2018]	C	35	2018	541	576
Hsieh *et al*. [2018]	S	22	2018	344	379
Ran *et al*. [2018]	R	8	2018	499	534
Wang *et al*. [2018]	C	33	2018	163	198
Grulich and Nawab [2018]	C	16	2018	68	103
Krishnan *et al*. [2018]	C	32	2018	28	53
Haynes *et al*. [2018]	C	12	2018	44	79
Lu *et al*. [2018]	C	14	2018	128	163
Liu *et al*. [2018]	C	26	2018	238	273
Kang *et al*. [2019a]	C	26	2019	161	201
Canel *et al*. [2019]	C	7	2019	188	228
Jain *et al*. [2019]	W	5	2019	98	138
Anderson *et al*. [2019]	R	12	2019	96	136
Xu *et al*. [2019]	C	5	2019	90	130
Xarchakos and Koudas [2019]	C	4	2019	35	75
Yadav [2021]	T	11	2019	3	43
Yadav and Curry [2019b]	C	11	2019	43	83
Zhang and Kumar [2019]	C	7	2019	44	84
Mao *et al*. [2019]	R	17	2019	30	70
Mullapudi *et al*. [2019]	C	9	2019	128	168
Kang *et al*. [2019b]	C	26	2019	50	90
Hsieh [2019]	T	22	2019	8	48
Fu *et al*. [2019]	R	15	2019	61	101
Yadav [2019]	S	11	2019	10	50
Liu *et al*. [2019]	C	10	2019	66	106
Yadav *et al*. [2020]	A	11	2020	12	57
Chao *et al*. [2020]	C	5	2020	25	70
Li *et al*. [2020b]	W	5	2020	2	47
Kraft *et al*. [2020]	C	12	2020	31	76
Kang *et al*. [2020b]	C	26	2020	56	101
Stonebraker *et al*. [2020]	W	0	2020	12	57
Bastani *et al*. [2020]	C	19	2020	100	145
Collins [2020]	M	0	2020	1	46
Sipser [2020]	M	0	2020	3	48
Li *et al*. [2020a]	C	5	2020	262	307
Yang *et al*. [2020]	J	30	2020	96	141
Lai *et al*. [2021]	C	6	2020	8	53
Du *et al*. [2020]	C	6	2020	8	53
Kang *et al*. [2020a]	C	26	2020	37	82
Haynes *et al*. [2020]	C	12	2020	25	70
Laskaridis *et al*. [2020]	C	11	2020	308	353
Wen *et al*. [2020]	D	45	2020	742	787
Ramachandra and Jones [2020]	D	8	2020	191	236
Jain *et al*. [2020]	S	5	2020	112	157
Nguyen-Duc *et al*. [2021]	C	6	2021	6	56
Yadav *et al*. [2021]	J	11	2021	24	74
Haynes *et al*. [2021]	R	12	2021	37	48
Daum *et al*. [2021]	C	0	2021	43	93
Qin *et al*. [2021]	C	6	2021	6	56
Kang *et al*. [2021]	C	26	2021	27	57
Chen *et al*. [2021]	C	6	2021	21	71
Guo *et al*. [2021]	C	8	2021	53	103
Kang *et al*. [2022]	C	26	2022	24	79
Moll *et al*. [2022]	C	7	2022	38	93
Chunduri *et al*. [2022]	C	3	2022	19	74
Yang *et al*. [2022]	C	0	2022	25	80
Romero *et al*. [2022]	C	11	2022	29	84
Koudas *et al*. [2022]	C	68	2022	43	98
Hwang *et al*. [2022]	C	2	2022	13	68
Dai *et al*. [2022]	A	4	2022	31	86
Chen *et al*. [2022]	C	6	2022	9	64
Khani *et al*. [2023]	S	8	2023	60	120
Agarwal and Netravali [2023]	R	5	2023	12	72
Rahmanian *et al*. [2023]	S	11	2023	4	64
Chao *et al*. [2023]	C	5	2023	2	62
Wu *et al*. [2023]	A	4	2023	9	69
Li *et al*. [2023a]	C	5	2023	20	80
Kakkar *et al*. [2023]	C	4	2023	7	67
Lv *et al*. [2023]	A	0	2023	11	71
Yang *et al*. [2023]	C	10	2023	15	75
Wang *et al*. [2023]	C	2	2023	7	67
Zhang *et al*. [2023]	C	16	2023	6	66
Li *et al*. [2023b]	C	5	2023	1	61
HÖnig *et al*. [2023]	C	3	2023	0	60
Kossoski *et al*. [2024]	A	1	2024	1	66
Dai *et al*. [2024]	C	4	2024	8	73
Madden *et al*. [2024]	A	144	2024	2	67
Chaudhary *et al*. [2024]	C	1	2024	1	66
Zhang *et al*. [2024]	C	5	2024	3	68
Rahmanian *et al*. [2024]	C	11	2024	1	66
Sun *et al*. [2024]	C	0	2024	2	67
Wang *et al*. [2024]	C	2	2024	2	67
Wen *et al*. [2024]	A	0	2024	0	65

Legend: A (Article), C (Conference), D (Dataset), M (Masters Thesis), R (Research Paper), S (Symposium), T (Ph.D. Thesis), W (Workshop).

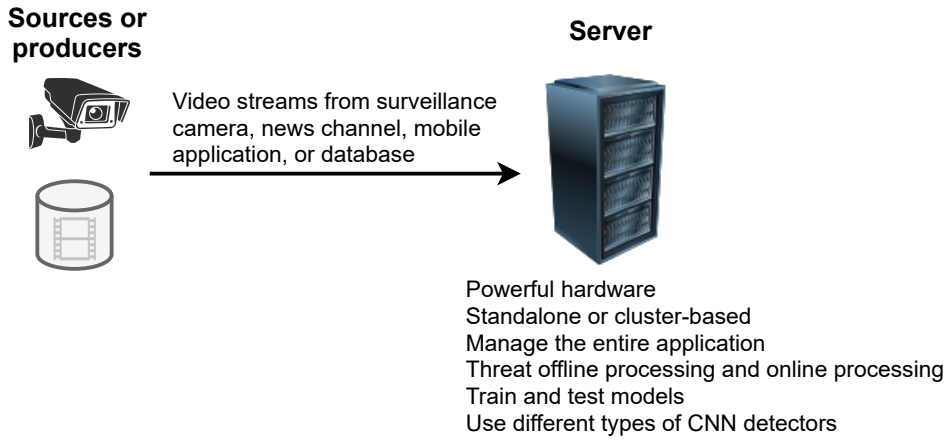

Figure 2. Basic overview of server-driven architecture

The advantages are simplicity, powerful hardware support, and low internal latency. However, the main drawback is related to the cost and quality of the external communication links, particularly if the data travels through the internet connection, and monolithic processing.

Based on a detailed analysis of the problems articulated by the authors, the papers are categorized into thematic groups as follows:

Efficient Large-Scale Video Analytics and Querying

This group addresses the scalability, efficiency, and computational cost associated with processing, analyzing, and querying vast amounts of video data, often from numerous cameras. Problems include managing high data volumes, optimizing deep learning inference for video, and enabling complex spatiotemporal queries.

- Lu *et al*. [2016]: **Optasia** focuses on the critical challenge of efficiently performing large-scale video analytics from numerous cameras, overcoming issues of scalability, efficiency, and error-proneness inherent in existing methods.
- Zhang *et al*. [2017]: **VideoStorm** deals with efficiently analyzing live video streams at scale, specifically addressing the high costs of vision processing and the necessity to effectively manage resources to meet diverse quality and latency goals.
- Kang *et al*. [2017]: The authors of **Noscope** identifies the high computational cost and inefficiency inherent in applying DNNs to large-scale video data for analysis, which traditional methods struggle to handle effectively.
- Kang *et al*. [2019a]: **Blazeit** addresses the high computational cost and the complex, imperative programming required for querying large volumes of video data using DNNs, especially concerning aggregation and limit queries that existing approximate filtering methods do not handle efficiently.
- Anderson *et al*. [2019]: The authors of **Tahoma** highlights that querying the content of images and videos using CNNs is computationally expensive and slow, making it prohibitive for massive video libraries. It also notes that existing visual data system optimizations primarily focus on computation and overlook significant data-handling costs like loading and transformation, which drastically impact overall query time.

Optimizing Video Streaming Protocols for AI Analytics

This group concentrates on redesigning or adapting video streaming protocols to better suit the specific requirements of AI, particularly DNNs, rather than human viewing. The problems stem from the inefficiency of traditional source-driven streaming for machine perception, leading to suboptimal accuracy or excessive bandwidth consumption.

- Pakha *et al*. [2018]: The authors of **SimpleProto** tackles the inefficiency of traditional video streaming protocols for distributed vision analytics, which are designed primarily for human viewing quality rather than the specific, distinct needs of DNNs.
- Du *et al*. [2020]: The authors of **DDS** identifies the inefficient video streaming for AI applications, where source-driven protocols lead to high bandwidth consumption and suboptimal DNN inference accuracy due to limited camera-side intelligence and lack of server feedback.

Dynamic Configuration and Resource Management for Video Analytics Pipelines

This group addresses the challenge of dynamically adjusting configurations, managing resources, and optimizing the performance of video analytics pipelines in response to varying workloads, resource availability, and quality-of-service demands.

- Jiang *et al*. [2018]: The authors of **Chameleon** highlights the prohibitive computational resource costs incurred when applying deep convolutional neural networks to video data at scale, necessitating dynamic optimization of pipeline configurations to manage these expenses effectively.

5.1.2 Edge-first

Edge-first architecture, also known as edge computing, client-first, or server-less, end-edge-cloud, is almost the opposite of server-driven architecture. In that sort of architecture, query processing is concentrated on edge devices

rather than advanced cloud servers, even if it uses some network connection between the nodes (Figure 3). Edge-first systems prioritize early data filtering and inference on edge devices, reducing bandwidth usage and enabling faster responses. With the proliferation of cheap IoT devices that allow local query processing, such as IP cameras and sensors, the number of processing algorithms on the end node has increased, following this trending architecture. Basically, the applications run where the data is produced, and data transfer to a cloud server is performed just when the device cannot handle it.

Although processing at the edge is a priority in this architecture, some applications use edge/cloudlet servers, connected a hop away, to process heavier loads, such as DNN models, computer vision tasks, and database operations. Another similar approach called federated learning coordinates mobile devices to train a machine learning model using wireless networks and keeping the training data on local devices [Hsieh, 2019].

The main advantage of the edge-first architecture is that processing is focused where the data is produced, allowing for low response times and saving network bandwidth, especially for latency-sensitive systems such as autonomous vehicles and surveillance. Given the conditions of the scheduler, the edge server processes less data and only complex tasks that are not supported by the edge node.

Despite their advantages, some solutions can have problems, especially those that rely on an internet connection for some tasks. Other disadvantages include limited computing power, low memory availability, simple CPU processing (rarely with GPU), high battery consumption (if battery-powered), and more complexity due to distributed processing.

Based on a detailed analysis of the problems articulated by the authors, the papers are categorized into thematic groups as follows:

Optimizing DNN Execution on Resource-Constrained Mobile/Edge Devices

This group focuses on the fundamental challenge of running computationally and memory-intensive DNNs efficiently on devices with limited resources, particularly for continuous or real-time applications. The core problem often revolves around bridging the gap between high DNN demands and constrained device capabilities, while accounting for dynamic environmental factors.

- Chen *et al*. [2015]: The authors of **Glimpse** addresses the challenge of achieving continuous, real-time object recognition on mobile devices, where persistent high-performance inference is difficult due to computational and power limitations
- Shen *et al*. [2014]: The authors of **MCDNN** concerns running computationally and memory-intensive DNNs efficiently on resource-constrained mobile devices, especially for continuous mobile vision applications, while accounting for varying resources and network conditions.
- Ran *et al*. [2018]: The authors of **DeepDecision** tackles the difficulty of executing computationally intensive deep learning models, particularly for real-time video analytics, on resource-constrained mobile devices that lack the processing power for full on-device execution.

Core DNN Inference Optimization for Edge-Cloud and Mobile Devices:

This group enhances the fundamental performance (latency, throughput, cost, accuracy) of DNN inference, particularly CNNs, within environments where edge devices and the cloud collaborate. Emphasis is placed on mechanisms such as model splitting, early-exit strategies, and data compression.

- Laskaridis *et al*. [2020]: The authors of **SPINN** focuses on the runtime co-optimization of early-exit policies and CNN splitting for dynamic adaptation and robustness. It also introduces a specific packing mechanism to minimize data transfer overhead.
- Grulich and Nawab [2018]: Addresses video processing in IoT and edge contexts by combining Neural Network techniques such as splitting, compression, and differential communication to optimize collaborative inference.
- Hsieh [2019]: **Focus** is a system designed for low-cost, low-latency ML serving over large datasets like videos, utilizing inexpensive CNNs for approximate indexing and more costly CNNs for query accuracy. Also features a geo-distributed ML training system that optimizes communication and eliminates insignificant inter-data center traffic.
- Rahmanian *et al*. [2023]: **RAVAS** is a framework that employed Q-Learning for DNN model selection based on GPU utilization and workload, alongside a resource allocation strategy to mitigate interference during concurrent GPU execution.

Bandwidth Optimization and Intelligent Video Data Filtering:

This group reduced the volume of video data transmitted from the edge to the cloud using smart filtering, compression, and other techniques to alleviate bandwidth bottlenecks and associated costs.

Papers:

- Lv *et al*. [2023]: **FDDIA** leverages feedback from prior frames and inter-frame differences to identify candidate regions, generating smaller images for efficient edge inference, thereby reducing both latency and bandwidth consumption.
- Li *et al*. [2020a]: **Reducto** dynamically adapts filtering decisions directly at the camera using lightweight machine learning and low-level video features, effectively reducing bandwidth usage and backend computation.
- Yadav *et al*. [2021]: **VID-WIN** targets efficient video event processing in IoMT environments. It employs a two-stage windowing approach (edge and cloud) with content-aware optimizations like micro-batch resizing and “eager” and “lazy” filtering to significantly reduce bandwidth requirements.
- Dai *et al*. [2022]: **Respire** reduces spatial-temporal redundancy in industrial video analytics at edge computing nodes by characterizing it with feature descriptors

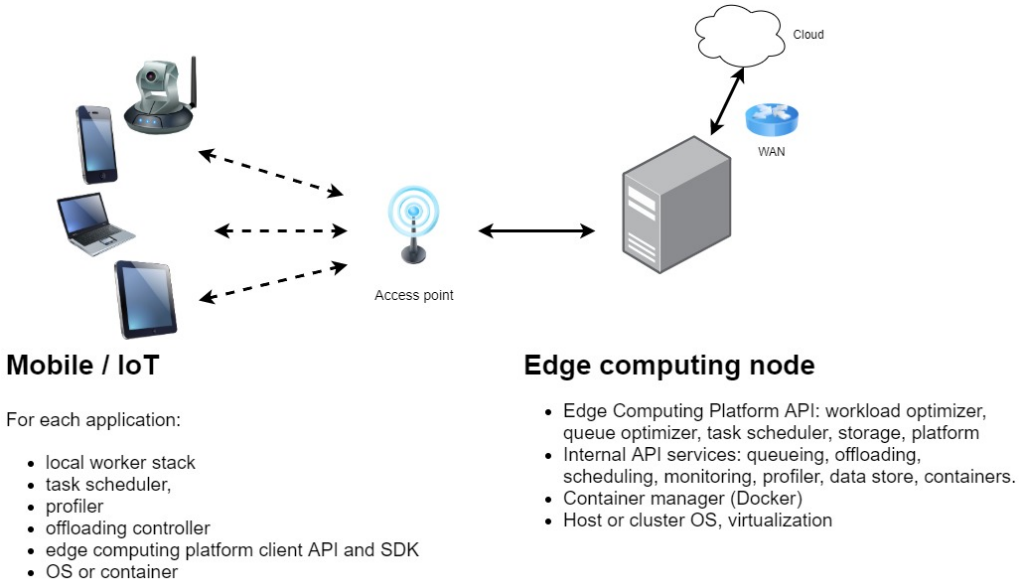

Figure 3. Basic overview of edge-first architecture

and an online heuristic algorithm for efficient frame selection and pruning, aiming to optimize transmission and processing costs.

Multi-Camera Collaboration and Cross-Camera Optimization:

This group leverages information from multiple cameras, particularly those with overlapping Fields of View (FoVs), to eliminate redundancies, optimize inference and tracking, and scale video analytics deployments.

- Dai *et al*. [2024]: **AxionVision** is a framework that utilizes a tiered edge-cloud architecture with continuous online learning and an efficient perspective-aware adaptation method. It also incorporates topology-guided clustering for accelerated model selection in multi-camera systems.
- Yang *et al*. [2023]: **CEVAS** is a framework that utilizes a fine-grained input filtering policy to eliminate spatial and temporal redundancy, an object manager for sharing detection results, and a content-aware model selection policy for multi-view video processing, particularly for vehicular perception.
- Wu *et al*. [2023]: **ILCAS** integrates a cross-camera collaboration scheme that quantifies spatio-temporal correlations between cameras and shares FoV information via motion feature maps.
- Li *et al*. [2023a]: **Polly** is a cross-camera inference system designed to share inference results among co-located cameras with overlapping FoVs, thereby eliminating redundant processing and optimizing resource utilization.
- Guo *et al*. [2021]: **CrossRoi** is a resource-efficient system that defines Regions of Interest (RoI) across a camera network with overlapping coverage to reduce redundant information. It operates with offline phases (data association, RoI calculation) and online phases (stream filtering).
- Jain *et al*. [2020]: **Spatula** is a cost-efficient system that leverages spatial and temporal correlations between cameras to reduce computation and communication overhead by pruning the search space for object tracking across large-scale camera networks.
- Liu *et al*. [2019]: **Caesar** is a hybrid edge computing-based system that intelligently partitions video processing tasks between cameras and an edge cluster to enable near-real-time detection of complex activities spanning multiple cameras.
- Jain *et al*. [2019]: Addresses the challenge of scaling video analytics to large camera deployments by exploiting spatio-temporal correlations among cameras to reduce the inference search space and improve efficiency.
- Rahmanian *et al*. [2024]: **CVF** addresses the challenge of significant computational resource waste that occurs when processing all frames from multiple real-time camera streams on edge devices, especially since only a fraction often contain objects of interest. Highlights the shortcomings of existing on-camera filtration methods, which often lack adaptability to varying workloads, available edge resources, camera numbers, frame prioritization, and resource-aware filtering.
- Wang *et al*. [2024]: **Gecko** is a framework to overcome these issues by: (i) obtaining optimal models from a model zoo and assigning them to edge devices for executing current queries, (ii) optimizing resource usage of the edge cluster at runtime through dynamic adjustment of the frame query interval for each video stream and flexible forking/joining of running models on edge devices, and (iii) improving accuracy in changing video scenes via fine-grained stream transfer and continuous learning of models.

Dynamic Adaptation and Automated Configuration for Video Analytics:

This group developed systems that automatically adapt to varying conditions (network, video content, workload) and

optimize model or analytics pipeline configurations, either through self-learning or without manual intervention.

- Jiang *et al*. [2018]: **Chameleon** is a system that dynamically adapts Neural Network configurations for video analytics. It amortizes profiling costs by exploiting temporal and spatial correlations within the video characteristics, enabling efficient resource utilization in dynamic environments.

5.1.3 Fully distributed

Fully distributed architectures decentralize both inference and data management. Multiple nodes—edge, fog, and cloud—collaborate dynamically to execute query tasks. As expected, this is more complex than server-driven and edge-first architectures because it needs to manage distributed processing, load balancing, device configurations, and time constraints among different machines using network connections. Figure 4 shows the basic overview of the fully distributed architecture. It includes a cloudlet server, usually between the edge devices and the cloud, to reduce overall latency when heavy data needs to be processed using the dynamic partitioning of tasks.

Based on a detailed analysis of the problems articulated by the authors, the papers are categorized into thematic groups as follows:

Web-Scale Visual Content Management

This group argues that traditional search engines face challenges in handling the massive volume of visual content and meeting the computational demands of evolving AI models, which are essential for efficiently and accurately understanding and retrieving images and videos.

- Qin *et al*. [2021]: **Mixer** is a system that efficiently understands and retrieves visual data for a search engine. It uses CNNs for categorization, generates unified feature vectors, and includes a comprehensive DL model production system with optimized execution and resource allocation. It employs approximate nearest neighbor search (ANNS) for image retrieval and a multi-step process for accurate video retrieval.

Edge-Cloud Optimization for Real-time Video Analytics

This group argues that real-time video inference on resource-constrained edge devices is adversely affected by data drift in dynamic environments, leading to performance degradation and reduced model accuracy and adaptability.

- Wang *et al*. [2023]: The authors of **Shoggoth** proposed an edge-cloud collaborative architecture that enhances real-time video inference by addressing data drift through adaptive online learning and decoupled knowledge distillation. It uses a lightweight edge model fine-tuned by a cloud-based teacher model, incorporating adaptive training and frame sampling.
- Ran *et al*. [2018]: **DeepDecision** is a framework that intelligently offloads parts of deep learning processing to powerful backend "helpers" (cloud/edge servers). It uses extensive measurements to understand tradeoffs between factors like video quality, network, battery, delay, and accuracy, and applies mathematical optimization to determine the optimal offloading strategy.

Distributed and Archival Video Analytics

This group argues that executing resource-intensive computer vision queries on edge devices for distributed video streams imposes significant bottlenecks on centralized servers, hindering system scalability and responsiveness. Also, managing and analyzing large archival videos efficiently is challenging due to complex format configurations and conflicting resource demands.

- Nguyen-Duc *et al*. [2021]: Presented an autonomous semantic stream fusion approach using "ASF agents" that push computing operations closer to video sources. It uses semantic-based representation for federated query processing across diverse hardware configuration.
- Xu *et al*. [2019]: **Vstore** is a holistic system for fast and resource-efficient analytics over large archival videos. It uses a "backward derivation" approach and novel techniques to manage the vast space of video format configurations ("knobs"), coalesce stored formats, and erode aging data, optimizing across multiple resource types.

5.2 Processing pipeline

This section presents the core concepts of the video query pipeline, outlining the key stages from data ingestion to consumption. Querying video data at scale introduces significantly greater challenges than managing a standalone application on a single server, primarily due to the integration of heterogeneous hardware components, software systems, network infrastructures, databases, and high-level applications. In response to these complexities, various solutions have been developed over time to support video management in big data contexts.

Currently, the video processing pipeline at scale is divided into four main phases: ingestion, storage, retrieval, and consumption [Xu *et al*., 2019], as depicted in Figure 5. Table 2 provides a glossary of the components involved in the processing pipeline. These phases are further explained in the following sections.

Table 2. Glossary of components in a video query pipeline

Component	Role
Preprocessing	Decoding, sampling, and normalization of video frames
Inference Engine	Object detection, tracking, and semantic extraction
Indexer	Organizes metadata and embeddings for fast retrieval
Query Parser	Translates declarative query into executable tasks
Scheduler	Allocates resources and manages parallelism
Result Filter	Applies confidence thresholds and ranks outputs

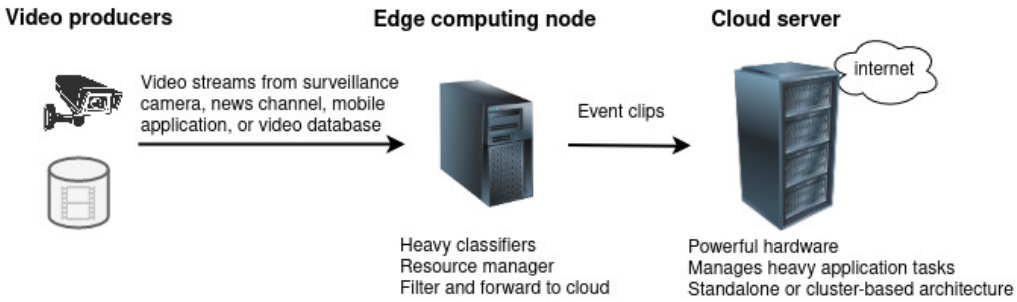

Figure 4. Basic overview of fully distributed architecture

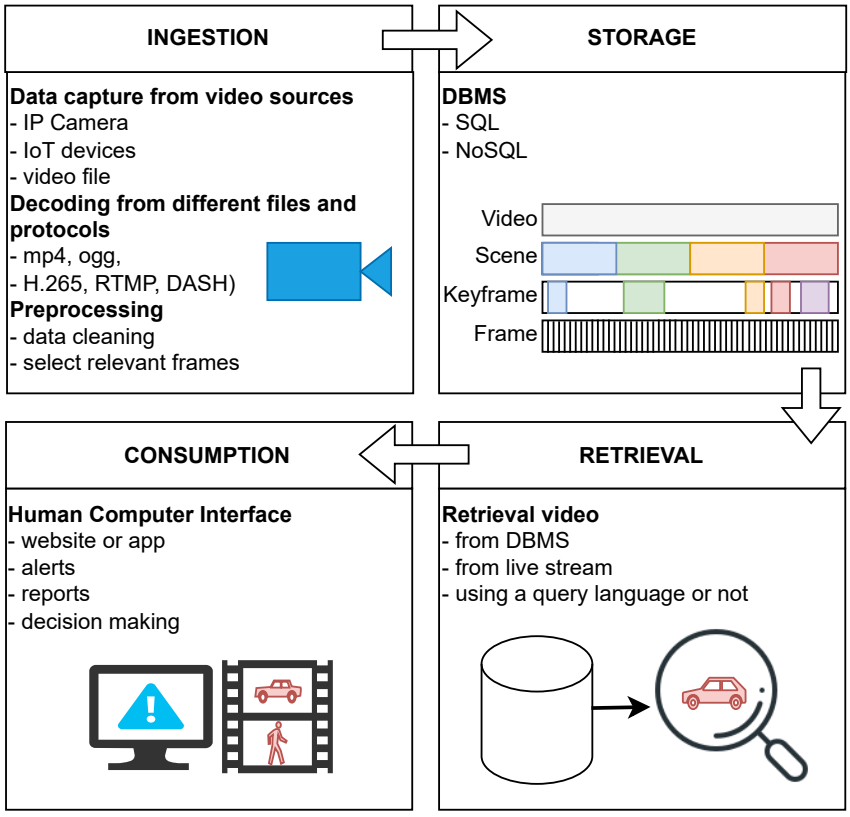

Figure 5. Processing pipeline overview

5.2.1 Ingestion

The ingestion phase has been extensively studied over the years, leading to the development of various approaches in the literature. To better understand this phase, it can be divided into several key components: video capture by CODECs (enCODer-DECoder), pre-processing tasks, object detection and labeling, ingestion time, and query time. Each of these components plays a crucial role in efficiently handling video data during the ingestion process.

Data preprocessing Image preprocessing involves manipulating raw image data to transform it into a usable and meaningful format. This process removes unwanted distortions and enhances specific characteristics that are crucial for computer vision applications. It serves as a fundamental initial step in preparing image data for input into machine learning models.

For example, consider the classic problem associated with a surveillance system: "The Road Traffic Authority may need a system that allows direct queries for tasks such as detecting high-volume traffic on specific roads, locating vehicles by characteristics (e.g., license plate, color, model, or a combination of features), or tracking suspicious vehicles traveling together (such as a criminal escort car" [Kang *et al.*, 2019a; Anderson *et al.*, 2019; Kang *et al.*, 2019b; Lu *et al.*, 2016; Zhang *et al.*, 2017; Xu *et al.*, 2019]. According to the literature, this scenario presents several challenges:

- Computational cost: Preprocessing can be viewed as an optimization problem or a multi-objective optimization

problem due to its computational demands. Applications dealing with large-scale videos or streaming data require solutions that avoid wasting computing power and communication resources.
- Many visual concepts: The interpretation of an image can range from simple (e.g., recognizing a car) to complex (e.g., identifying activities related to that car). Moreover, image detection models can fail under varying lighting conditions, object size, partial occlusion, and changes in position.
- Limitation of training data: To develop an efficient and generalized image detection model, a large dataset of images with appropriate parameters must be used. This requires skilled programmers, substantial time, and suitable hardware resources, all of which can be costly.
- Dynamic queries: In video surveillance systems, users expect the ability to query past events (e.g., how many people were at a particular bus station in July?) or real-time events (e.g., which streets are busiest at the moment?). This dynamic nature introduces additional complexity.
- Complex code: Many computer vision projects and libraries, often shared in repositories such as GitHub, feature complex codebases and data structures, adding further challenges for integration and maintenance.

Common preprocessing tasks include:

- Resizing: Standardizes the frame size to ensure consistent input for machine learning algorithms.
- Grayscaling: Converts color images to grayscale, simplifying the data and reducing computational requirements.
- Noise reduction: Removes unwanted noise from images using filters such as smoothing and blurring.
- Normalization: Normalizes pixel intensity values to a desired range, usually between 0 and 1, to enhance model training.
- Binarization: Converts grayscale images into black and white by applying a threshold.
- Contrast enhancement: Adjusts image contrast using techniques like histogram equalization to improve the visual quality of images.

Moreover, video processing on a large scale may require additional preprocessing strategies, particularly due to the volume and speed at which data is generated and stored:

- Varying image resolution: Lower frame resolution can accelerate inference time but may reduce detection accuracy [Xu *et al*., 2019].
- Frame compression: Video CODECs use redundant information within each frame to improve compression by identifying regions with high similarity and storing each region only once [Haynes *et al*., 2021].
- Adjusting frame rate ingestion: This strategy is similar to varying image resolution but focuses on accelerating inferences on live streams.
- Background subtraction: Helps identify foreground objects and extract motion vectors, isolating areas with moving objects [Daum *et al*., 2021].
- Frame difference detector: Highlights temporal differences between frames to determine whether the video content has changed [Kang *et al*., 2017].
- Analysis of specific parts of the video or frame: Identifies video sequences or frame regions most relevant for offloading to cloud servers for heavy computations, especially when the local node cannot handle the load [Pakha *et al*., 2018; Canel *et al*., 2019; Chao *et al*., 2020; Chaudhary *et al*., 2024].
- Automatic configuration adjustments for improved processing speed: Optimizes input configurations, including resolution, segment length, and sampling rate, to enhance query processing in accordance with predefined thresholds for accuracy, latency, and resource consumption [Chunduri *et al*., 2022; Zhang *et al*., 2024].
- Window size to process events: Defines a specific number of frames or time interval for event processing, avoiding the need to process the entire video due to high computational cost [Yadav and Curry, 2019a; Yadav *et al*., 2021; Yadav and Curry, 2019b; Yadav *et al*., 2020; Koudas *et al*., 2022].
- Mix of many different configurations: Employs various strategies to reduce computational costs according to specific thresholds or resource constraints [Poms *et al*., 2018a].

Heavy models and specialized models As previously stated, currently, CNNs are the most common type of DNNs, and they are widely used for locating and classifying objects in frames or images. Table 3 provides a temporal overview of the deep learning models utilized by the authors in their experiments.

Table 3. Temporal trends of deep learning models in reviewed studies

Models	First Use	Most Recent	Total Papers
YOLO (all variants)	2017	2024	42
ResNet (all variants)	2018	2024	27
Faster-RCNN	2018	2024	13
Specialized CNNs	2017	2022	13
MobileNet (all variants)	2018	2024	8
Mask RCNN	2019	2023	8
Inception (all variants)	2017	2020	7
VGG (all variants)	2017	2020	7
DeepSORT	2019	2024	5
SSD	2019	2024	5
AlexNet	2018	2021	3
SORT	2022	2023	2

In the recent years, two variants of CNN have been extensively used in the ingestion phase: Ground-Truth Models (GT Models) and Specialized Models, also called Cheap CNNs. GT models are the most widely studied because they are highly accurate, free, ready to use, and have a wide range of support for many programming languages. For example, the most used ground-truth models include Yolo, ResNet, Faster R-CNN, MobileNet, and Mask RCNN [Wang *et al*., 2021; Liu *et al*., 2016; Sandler *et al*., 2018; He *et al*., 2017].

However, they are costly in the context of video processing due to the high inference time. For example, according to Khani *et al*. [2023], a current state-of-the-art NVIDIA V100 GPU can support only two video streams running the YOLOv5-L model at 30 Frames Per Second (FPS). If running on large processing clouds, the total cost will be around $1,100 per month. This is also evidenced by the Keras[8] website, which shows the most common models and their CPU and GPU inference times.

As naive processing of all frames using expensive models is impractical for many tasks, especially in video or big data contexts, many researchers have proposed different types of small or Specialized Models [Kang *et al*., 2017, 2019a]. They are based on GT models but trained with fewer layers and focused on particular contexts. Because they are not generalists, Specialized Models are suitable for some types of applications, i.e., they act as binary classifiers, returning whether the specified objects are present in a given frame or image (e.g., a white car).

For example, Figure 6 presents a basic scheme for making Specialized Models. Training the specialized model requires a dataset of labeled data, usually from the same video stream in which this model will be used. Basically, the video stream is previously stored in a labeled dataset and then this dataset is used to train the specialized model.

Due to its limitations, the use of just one specialized model is not practical. For this reason, the common approach is to use a cascade of Specialized Models, each of which is more accurate and usually more expensive than the other [Xu *et al*., 2019], as shown in Figure 7. In the cascade set, each classifier only checks for the presence of a specific feature or class. If the accuracy is below a threshold, then the next classifier is called to check. The execution is stopped when the model produces a prediction with high-confidence (e.g., greater than 90%). Only when the cheap models fail, the ground-truth model used. Several authors developed their Specialized Models and cascade classifiers. They called cheap CNNs, small CNNs, proxy models, hierarchical models, specialized neural networks, and micro-classifiers [Anderson *et al*., 2019; Kang *et al*., 2019a,b; Hsieh *et al*., 2018; Kang *et al*., 2020b; Shen *et al*., 2014; Han *et al*., 2016; Shen *et al*., 2017; Mullapudi *et al*., 2019; Canel *et al*., 2019; Kang *et al*., 2020a; Chao *et al*., 2020; HÖnig *et al*., 2023].

Considering the significant human effort for proper software development and hardware capacity required to create a new detection model, several papers have proposed different strategies to create labeled video datasets and inexpensive models, including class refinement [Zhang and Kumar, 2019], CNN hyperparameter tuning [Anderson *et al*., 2019], approximate filtering Kang *et al*. [2019a], inference cost reduction [Hsieh, 2019], noisy data reduction [Agarwal and Netravali, 2023], query optimization [Chao *et al*., 2020], trade-off between accuracy and inference speed [Li *et al*., 2020b], optimization of temporal relations [de Boer *et al*., 2017], and probabilistic predicates [Kang *et al*., 2021; Yang *et al*., 2022; Romero *et al*., 2022; Moll *et al*., 2022; Lu *et al*., 2018].

A complete pipeline, from data ingestion to classification, is exemplified in Figure 8. In essence, detection models can be categorized into two groups, known as early and late operators, depending on the computational cost, the function, and the order in which they are employed. In a holistic view, the early operators, such as video ingest and decode, frame difference detector, and cheap models, are faster but intended for basic, low-cost tasks. They activate later operators such as GT models, motion detectors, object trackers, and OCR readers on a small fraction of the video for more in-depth analysis. Generally, the cost of late operators can differ by three orders of magnitude from early operators [Kang *et al*., 2017].

Ingestion time and querying time When querying a large number of images, such as video datasets or streaming video in real-time, the latency between the search command and the retrieval of the video is often regarded a very important problem [Yi *et al*., 2017; Hsieh *et al*., 2018; Hsieh, 2019]. However, the processing time between the user's search command and the system's retrieval typically varies from seconds to hours of intensive computing, depending on the activity envisaged [Kang *et al*., 2017; Hsieh *et al*., 2018]. In this context, a video query system usually performs tasks that need to be fast (or even in real-time), such as video ingestion and object detection. These systems also have to perform tasks that normally do not need to be done in real-time or even so fast, such as database reading, or storing consolidated data for persistence. For this reason, some studies have divided processing strategies into ingest time and query time [Hsieh *et al*., 2018; Hsieh, 2019; Chao *et al*., 2023].

a) Ingestion time (or online mode): focuses on data ingestion, data filtering, read frames, frame difference detector, and detection using cheap models. Given the fast response time, ingestion time is usually also referred to as online processing.
b) Query time (or offline mode): focuses on heavy processing, such as query processing, inference using GT models, cluster analysis, complex database operations, feeding trend dashboards, human-machine interaction (e.g., web page or mobile app), and expensive machine learning operations. Query time is also known as search time, profiling Zhang *et al*. [2017], or offline processing Anderson *et al*. [2019].

An overview of ingestion time and query time is shown in Figure 9. Hsieh *et al*. [2018] presented the first architecture that formalized and divided processing between ingestion time and query time. At ingest time, the solution uses cheap models to create an approximate index of all possible object classes for each frame. At query time, it takes advantage of this approximate index to provide low latency and, at the same time, compensate for the low accuracy of cheap models by using GT models when necessary. The objective is to respond "after the fact", or retrospectively, queries about objects of certain classes (e.g., cars, people) over many days of recorded video.

Similarly, retrospective video analysis has also been explored by Agarwal and Netravali [2023]; Kang *et al*. [2017, 2019a]; Wen *et al*. [2024]. In addition, some studies also consider a variety of ingestion time and query time separately,

[8] https://keras.io/api/applications/

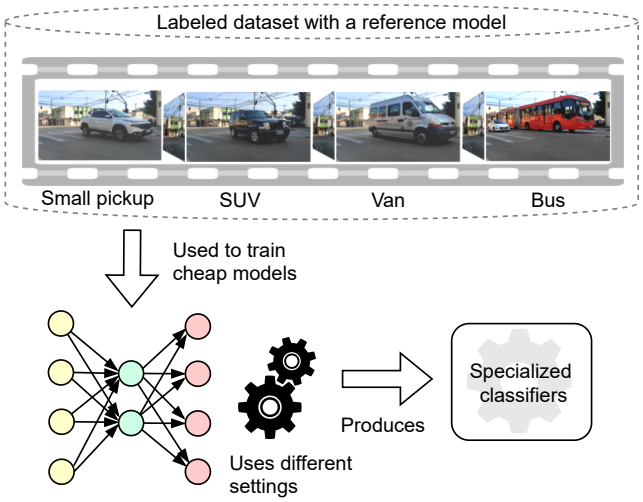

Figure 6. Basic scheme for making specialized classifiers

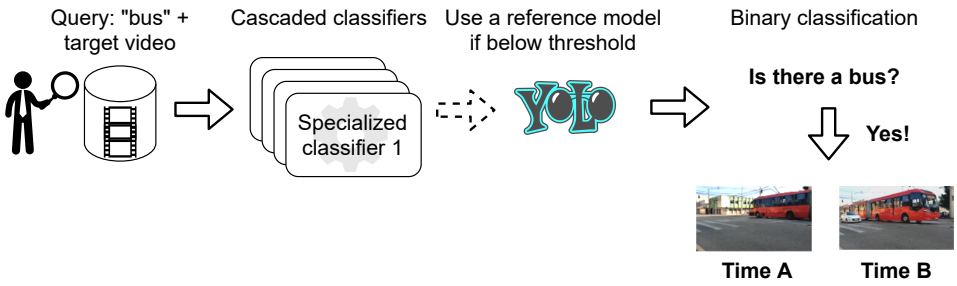

Figure 7. The use of cascaded classifiers to process video queries

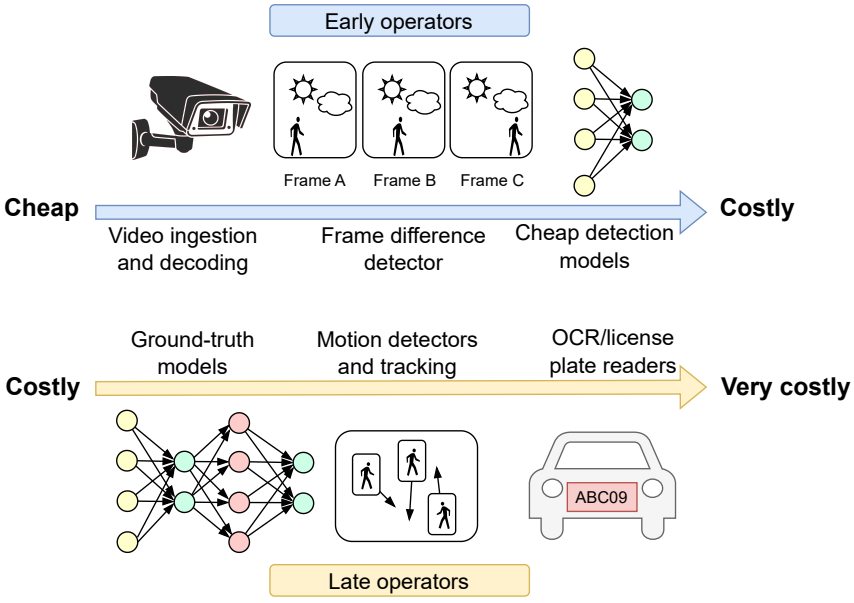

Figure 8. Cascade of operators and their computing cost

using different strategies to manage resources, online and offline processing Zhang *et al*. [2017]; Hsieh *et al*. [2018]; Kang *et al*. [2017, 2019a].

Moreover, various approaches have been proposed for different types of problems. For example, Zhang *et al*. [2017] perform mathematical operations to distribute processing in a cluster in which a scheduler optimizes video resolution, frame rate and sliding window settings to minimize the delay caused between processing and search time. Hung *et al*. [2018] used a centralized manager and working machines to execute queries configured as a Direct Acyclic Graph (DAG), in which each transformation processes a

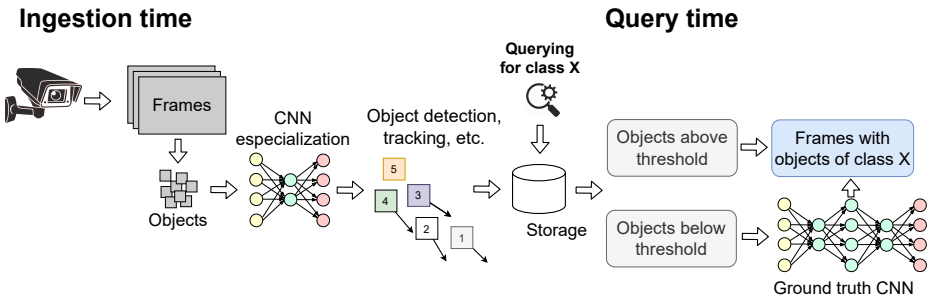

Figure 9. Basic overview of ingestion time and query time

time-ordered stream of messages (e.g., video frames). Chen *et al*. [2022] built indexes for all desired objects of the given videos during an ingestion time and evaluates query answers efficiently in the query time. In a different way, Poms *et al*. [2018a] represent video collections as a table in a database that performs calculations expressed as data flow graphs on those frames.

5.2.2 Storage

With advances in information technology, the amount of data produced and stored is increasing very fast. Consequently, multimedia content (e.g., audio, image, video) is widely used by humans and machines (e.g., IoT devices) for many applications today. Therefore, the need to store, organize, and efficiently retrieve this data has been a research motivation for companies and universities for decades [Collins, 2020; Lu *et al*., 2015]. However, traditional relational databases are not suitable for dealing with multimedia data due to the type of unstructured data, the large volume, and the speed with which it is produced. Even more recently, streaming applications bring additional complexity because they need to process, store, and retrieve in real-time [Yadav and Curry, 2019b].

Figure 10 presents an overview of a common video database system. Typically, video sources such as closed-circuit television (CCTV), cell phones, and IoT devices send content to a server responsible for storing the raw footage and video metadata, such as objects, events and the time of occurrences. The metadata is then used by query engine to help search the raw footage. On the other side, an user interface such as a webpage or mobile application allows some interaction with this database to obtain relevant videos of interest, alerts of events, and situations that happened.

An interesting method for storing the spatiotemporal attributes of video events was introduced by Yadav and Curry [2019a,b]; Yadav *et al*. [2020]. Figure 11 shows an example of this approach. Given a video stream, the system detects objects along with their attributes (e.g., car1, red) and stores them in the graph database. The objects are represented as nodes, while their spatiotemporal relationships (e.g., left, below, after) are stored as edges connecting the nodes. For each frame the database is updated according to the video stream. While this approach is quite interesting, it can be costly in terms of processing time and query latency.

Moreover, while NoSQL databases offer several advantages, they generally lack the key transaction properties—Atomicity, Consistency, Isolation, and Durability (ACID) – that are inherent to relational databases. An ACID transaction is defined by a database operation that exhibits these properties, and systems that support these operations are classified as transactional systems [Andreas Meier, 2019]. The properties of each read, write, or modification of a table are guaranteed by ACID transactions. However, NoSQL intentionally sacrifices these properties for improved performance and scalability.

Even so, several authors have developed new video database management systems. For instance, Haynes *et al*. [2021] presented a new video storage system designed to automatically organize the data in the storage hardware structure in an efficient and granular format, eliminating the redundancies found in videos captured from multiple cameras. On the other hand, Daum *et al*. [2021] proposed a new system that uses spatial random access to encode videos and optimize the file layout according to the content.

In turn, Collins [2020] presented a prototype of a query engine for video data using active and relational database concepts. The author developed an ingest module that processes video sources with a GPU server and stores data and metadata about each frame. In addition, a retrieval module provides a user interface that accepts some search and query criteria and presents matching video queries.

Still, Krishnan *et al*. [2018] studied execution trade-offs in visual analytics and illustrated a complex relationship between storage, latency, and accuracy. Also, Haynes *et al*. [2018] presented a new database management system for managing virtual, augmented, and mixed reality video content, offering a query language and algebra, allowing declarative queries. Madden *et al*. [2024] developed a new database capable of ingesting, storing, processing, and querying all types of data, integrating both relational and non-relational database features.

5.2.3 Retrieval

There are several image indexing and retrieval techniques. The first technique consists of associating images with a specific object or location of interest (e.g., a given image is a photograph of a specific building on the Stanford campus [Kang *et al*., 2017]).

The second technique consists of searching for similarities in a large corpus of images (e.g., reverse image search

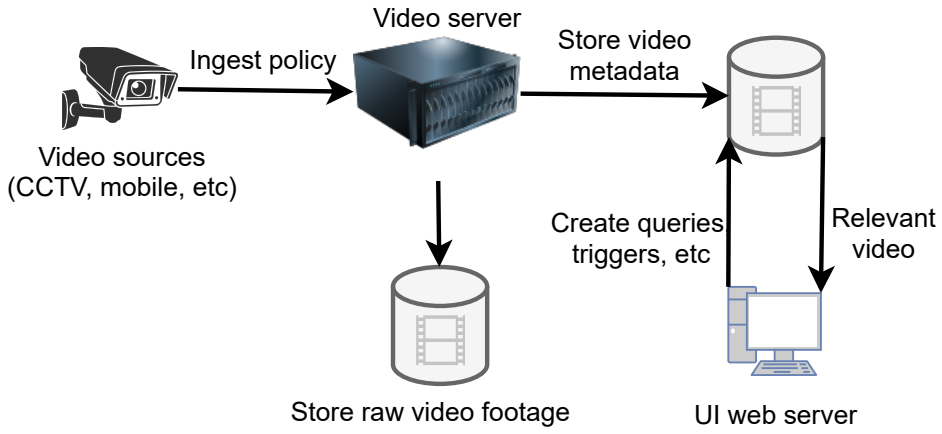

Figure 10. Ingestion and retrieval scheme of the video database

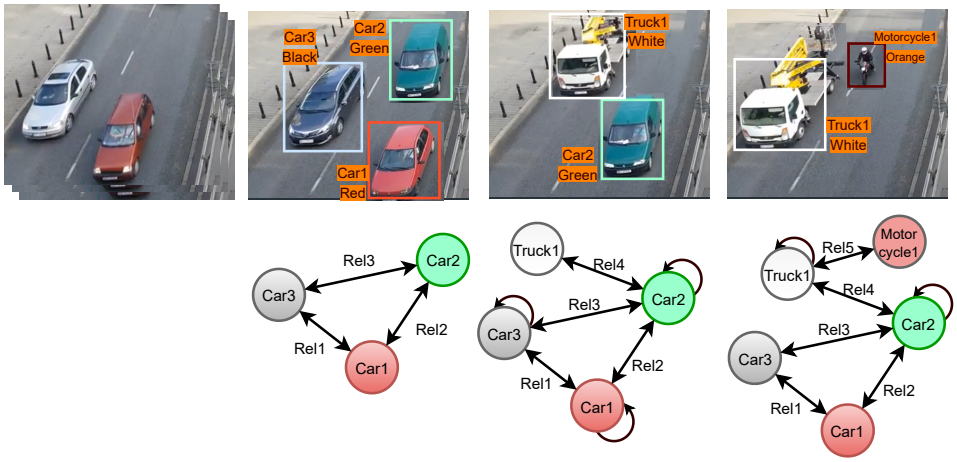

Figure 11. How the graph database represents spatiotemporal relationships

on Bing Visual Search). It can be achieved, for instance, using histogram comparisons, template matching, and feature matching. Data mining algorithms, mainly based on clustering, such as k-nearest neighbors, can also perform binary classification.

The third technique is content-based video indexing and retrieval, which uses advanced video segmentation, feature extraction, dimensionality reduction, and machine learning algorithms Spolaôr *et al*. [2020].

The fourth technique uses approaches based on deep learning (DL), since it revolutionized image similarity tasks using pre-trained Deep Convolutional Neural Networks (DCNNs), such as ResNet, VGG, and Inception, or inexpensive models.

The fifth technique consists of storing video metadata, such as object, color, plate, and timestamp, and using it to retrieve the original footage. These metadata usually need to be obtained using specific detection models for each task.

The sixty technique is a novel approach called NOP Query, introduced by Kossoski *et al*. [2024]. It is a video event query engine based on the Notification Oriented Paradigm (NOP) and leverages existing tools [Neves, 2021] to efficiently handle multiple video events and respond with low latency when matches are found. NOP offers a fresh perspective on software development, where small, collaborative entities execute tasks and make logical decisions based on accurate notifications. One of the key advantages of NOP Query is its ability to avoid resource-intensive database operations, such as storage, retrieval, and triggers.

5.2.4 Consumption

The consumption phase interacts directly with end users or other machines to provide the results obtained by the video processing system. In general, end users write some query language to perform tasks, such as finding a specific object or event or configuring triggers to be fired when a certain condition occurs. Some solutions offer a human computer interface (HCI), such as a web page or mobile application, to improve usability. However, most studies do not offer query language or HCI. More details about video query languages are presented later in Section 6.

For example, Hsieh *et al*. [2018] offer a simple interface in which the user selects the target video and the object class. A piece of video is displayed if the system encounters the object.

However, Kang *et al*. [2019a] also introduce a user-friendly web-based interface that supports a SQL-like query language, including operators such as SELECT, FROM,

GROUP BY, HAVING, SUM, LIMIT, and GAP. Users can search by video name, timestamp, and object class, and filter results based on the frequency of object instances. When a query matches metadata in the database, the system returns the corresponding event timestamp and associated video segment.

6 RQ2: What video query languages exist, and what operations do they support?

Query languages are essential for efficient event retrieval in large-scale video datasets, providing a structured means to express complex temporal, spatial, and semantic conditions. They allow users to abstract low-level visual features into high-level event definitions, thereby enhancing scalability, reproducibility, and automation in video analytics. When designed to support real-time applications, such languages enable the rapid and accurate detection of relevant events.

Section 6.1 discusses commonly used video query operators, while Section 6.2 introduces advanced video query languages designed to support more complex event retrieval.

6.1 Common video query operators

In the literature, there are several ways to perform video queries. The most common approach is to utilize the same query language as the database system that stores the video or its associated metadata [Anderson *et al*., 2019; Collins, 2020; Stonebraker *et al*., 2020; Sipser, 2020; Chao *et al*., 2023]. Some works extended SQL with spatiotemporal operators or event query capabilities, creating a new language [Xarchakos and Koudas, 2019; Lu *et al*., 2015; Chao *et al*., 2020; Yadav and Curry, 2019b; Kang *et al*., 2019a]. Still, other works do not mention any query language, although they perform queries [Kang *et al*., 2017; Zhang *et al*., 2017; Kang *et al*., 2022; Hsieh *et al*., 2018]. In any case, all of these approaches use the so-called query operator. For example, according to this literature, the most common query operators are the following:

- Objects: This operator is used to retrieve the object of interest, such as a person or a vehicle.
- Object attributes: Attributes provide additional information about objects. For instance, when considering a specific person as an object, attributes may include height, clothing, and accessories such as a hat or glasses.
- Spatiotemporal operators: Spatiotemporal operators are two key features that distinguish video query languages from others, as illustrated in Figure 12. Spatial operators pertain to the positioning of objects in space and their spatial relationships. For example, "The white truck is to the *left* of the red car". Temporal operators, on the other hand, relate to the order in which objects appear over time and their temporal relationships. For example, "The white truck appears *before* the red car". A detailed discussion of spatiotemporal operators is provided in Yadav and Curry [2019b]; Yadav *et al*. [2020].

Table 4 summarizes the many video query languages and their support for the most common operators. Moreover, these primary operators can be subcategorized as detailed in Table 5.

6.2 Advanced video query operators

Some video query languages incorporate advanced features that enhance expressiveness and usability. For instance, Streaming Video Queries (SVQ) [Lu *et al*., 2015; Xarchakos and Koudas, 2019; Chao *et al*., 2020] extends traditional SQL syntax while leveraging a relational database management system to translate each statement into corresponding SQL queries. Its grammar supports standard SQL operators as well as variable declarations and spatiotemporal constructs, enabling more effective video stream querying. Listing 1 selects all the frames in which there is a car to the left of a yellow bus. In the query syntax, Ci are classifiers for different types of objects, such as vehicle types and colors, and Fi are bounding box features for objects in the frame, using vehDetector, an object detection algorithm.

Algorithm 1 Example of the SVQ query language

```
1:  SELECT sensorID, timeStamp, Object1
2:  (Feature1 (objectBox1)) AS objectClass1,
3:  Object1 (Feature1 (objectBox2)) AS objectClass2,
4:  Object2 (Feature2 (objectBox1)) AS objectColor
5:  FROM (PROCESS inputStream
6:  PRODUCE sensorID, timeStamp, objectBox1, objectBox2
7:  USING ObjectDetector)
8:  WHERE objectClass1 = truck
9:  AND objectColor = red
10: AND objectClass2 = sedan
11: AND (ORDER(objectClass1, objectClass2) = PRIOR) =0
```

FrameQL [Kang *et al*., 2019a] extends SQL with spatiotemporal capabilities, enabling video queries using relational algebra over a table-like schema. Videos are represented as virtual relations, where each tuple corresponds to an object in a frame. Attributes such as time, location, object class, bounding boxes, and object identifiers are automatically populated through computer vision techniques. This structure supports object tracking across multiple frames and facilitates expressive, structured video querying.

Some of the video query operators discussed below have emerged more recently in the literature, particularly in the context of event query languages and CEP systems. Further details can be found in Kang *et al*. [2019a, 2022]; Yadav and Curry [2019a]; Cugola and Margara [2012].

- The $aggregation$ operator allows computing some statistics over the video frames. Common aggregations include count, sum, average, maximum, and minimum operator. For example, compute the average number of cars per frame.
- The $conj$ operator retrieves the objects(s) of interest, evaluating two or more attributes. For example, show

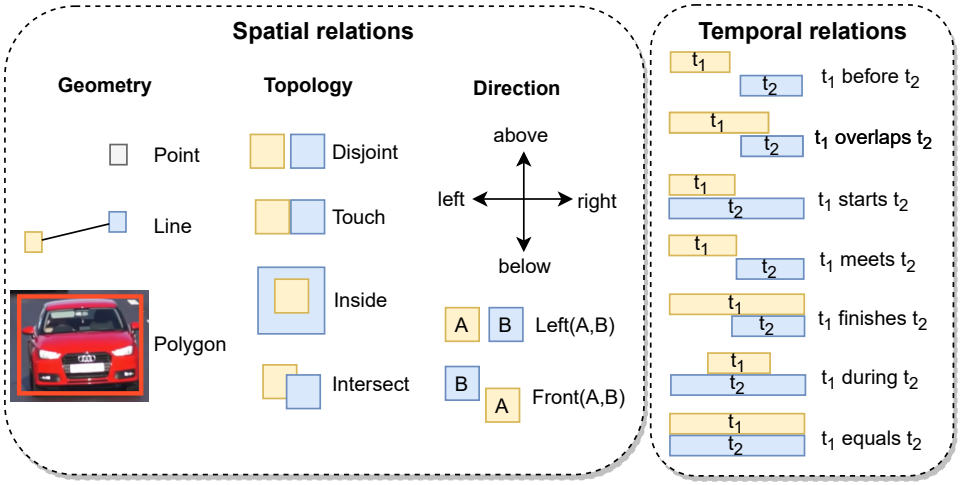

Figure 12. Overview of spatiotemporal relationships

Table 4. Video query languages and their support for common operators: ✓="yes", *P*="partially", –="no"

	Objects	Attributes	Spatial operators	Temporal operators
EVAQL [Kakkar *et al.*, 2023]	✓	*P*	–	–
FRAMEQL [Kang *et al.*, 2019a]	✓	✓	–	–
ROLE [Liu *et al.*, 2018]	✓	✓	✓	✓
SVQ [Xarchakos and Koudas, 2019; Chao *et al.*, 2020]	✓	✓	✓	–
SVQL [Lu *et al.*, 2015]	✓	✓	✓	✓
VEQL [Yadav and Curry, 2019b]	✓	✓	✓	✓
VRQL [Haynes *et al.*, 2018, 2020]	✓	✓	✓	✓

Table 5. Other types of operators in video query languages

Operator type	Description
Semantic	Involve high-level meaning (e.g., person carrying object).
Probabilistic	Allow uncertainty and confidence scores in matching patterns.
Logical	Combine multiple conditions (e.g., AND, OR, NOT).

the number of instances of cars and trucks in a time window.

- The *count* operator is a type of aggregation. Some approaches extend this operator by supporting more features, such as counting objects between frames. For example, to warn if there is high-volume traffic on a street.
- The *iteration* operator defines the repeated occurrences of a matching event, similar to a loop in a programming language.
- The *join* operator allows performing a join and, subsequently, a *selection*, *aggregation*, or *limit* query. For example, an Amber Alert application can check whether the detected license plates belong to a stolen car database.
- The *limit* operator allows finding a cardinality-limited number of events occurring within some time interval. For example, selecting ten instances of buses at stop signs.
- The *projection* operator extracts only a part of the information to compute video events.
- The *renaming* operator changes the name of a field.
- The *selection* operator allows the selection of particular objects or events of interest. For example, selecting from all instances of a person in a video database.
- The *sequence* operator retrieves the occurrence of objects of interest, evaluating their temporal relationship. Typically, the sequence operator is used with the *window* operator. For example, select a "car" object and a "truck" object that appears in the same 10-second time window.
- The *similatiry* operator allows searching for portions of the video similar to a reference frame or video clip. For example, given a picture or video clip of a soccer player's goal, find similar events. Such queries often involve iterative, ad-hoc analysis to arrive at the final query.
- The *window* operator establishes an interval scope in which the query should be performed, generally based on frame numbers or timestamps. It is often used to limit the time interval that a query works, aiming at reducing the computational cost of searches. For example, bus and truck objects can be selected to travel together within 10 seconds.

Complex Event Processing (CEP) [Cugola and Margara, 2012], Complex Event Recognition (CER) [Giatrakos *et al.*, 2020], or Complex Event Detection (CED) [Honarparvar *et al.*, 2024] refers to collections of simple events that derive

from complex events when they satisfy some pattern. This feature allows processing systems to react to events. CEP languages, for example, can query complex patterns that correspond to input events based on their content, input order, and relationships. There are several CEP systems and languages in the literature that differ in their architectures, data models, pattern languages, and processing mechanisms. Two surveys about CEP systems were published in Cugola and Margara [2012]; Giatrakos *et al*. [2020].

Basically, the input of a CEP system is a stream of events, also called Simple Derived Events (SDEs), along with a set of patterns, defining relationships between the SDEs. An event has the structure of a tuple of values that can be numeric or categorical (e.g., event type, timestamp). During the event computation, it detects instances that satisfy an expected pattern and produces complex event output. Time is critical, so the temporal formalism defines the detection patterns According to Giatrakos *et al*. [2020], there are three main types of CEP systems:

- In automata-based systems, patterns are usually defined in a language similar to SQL that is later compiled into some form of automata (often non-deterministic) for pattern matching. The automaton is then fed with the data stream, changing state as the predicates on the current state transitions are satisfied.
- In logic-based systems, patterns often have the form of a rule, with antecedent, consequent, and condition which, if satisfied, lead to the detection of a CEP. Many underlying mechanisms for performing inference can be used, from PROLOG-based systems to directed graphs (similar to automata).
- The tree-based systems differ from the other two above-mentioned systems in many ways: they assume that CEPs avoid semantic ambiguities when hierarchies of events are present, and translate patterns to trees, whose leaves store events and internal nodes correspond to operators.

Beyond the above-mentioned systems, there are also hybrid approaches that consider a mixture of trees, automata, and logic concepts, but the consequences on the semantics, soundness, and completeness are unclear. The CEP languages use operators to perform queries. The most basic are $selection$ and $sequence$. There are many operators, but not all languages support them [Giatrakos *et al*., 2020].

- $Selection$: Selects those events whose attributes satisfy a set of predicates/relations, temporal or otherwise.
- $Sequence$: Two events following each other in time.
- $Disjunction$: Either of two events occurring, regardless of their temporal relation.
- $Iteration$: An event occurring N times in sequence, where $N > 0$.
- $Conjunction$: Both events occur, regardless of their temporal relation.
- $Negation$: Absence of event occurrence.
- $Projection$: Returns an event whose attribute values are a transformed subset of the attribute values of its sub-events.
- $Windowing$: The event pattern must occur within a specified time window.

For instance, the Video Event Query Language (VEQL) [Yadav and Curry, 2019b; Yadav, 2019, 2021; Yadav *et al*., 2021] distinguishes itself from other query languages by being based on Complex Event Processing (CEP) systems and graph databases. In a CEP system, information is conveyed as notifications of events occurring in the external environment. The engine filters these events and combines them into conditions that represent higher-level events [Cugola and Margara, 2012]. Consequently, VEQL focuses on detecting specific occurrences of low-level patterns and events that form part of higher-level events. When a match is detected, the CEP mechanism notifies the relevant parties. According to Yadav and Curry [2019b], the key features of CEP systems include the straightforward expression of events and the real-time detection of patterns. VEQL offers a wide variety of spatiotemporal operators and low-latency query correspondences.

7 RQ3: How is the quality of existing approaches assessed, both quantitatively and qualitatively?

Evaluation metrics are essential for assessing the quality of the proposed approaches and for enabling comparisons with existing literature. In this section, the metrics are grouped into three categories: (i) machine learning and throughput, (ii) video query latency, and (iii) system configuration. Section 7.1 describes the machine learning metrics employed to evaluate throughput. Section 7.2 focuses on the metrics used to measure video query latency. Finally, Section 7.3 outlines the system configuration metrics considered in the reviewed studies.

7.1 Machine learning metrics and throughput

Five measures are frequently used in many deep learning algorithms to evaluate the performance and processing cost of models, which makes them valuable comparisons: accuracy (equation 2), Precision (equation 3), Recall (equation 4), F1 score (equation 5) and Average Precision (equation 6) Hsieh *et al*. [2018]; Bastani *et al*. [2020]; Romero *et al*. [2022]; Yadav [2021]; Fu *et al*. [2019]; Mao *et al*. [2019]. In the aforementioned equations, *TP*, *FP*, *FN* means, respectively, True Positives, False Positives, and False Negatives. *Precision* indicates the proportion of true positive predictions. Classification *accuracy* is the total number of correct predictions divided by the total number of predictions made for a data set. *Precision* quantifies the number of positive class predictions that belong to the positive classes. *Recall* is a completeness metric that specifies the proportion of detected positives. *F1-score* combines Precision and Recall and is the harmonic mean of these two metrics. Average Precision (AP) is a way to summarize the precision-recall curve into a single value representing the average of all precisions.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

$$F1\_score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (5)$$

$$\text{AP} = \sum_{k=0}^{k=n-1} (Recalls_n - Recalls_{n-1}) Precisions_n \quad (6)$$

Moreover, the authors in Kang *et al*. [2017] proposed the *cost-based model* (CBO), which combines model specialization and difference detectors using inference-optimized model search. It is used to find a cascade of high-quality models (e.g., two vs. four-layer specialized model) and thresholds (δ_{diff}, c_{low}, and c_{high}). The CBO receives a video and target precision values as input, where $FP*$ and $FN*$, for the false positive and false negative rates, respectively. Formally, it solves the following problem (Equation 7):

$$\begin{aligned} &maximize\ E(throughput) \\ &false\ positive\ rate < FP* \\ &false\ negative\ rate < FN* \end{aligned} \quad (7)$$

Equation 8 presents the CBO. The selectivities (f_s, f_m, and f_c) are estimated from a sample of the data. The expected execution time per frame is given using T_{MSE}, $T_{SpecializedNN}$ and T_{FullNN} which are the execution times per frame for the MSE filter, the specialized NN and the reference NN respectively. These execution times are data-independent, so that they can be measured once per hardware platform.

$$CBO = f_s T_{MSE} + f_s f_m T_{SpecializedNN} + f_s f_m f_c T_{FullNN} \quad (8)$$

The authors in Mao *et al*. [2019] proposed the metric *mean delay*, which emphasizes the early detection of objects in an instance. Delay means the number of frames from the initial frame of an object sequence to the first frame in which the object is detected.

Moreover, the video querying community used more metrics, including throughput (FPS) [Hwang *et al*., 2022; Koudas *et al*., 2022; Haynes *et al*., 2018; Bastani *et al*., 2020; Kang *et al*., 2017; Chen *et al*., 2021] and bandwidth usage [Yadav, 2021; Poms *et al*., 2018b; Anderson *et al*., 2019; Hsieh, 2019].

7.2 Video query latency metrics

Query latency refers to the time taken between the start of the query and the retrieval or matching of the video. It is an important topic due to the recent advances in hardware and software mentioned in other sections. Recently, two doctoral theses and related papers have presented solutions in this research area [Hsieh, 2019; Yadav, 2021].

According to Hsieh *et al*. [2018]; Hsieh [2019], current systems suffer from high costs at the ingestion phase and high latency at the query time. The solution presented divides the query processing work between ingest time and query time, performing low-cost analysis at ingest time and facilitating low-latency queries on recorded videos. In summary, those work presented three approaches to keeping ingestion cost and query latency low whereas still achieving the retrieval and precision goals specified by the user:

- Top-K Index: It makes indexing more efficient during ingestion, using specialized models for each video.
- Grouping similar objects. At ingestion time, it groups similar objects using cheap resource vectors. At the query time, each cluster executes only the centroid using GT-CNN and applies the result to all the objects in the cluster.
- Balance ingestion and query costs: It automatically calculates the ingestion CNN, the K value, the specialized models, and the parameters for the desired precision and recall targets.

In turn, Yadav and Curry [2019b] proposed a series of throughput calculations for video streams.

- Matcher latency (Equation 9): It is the time difference between when the window state is sent to the matcher and when the pattern matches.
- System latency (Equation 10): It is the sum of the average event representation time, the window size (w), and the matcher latency.
- Event Representation: It is the time taken by the video stream processor to convert the frame into a graph.
- Event query accuracy: It examines how many relevant event patterns were detected for each query compared to the ground truth.

$$t_{matcher-latency} = t_{notify} - t_{window\ send\ to\ matcher} \quad (9)$$

$$\begin{aligned} t_{system-latency} = \\ (\sum_{i=1}^{w} (t_{event-rep\ for\ window\ size(w))/w}) \\ + t_{matcher-latency} \end{aligned} \quad (10)$$

7.3 Configuration "knobs" metrics

The frame and video configuration (also called knobs) refers to the set of values, options, or parameters of a frame or video. Examples of settings for frames are size, format, and colors, and for videos, are codecs, sampling rate, and playback speed. In turn, configuration metrics act as filters that determine which and how many frames to process. Because of their importance and diversity, knobs directly influence the system's performance. Therefore, choosing the right structures to use is an essential task. A module called profiler typically extracts measurements such as throughput, processing time, memory consumption, and GPU usage in order to allocate resources in the system.

For instance, as noted by Romero *et al*. [2022]; Kang *et al*. [2022], processing all frames with expensive models is impractical. To address this, they propose optimizations like selectively using faster, less accurate models to replace or filter

out frames that would otherwise require expensive models. In addition to frames and videos, the optimization process also considers all relevant features, which impact the overall ingestion and retrieval time.

The authors in Du *et al*. [2020] introduced the bandwidth usage metric, which calculates the total system cost, encompassing the camera processing, network transmission of the video, and server operations. Additionally, the average response delay metric measures the average processing delay per object, pixel, or segmentation. This metric accounts for detection tasks, classification, and the time required to send data to the server and perform inference.

The authors in Chen *et al*. [2021]; Dai *et al*. [2022] proposed techniques for organizing detected objects during an intermediate processing stage, aiming to reduce the number of objects and frames that need to be evaluated.

The authors in Poms *et al*. [2018b] outline various methods for organizing video collections, including using tables for frame sampling and pixel processing calculations represented as data flow graphs. Additionally, the paper examines the performance of video decoding, the scheduling of graphs, and the scalability of both single and multiple machine setups.

The authors in Moll *et al*. [2022] propose processing queries through chunk-based adaptive sampling. This approach identifies video frames containing objects of interest without the need to run an object detection algorithm on every frame, which is typically computationally expensive. They also present an extensible set of definitions and equations to support their method.

The authors in Zhang *et al*. [2017] approach query processing as an optimization problem, offering various equations to predict query delay, load balancing, query distribution, and other related factors.

The authors in Zhang *et al*. [2023] initially employed a lightweight preprocessing module to eliminate noise and minimize the data transmitted in the video. Subsequently, they retrieved information from the noise-free video through a server-side enhancement module. Several strategies were proposed in this context.

8 RQ4: What datasets are available for training and evaluating models and frameworks?

The availability of suitable datasets is crucial for advancing research in querying large volumes of video data, particularly for the training and evaluation of machine learning models and frameworks. Table 6, presented in this article, provides an overview of the datasets identified in the reviewed literature.

Characteristics and Sources of Datasets

A substantial portion of these datasets consists of YouTube streams. These videos encompass various sources, such as:

- Traffic Intersections: Used for analyzing vehicle flow and detecting anomalies.
- Surveillance Cameras: Employed in security scenarios, enabling the detection and tracking of people and vehicles.
- News Channels: Offer varied content for activity and event recognition.

Challenges and Needs Regarding Datasets

Despite the variety of available datasets, this section highlights significant challenges regarding their accessibility and practical use. A major limitation identified is the reliance on private or restricted datasets in many studies, which compromises reproducibility. However, to enable benchmarking and fair comparisons, the use of publicly available datasets is essential.

This underscores the critical need to:

- Increase the availability of public and labeled datasets: This would facilitate the validation and comparison of new approaches within a standardized environment.
- Standardize data formats and annotations: This would promote interoperability and reduce the effort required for pre-processing.
- Encourage collaboration in dataset creation: This would help address more diverse and complex scenarios, overcoming limitations in training data.

Overcoming these challenges is fundamental to advancing research and developing more robust and efficient systems for querying large video databases.

9 Conclusion, trends and challenges

This is the first research on query video analytics, also called query video streams, a niche with many approaches and contributions that differ from other video processing methods. To carry out this research, this SLR presents several relevant works published in 99 studies from early 2014 to 2024.

As a result of this large volume of information, it is possible to summarize the following trends and challenges:

Standardization and Query Interfaces

- **Lack of a standard video querying language:** Despite various initiatives to develop video querying languages, none have achieved widespread adoption. Currently, there is no standardized, ANSI-SQL-like language specifically designed for querying video content.
- **High-level querying interfaces:** Recent research explores the use of modern human-computer interaction concepts to develop intuitive interfaces that enable users to declare queries and receive responses efficiently.

Processing Location and Infrastructure Constraints

- **Edge-first processing paradigms:** Emerging approaches increasingly avoid transferring data between edge nodes and the cloud, due to high-speed internet requirements and latency issues. Furthermore, many IoT devices are now capable of performing advanced tasks locally.

Table 6. Datasets used for video querying

Dataset	Classes or event pattern	Available data
Jackson-town-square [Kang *et al*., 2017]	Vehicles	Complete dataset and annotations
Coral [Kang *et al*., 2017]	People	Complete dataset and annotations
Jackson hole [Hole, 2018]	People, vehicles	Live video streaming only
Taipei [Technologies, 2019]	People, vehicles	Live video streaming only
Lausanne [de Lausanne, 2020]	People	Live video streaming only
Oxford [School, 2018]	People	Live video streaming only
Alburn [of Auburn, 2023]	People, vehicles	Live video streaming only
Kabukicho [Kabukicho, 2020]	People, vehicles	Live video streaming only
Amsterdam [Lemmer, 2019]	People, vehicles	Live video streaming only
HMDB [Kuehne *et al*., 2011]	Horse, bike	Human actions
UCF-101 [Soomro *et al*., 2012]	Horse, bike	Action recognition data set
UT-Interaction [Ryoo *et al*., 2010]	Handshaking, punching	Human-human interactions
SBU Kinectic [Yun *et al*., 2012]	Handshaking, punching	Humans performing interaction activities
DETRAC [Wen *et al*., 2020]	High volume traffic	Multi-object detection and multi-object tracking benchmark
Street scene [Ramachandra and Jones, 2020]	Jaywalking	Various videos
VIRAT [Oh *et al*., 2011]	Parking lot status	Activity detection in multi-camera environments
PEXELS [Pexels, 2021]	Vehicles	Image and video database
Caltech Vision Group [Caltech, 2016]	People	Images for 101 categories with about 40 to 800 images per category
PETS [Patino *et al*., 2016]	People, vehicles	Surveillance systems
TRECVID [National Institute of Standards and Technology, 2016]	People	Video retrieval evaluation
DukeMTMC [Ristani *et al*., 2016]	People	Video tracking, person re-identification, and low-resolution facial recognition
Xipg [Xipg, 2021]	People	Video test media
MOT16 [MOT2016, 2022]	Pedrestrians, vehicles	Multiple object tracking
Urban tracker [Jodoin *et al*., 2014]	Vehicles	Multiple object tracking in urban mixed traffic

- **Resource-intensive pipelines and optimization needs:** Video query pipelines require significant GPU, CPU, memory, and network resources. Therefore, algorithms focused on filtering activities and data retrieval must be further developed to alleviate existing bottlenecks.

Real-Time and Scalable Processing

- **Real-time query processing and adaptive models:** There is a rising demand for query languages that support live video streams rather than solely historical data. Online learning algorithms are also essential to address concept drift and enable continuous model adaptation.
- **In-memory computation and search space reduction:** To manage large-scale video data efficiently, minimizing read/write operations to databases is critical. Innovative algorithms and protocols are needed to reduce computational costs and improve search efficiency.

Machine Learning Models and Data Availability

- **Cost-effective model generation:** Ground truth model generation remains expensive at scale. New libraries aim to automate the creation of lightweight and context-aware models.
- **Support from oracle users:** Incorporating expert users who can annotate events or identify emerging classes can enhance the accuracy and responsiveness of detection models.
- **Need for labeled datasets and reproducibility:** Many studies rely on private or inaccessible datasets, hindering reproducibility. Benchmarking and fair comparisons require publicly available datasets.

Privacy, Language, and Implementation Concerns

- **Privacy-preserving video processing:** Privacy concerns are growing due to recent legal frameworks, yet research addressing these challenges remains limited.
- **Programming language limitations:** Python is widely used in video processing pipelines, but it may introduce performance bottlenecks compared to more efficient languages such as C++ or Java.

In short, these are the main gaps found:

High Computational and Infrastructure Costs

- Video processing remains a resource-intensive task, with numerous studies reporting high computational and operational costs [Kang *et al*., 2017; Hung *et al*., 2018; Wang *et al*., 2018; Kang *et al*., 2019a; Kraft *et al*.,

2020; Lai *et al*., 2021; Hwang *et al*., 2022; Li *et al*., 2023b; Khani *et al*., 2023].
- Both relational and graph-based databases contribute to system overhead. Relational models incur significant costs due to triggers, stored procedures, and transactional guarantees [Collins, 2020; Stonebraker *et al*., 2020; Sipser, 2020; Chao *et al*., 2023], while graph databases, although flexible for representing video semantics, also demand substantial resources [Poms *et al*., 2018a; Qin *et al*., 2021; Yadav *et al*., 2021; Yadav, 2021].

Query Language and Programming Challenges

- The lack of a standardized query language tailored for video data severely hinders interoperability and scalability, highlighting the need for a solution akin to ANSI SQL for structured data.
- Additionally, current deployment strategies rely heavily on complex imperative programming paradigms, which complicates the development and maintenance of video query pipelines [Kang *et al*., 2019a; Kang, 2022].

Query Latency and Spatiotemporal Complexity

- High latency remains a critical challenge in video query processing, particularly for real-time or interactive systems [Hsieh *et al*., 2018; Yadav, 2019; Chao *et al*., 2020; Hsieh, 2019; Chao *et al*., 2023].
- Furthermore, the complexity of detecting and retrieving spatiotemporal events imposes an additional burden, often requiring sophisticated indexing and event modeling techniques [Yadav and Curry, 2019a; Yadav, 2021].

Finally, this study provides a set of guidelines to aid readers in understanding the broader context of the field, synthesizing key approaches, emerging trends, and prevailing challenges through a comprehensive analysis of the existing literature.

Declarations

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Authors' Contributions

CK contributed to the development of the review, while HSL and JMS assisted with the review process. All authors have read and approved the final manuscript.

Competing interests

The authors declare no conflicts of interest.

Availability of data and materials

No datasets or additional materials were generated or made available in the course of this research.

References

Aafaq, N., Mian, A., Liu, W., Gilani, S. Z., and Shah, M. (2019). Video description: A survey of methods, datasets, and evaluation metrics. *ACM Computing Surveys*, 52(6):1–37. DOI: 10.1145/3355390.

Agarwal, N. and Netravali, R. (2023). Boggart: Towards General-Purpose acceleration of retrospective video analytics. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 933–951, Boston, MA. USENIX Association. Available at:`https://www.usenix.org/conference/nsdi23/presentation/agarwal-neil`.

Alam, A., Khan, M. N., Khan, J., and Lee, Y.-K. (2020a). Intellibvr-intelligent large-scale video retrieval for objects and events utilizing distributed deep-learning and semantic approaches. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 28–35. IEEE. DOI: 10.1109/BigComp48618.2020.0-103.

Alam, A., Ullah, I., and Lee, Y.-K. (2020b). Video big data analytics in the cloud: A reference architecture, survey, opportunities, and open research issues. *IEEE Access*, 8:152377–152422. DOI: 10.1109/access.2020.3017135.

Anderson, M. R., Cafarella, M., Ros, G., and Wenisch, T. F. (2019). Physical representation-based predicate optimization for a visual analytics database. In *Proceedings of the International Conference on Data Engineering*, volume 2019-April, pages 1466–1477. DOI: 10.1109/icde.2019.00132.

Andreas Meier, M. K. (2019). Sql & nosql databases: Models, languages, consistency options and architectures for big data management. Springer. DOI: 10.1007/978-3-658-24549-8.

Bastani, F., He, S., Balasingam, A., Gopalakrishnan, K., Alizadeh, M., Balakrishnan, H., Cafarella, M., Kraska, T., and Madden, S. (2020). Miris: fast object track queries in video. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 1907–1921. DOI: 10.1145/3318464.3389692.

Bochkovskiy, A., Wang, C., and Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934. DOI: 10.48550/arXiv.2004.10934.

Caltech (2016). Computational vision at caltech. Availabe at:`https://www.vision.caltech.edu/`.

Canel, C., Kim, T., Zhou, G., Li, C., Lim, H., Andersen, D. G., Kaminsky, M., and Dulloor, S. (2019). Scaling video analytics on constrained edge nodes. In Talwalkar, A., Smith, V., and Zaharia, M., editors, *Proceedings of Machine Learning and Systems*, volume 1, pages 406–417. Available at:`https://proceedings.mlsys.org/paper_files/paper/2019/hash/6bcfac823d40046dca25ef6d6d59cc3f-Abstract.html`.

Chao, D., Chen, K., and Koudas, N. (2023). SVQ-ACT: Querying for Actions over Videos. In *IEEE 39th International Conference on Data Engineering (ICDE)*, pages 3599–3602. DOI: 10.1109/ICDE55515.2023.00277.

Chao, D., Koudas, N., and Xarchakos, I. (2020). SVQ++: Querying for Object Interactions in Video Streams. In *Pro-*

ceedings of the 2020 ACM SIGMOD International Conference on Management of Data, SIGMOD '20, page 2769–2772, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3318464.3384701.

Chaudhary, S., Taneja, A., Singh, A., Roy, P., Sikdar, S., Maity, M., and Bhattacharya, A. (2024). {TileClipper}: Lightweight selection of regions of interest from videos for traffic surveillance. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 967–984. Available at:`https://www.usenix.org/conference/atc24/presentation/chaudhary`.

Chen, T. Y. H., Ravindranath, L., Deng, S., Bahl, P., and Balakrishnan, H. (2015). Glimpse: Continuous, real-time object recognition on mobile devices. In *Proc. of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168. DOI: 10.1145/2809695.2809711.

Chen, Y., Yu, X., and Koudas, N. (2022). Ranked window query retrieval over video repositories. In *IEEE International Conference on Data Engineering*, pages 2776–2791. DOI: 10.1109/ICDE53745.2022.00253.

Chen, Y., Yu, X., Koudas, N., and Yu, Z. (2021). Evaluating temporal queries over video feeds. In *Proc. of the 2021 International Conference on Management of Data*, pages 287–299. DOI: 10.1145/3448016.3452803.

Chollet, F. (2021). *Deep learning with Python*. Simon and Schuster, Shelter Island, NY, USA. Book.

Chunduri, P., Bang, J., Lu, Y., and Arulraj, J. (2022). Zeus: Efficiently localizing actions in videos using reinforcement learning. In *Proc. of the 2022 International Conference on Management of Data*, page 545–558, New York, NY, USA. ACM. DOI: 10.1145/3514221.3526181.

Collins, Z. (2020). Active database interface for video search. Master of engineering thesis, Massachusetts Institute of Technology. Available at:`https://dspace.mit.edu/handle/1721.1/127391`.

Cugola, G. and Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3):1–62. DOI: 10.1145/2187671.2187677.

da Silveira, T. B. N. (2023). *Semantic-Related Challenges in Computational Intelligence: a Transdisciplinary Approach*. Phd thesis, Universidade Tecnológica Federal do Paraná, Curitiba, PR, Brazil. Available at:`https://repositorio.utfpr.edu.br/jspui/handle/1/33531`.

Dai, X., Yang, P., Zhang, X., Dai, Z., and Yu, L. (2022). Respire: Reducing spatial–temporal redundancy for efficient edge-based industrial video analytics. *IEEE Transactions on Industrial Informatics*, 18(12):9324–9334. DOI: 10.1109/TII.2022.3162598.

Dai, X., Zhang, Z., Yang, P., Xu, Y., Liu, X., and Lui, J. C. (2024). Axiomvision: Accuracy-guaranteed adaptive visual model selection for perspective-aware video analytics. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 7229–7238, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3664647.3681269.

Daum, M., Haynes, B., He, D., Mazumdar, A., and Balazinska, M. (2021). TASM: A tile-based storage manager for video analytics. In *Proc. of IEEE 37th International Conference on Data Engineering*, pages 1775–1786. DOI: 10.1109/icde51399.2021.00156.

de Boer, M. H. T., Escher, C., and Schutte, K. (2017). Modelling temporal structures in video event retrieval using an AND-OR graph. In *Proc. of the Ninth International Conferences on Advances in Multimedia*, pages 85–88, Venice, Italy. Available at:`https://repository.ubn.ru.nl/bitstream/handle/2066/182006/182006.pdf`.

de Lausanne, V. (2020). Place de la palud. Available at:`https://www.youtube.com/watch?v=GbAZX-NDPLg`, Date: 2023-07-01.

Dong, S., Wang, P., and Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Reviews*, 40(C). DOI: 10.1016/j.cosrev.2021.100379.

Du, K., Pervaiz, A., Yuan, X., Chowdhery, A., Zhang, Q., Hoffmann, H., and Jiang, J. (2020). Server-driven video streaming for deep learning inference. In *Proc. of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, technologies, architectures, and Protocols for Computer Communication*, pages 557–570. DOI: 10.1145/3387514.3405887.

Farhadi, A. and Redmon, J. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 1804:1–6. DOI: 10.48550/arxiv.1804.02767.

Fu, D. Y., Crichton, W., Hong, J., Yao, X., Zhang, H., Truong, A., Narayan, A., Agrawala, M., Ré, C., and Fatahalian, K. (2019). Rekall: Specifying video events using compositions of spatiotemporal labels. *CoRR*, abs/1910.02993. DOI: https://doi.org/10.48550/arXiv.1910.02993.

Furht, B. and Villanustre, F. (2016). Introduction to big data. *Big data technologies and applications*, pages 3–11. DOI: $10.1007/978\text{-}3\text{-}319\text{-}44550\text{-}2_1$.

Giatrakos, N., Alevizos, E., Artikis, A., Deligiannakis, A., and Garofalakis, M. (2020). Complex event recognition in the big data era: a survey. *The VLDB Journal*, 29(1):313–352. DOI: 10.1007/s00778-019-00557-w.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. DOI: 10.1038/nature14539.

Grulich, P. M. and Nawab, F. (2018). Collaborative edge and cloud neural networks for real-time video processing. *Proceedings of the VLDB Endowment*, 11(12):2046–2049. DOI: 10.14778/3229863.3236256.

Guo, H., Yao, S., Yang, Z., Zhou, Q., and Nahrstedt, K. (2021). Crossroi: cross-camera region of interest optimization for efficient real time video analytics at scale. In *Proceedings of the 12th ACM Multimedia Systems Conference*, MMSys '21, page 186–199, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3458305.3463381.

Gutoski, M., Lazzaretti, A. E., and Lopes, H. S. (2021). Deep metric learning for open-set human action recognition in videos. *Neural Computing and Applications*, 33:1207–1220. DOI: 10.1007/s00521-020-05009-z.

Han, S., Shen, H., Philipose, M., Agarwal, S., Wolman, A., and Krishnamurthy, A. (2016). MCDNN: An approximation-based execution framework for deep stream processing under resource constraints. In *Proc.*

of the 14th Annual International Conference on Mobile Systems, Applications, and Services, pages 123–136, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2906388.2906396.

Haynes, B., Daum, M., He, D., Mazumdar, A., Balazinska, M., Cheung, A., and Ceze, L. (2021). Vss: A storage system for video analytics. In *Proc. of International Conference on Management of Data*, SIGMOD '21, pages 685–696, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3448016.3459242.

Haynes, B., Daum, M., Mazumdar, A., Balazinska, M., Cheung, A., and Ceze, L. (2020). VisualWorldDB: A DBMS for the Visual World. In *Proc. of Conference on Innovative Data Systems Research*. Available at:https://par.nsf.gov/biblio/10257106.

Haynes, B., Mazumdar, A., Alaghi, A., Balazinska, M., Ceze, L., and Cheung, A. (2018). LightDB: A DBMS for Virtual Reality Video. *Proceedings of the VLDB Endowment*, 11(10):1192–1205. Available at:https://par.nsf.gov/biblio/10104525.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969. DOI: 10.1109/iccv.2017.322.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. DOI: 10.1109/cvpr.2016.90.

Hirsch, J. and Buela-Casal, G. (2014). The meaning of the h-index. *International Journal of Clinical and Health Psychology*, 14(2):161–164. DOI: 10.1016/S1697-2600(14)70050-X.

Hole, S. J. (2018). Jackson hole, wyoming – town square live cam. Available at:https://www.youtube.com/watch?v=1EiC9bvVGnk.

Honarparvar, S., Ashena, Z. B., Saeedi, S., and Liang, S. (2024). A systematic review of event-matching methods for complex event detection in video streams. *Sensors*, 24(22). DOI: 10.3390/s24227238.

HÖnig, R., Ackermann, J., and Chi, M. (2023). Bi-encoder cascades for efficient image search. In *IEEE/CVF International Conference on Computer Vision Workshops*, pages 1350–1355. Available at:https://openaccess.thecvf.com/content/ICCV2023W/RCV/html/Honig_Bi-Encoder_Cascades_for_Efficient_Image_Search_ICCVW_2023_paper.html.

Hsieh, K. (2019). *Machine Learning Systems for Highly-Distributed and Rapidly-Growing Data*. PhD thesis, Pittsburgh, USA. Available at:https://pdfs.semanticscholar.org/231b/ab35b215632fb343180075d1fab937f72384.pdf.

Hsieh, K., Ananthanarayanan, G., Bodik, P., Venkataraman, S., Bahl, P., Philipose, M., Gibbons, P. B., and Mutlu, O. (2018). Focus: Querying large video datasets with low latency and low cost. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation*, pages 269–286, Carlsbad, CA, USA. Available at:https://www.usenix.org/conference/osdi18/presentation/hsieh.

Hung, C. C., Ananthanarayanan, G., Bodik, P., Golubchik, L., Yu, M., Bahl, P., and Philipose, M. (2018). VideoEdge: Processing camera streams using hierarchical clusters. In *Proceedings - 2018 3rd ACM/IEEE Symposium on Edge Computing, SEC 2018*, pages 115–131. IEEE. DOI: 10.1109/sec.2018.00016.

Hwang, J., Kim, M., Kim, D., Nam, S., Kim, Y., Kim, D., Sharma, H., and Park, J. (2022). CoVA: Exploiting Compressed-Domain analysis to accelerate video analytics. In *USENIX Annual Technical Conference*, pages 707–722. USENIX Association. Available at:https://www.usenix.org/conference/atc22/presentation/hwang.

Inacio, A. S., Gutoski, M., Lazzaretti, A. E., and Lopes, H. S. (2021). OSVidCap: a framework for the simultaneous recognition and description of concurrent actions in videos in an open-set scenario. *IEEE Access*, 9:137029–137041. DOI: 10.1109/ACCESS.2021.3116882.

Jain, S., Ananthanarayanan, G., Jiang, J., Shu, Y., and Gonzalez, J. (2019). Scaling video analytics systems to large camera deployments. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, HotMobile '19, page 9–14, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3301293.3302366.

Jain, S., Zhang, X., Zhou, Y., Ananthanarayanan, G., Jiang, J., Shu, Y., Bahl, P., and Gonzalez, J. (2020). Spatula: Efficient cross-camera video analytics on large camera networks. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 110–124. DOI: 10.1109/SEC50012.2020.00016.

Jiang, J., Ananthanarayanan, G., Bodik, P., *et al*. (2018). Chameleon: Scalable adaptation of video analytics. In *Proceedings of Conference of the ACM Special Interest Group on Data Communication*, pages 253–266. DOI: 10.1145/3230543.3230574.

Jodoin, J.-P., Bilodeau, G.-A., and Saunier, N. (2014). Urban tracker: Multiple object tracking in urban mixed traffic. In *Proc. of IEEE Winter Conference on Applications of Computer Vision*, pages 885–892. DOI: 10.1109/wacv.2014.6836010.

Kabukicho, S. (2020). Shinjuku kabukicho live camera. Available at:https://www.youtube.com/watch?v=EHkMjfMw7oU.

Kakkar, G. T., Cao, J., Chunduri, P., *et al*. (2023). Eva: An end-to-end exploratory video analytics system. In *Proceedings of the Seventh Workshop on Data Management for End-to-End Machine Learning*, New York, NY, USA. ACM. DOI: 10.1145/3595360.3595858.

Kang, D. (2022). *Efficient and Accurate Systems for Querying Unstructured Data*. Phd thesis, Stanford University, Palo Alto, USA. Available at:https://www.proquest.com/openview/9ddd50ee37cee3075c0cb56e97becd4f/1?pq-origsite=gscholar&cbl=18750&diss=y.

Kang, D., Bailis, P., and Zaharia, M. (2019a). Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics. *Pro-*

ceedings of VLDB Endowment, 13(4):533–546. DOI: 10.48550/arxiv.1805.01046.

Kang, D., Bailis, P., and Zaharia, M. (2019b). Challenges and opportunities in DNN-based video analytics: A demonstration of the blazeit video query engine. In *Proceedings of the 9th Biennial Conference on Innovative Data Systems Research*, Asilomar, California, USA. Available at:https://www.cs.purdue.edu/homes/bb/2020-fall-cs590bb/docs/at/ml/paper_Challenges_and_Opportunities_in_DNN-Based_Video_Analytics.pdf.

Kang, D., Emmons, J., Abuzaid, F., Bailis, P., and Zaharia, M. (2017). Noscope: Optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597. DOI: 10.48550/arxiv.1703.02529.

Kang, D., Gan, E., Bailis, P., Hashimoto, T., and Zaharia, M. (2020a). Approximate selection with guarantees using proxies. *Proceedings of the VLDB Endowment*, 13(12):1990–2003. DOI: 10.14778/3407790.3407804.

Kang, D., Guibas, J., Bailis, P., Hashimoto, T., Sun, Y., and Zaharia, M. (2021). Accelerating approximate aggregation queries with expensive predicates. *Proceedings of the VLDB Endowment*, 14(11):2341–2354. DOI: 10.14778/3476249.3476285.

Kang, D., Mathur, A., Veeramacheneni, T., Bailis, P., and Zaharia, M. (2020b). Jointly optimizing preprocessing and inference for dnn-based visual analytics. volume 14, pages 87–100. DOI: 10.14778/3425879.3425881.

Kang, D., Romero, F., Bailis, P., Kozyrakis, C., and Zaharia, M. (2022). VIVA: An end-to-end system for interactive video analytics. In *Proceedings of the 12th Conference on Innovative Data Systems Research (CIDR)*, Chaminade, USA. Available at:https://info290.github.io/papers/viva.pdf.

Khani, M., Ananthanarayanan, G., Hsieh, K., Jiang, J., Netravali, R., Shu, Y., Alizadeh, M., and Bahl, V. (2023). RECL: Responsive Resource-Efficient continuous learning for video analytics. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 917–932, Boston, MA. USENIX Association. Available at:https://www.usenix.org/conference/nsdi23/presentation/khani.

Kossoski, C. (2024). *NOP Query: a new notification-based method for processing video queries on the fly*. PhD thesis, Federal University of Technology - Paraná (UTFPR), Curitiba, PR, Brazil. Available at:https://repositorio.utfpr.edu.br/jspui/bitstream/1/35731/1/nopquerymethod.pdf.

Kossoski, C., Simão, J. M., and Lopes, H. S. (2024). Modeling and performance analysis of a notification-based method for processing video queries on the fly. *Applied Sciences*, 14(9):3566. DOI: 10.3390/app14093566.

Koudas, N., Li, R., and Xarchakos, I. (2022). Video monitoring queries. *IEEE Transactions on Knowledge and Data Engineering*, 34(10):5023–5036. DOI: 10.1109/icde48307.2020.00115.

Kraft, P., Kang, D., Narayanan, D., Palkar, S., Bailis, P., and Zaharia, M. (2020). A demonstration of willump: A statistically-aware end-to-end optimizer for machine learning inference. *Proceedings of the VLDB Endowment*, 13(12):2833–2836. DOI: 10.14778/3415478.3415487.

Krishnan, S., Dziedzic, A., and Elmore, A. J. (2018). Deeplens: Towards a visual data management system. In *Proc. of 9th Biennial Conference on Innovative Data Systems Research*, pages 1–10. DOI: 10.48550/arXiv.1812.07607.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25. DOI: 10.1145/3065386.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. (2011). HMDB: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE. DOI: 10.1109/iccv.2011.6126543.

Lai, Z., Han, C., Liu, C., Zhang, P., Lo, E., and Kao, B. (2021). Top-k deep video analytics: A probabilistic approach. SIGMOD '21, page 1037–1050, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3448016.3452786.

Laskaridis, S., Venieris, S. I., Almeida, M., Leontiadis, I., and Lane, N. D. (2020). Spinn: synergistic progressive inference of neural networks over device and cloud. In *Proc. of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–15. DOI: 10.1145/3372224.3419194.

Latif, A., Rasheed, A., Sajid, U., Ahmed, J., Ali, N., Ratyal, N. I., Zafar, B., Dar, S. H., Sajid, M., and Khalil, T. (2019). Content-based image retrieval and feature extraction: a comprehensive review. *Mathematical Problems in Engineering*, 2019. DOI: 10.1155/2019/9658350.

Lemmer, W. (2019). Binnenhaven lemmer. Available at:https://www.youtube.com/watch?v=NyzxJMWxDeo, Date: 2023-07-01.

Li, J., Liu, L., Xu, H., Wu, S., and Xue, C. J. (2023a). Cross-camera inference on the constrained edge. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10. DOI: 10.1109/INFOCOM53939.2023.10229045.

Li, J. Z., Ozsu, M. T., Szafron, D., and Oria, V. (1997). MOQL: A multimedia object query language. In *Proc. of the 3rd International Workshop on Multimedia Information Systems*, pages 19–28. Available at:https://www.researchgate.net/publication/2438308_MOQL_A_Multimedia_Object_Query_Language.

Li, Y., Padmanabhan, A., Zhao, P., Wang, Y., Xu, G. H., and Netravali, R. (2020a). Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 359–376, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3387514.3405874.

Li, Z., Katsifodimos, A., Bozzon, A., and Houben, G. J. (2020b). Complex event processing on real-time video streams. In *CEUR Workshop Proceedings*, page 2652. Available at:https:

//repository.tudelft.nl/file/File_3cd89043-eddc-4a95-aa49-2d5eae0f7c51?preview=1.

Li, Z., Schönfeld, M., Hai, R., Bozzon, A., and Katsifodimos, A. (2023b). Optimizing machine learning inference queries for multiple objectives. In *2023 IEEE 39th International Conference on Data Engineering Workshops (ICDEW)*, pages 74–78. DOI: 10.1109/ICDEW58674.2023.00017.

Liu, M., Wang, X., Nie, L., Tian, Q., Chen, B., and Chua, T.-S. (2018). Cross-modal moment localization in videos. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 843–851, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3240508.3240549.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science*, 9905:21–37. DOI: 10.1007/978-3-319-46448-0_2.

Liu, X., Ghosh, P., Ulutan, O., Manjunath, B. S., Chan, K., and Govindan, R. (2019). Caesar: cross-camera complex activity recognition. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, SenSys '19, page 232–244, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3356250.3360041.

Lu, C., Liu, M., and Wu, Z. (2015). SVQL: A SQL extended query language for video databases. *International Journal of Database Theory and Application*, 8(3):235–248. Available at:https://www.earticle.net/Article/A249629.

Lu, Y., Chowdhery, A., and Kandula, S. (2016). Optasia: A Relational Platform for Efficient Large-Scale Video Analytics. SoCC '16, page 57–70, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2987550.2987564.

Lu, Y., Chowdhery, A., Kandula, S., and Chaudhuri, S. (2018). Accelerating machine learning inference with probabilistic predicates. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1493–1508, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3183713.3183751.

Lv, X., Wang, Q., Yu, C., and Jin, H. (2023). A feedback-driven dnn inference acceleration system for edge-assisted video analytics. *IEEE Transactions on Computers*, 72(10):2902–2912. DOI: 10.1109/TC.2023.3275094.

Madden, S., Cafarella, M., Franklin, M., and Kraska, T. (2024). Databases unbound: Querying all of the world's bytes with ai. *Proc. VLDB Endow.*, 17(12):4546–4554. DOI: 10.14778/3685800.3685916.

Mao, H., Kong, T., *et al*. (2019). CaTDet: Cascaded tracked detector for efficient object detection from video. In *Proc. of Conference on Machine Learning and Systems*, volume 1, pages 201–211. Available at:https://proceedings.mlsys.org/paper_files/paper/2019/hash/146b291bfc50fb43464b8ef8b1fea5f0-Abstract.html.

Moll, O., Bastani, F., Madden, S., Stonebraker, M., Gadepally, V., and Kraska, T. (2022). Exsample: Efficient searches on video repositories through adaptive sampling. In *Proc. of IEEE 38th International Conference on Data Engineering (ICDE)*, pages 3065–3077. DOI: 10.1109/ICDE53745.2022.00266.

MOT2016 (2022). Multiple object tracking benchmark. Available at:https://motchallenge.net/data/MOT16/.

Mullapudi, R. T., Chen, S., Zhang, K., Ramanan, D., and Fatahalian, K. (2019). Online model distillation for efficient video inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3573–3582. DOI: 10.1109/iccv.2019.00367.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *27th International Confer- ence on Machine Learning (ICML)*, Haifa, Israel. Available at:https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf.

National Institute of Standards and Technology (2016). Trec video retrieval evaluation (trecvid). Available at:https://www-nlpir.nist.gov/projects/tv2016/tv2016.html.

Neves, F. S. (2021). Framework PON C++ 4.0: Contribuição para concepção de aplicações no paradigma orientado a notificações por meio de programação genérica [in portuguese]. Master's thesis, Federal University of Technology - Paraná (UTFPR), Curitiba, PR, Brazil. Available at:https://repositorio.utfpr.edu.br/jspui/handle/1/26270.

Nguyen-Duc, M., Le-Tuan, A., Hauswirth, M., and Le-Phuoc, D. (2021). Towards autonomous semantic stream fusion for distributed video streams. In *Proc. of the 15th ACM International Conference on Distributed and Event-based Systems*, pages 172–175. DOI: 10.1145/3465480.3467837.

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 25. Determination Press, San Francisco, CA, USA. Available at:http://neuralnetworksanddeeplearning.com/about.html.

of Auburn, C. (2023). Toomer's corner webcam 1. Available at:https://www.youtube.com/watch?v=m_oIiAGXj1g, Date: 2023-07-01.

Ogle, V. E. and Stonebraker, M. (1995). Chabot: Retrieval from a Relational Database of Images. *Computer*, 28(9):40–48. DOI: 10.1109/2.410150.

Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C.-C., Lee, J. T., Mukherjee, S., Aggarwal, J. K., Lee, H., Davis, L., and others (2011). A large-scale benchmark dataset for event recognition in surveillance video. In *Proc. of IEEE Computer Vision and Pattern Recognition Conference*, pages 3153–3160. DOI: 10.1109/cvpr.2011.5995586.

Olatunji, I. E. and Cheng, C.-H. (2019). *Video analytics for visual surveillance and applications: An overview and survey*, pages 475–515. Springer. DOI: 10.1007/978-3-030-15628-$2_1$5.

Oussous, A., Benjelloun, F.-Z., Lahcen, A. A., and Belfkih, S. (2018). Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, 30(4):431–448. DOI: 10.1016/j.jksuci.2017.06.001.

Pagani, R. N., Kovaleski, J. L., and Resende, L. M. (2015). Methodi Ordinatio: a proposed methodology to select and

rank relevant scientific papers encompassing the impact factor, number of citation, and year of publication. *Scientometrics*, 105(3):2109–2135. DOI: 10.1007/s11192-015-1744-x.

Pakha, C., Chowdhery, A., and Jiang, J. (2018). Reinventing video streaming for distributed vision analytics. In *Proceedings of 10th USENIX Workshop on Hot Topics in Cloud Computing*, Boston, MA, USA. USENIX Association. Available at:`https://www.usenix.org/conference/hotcloud18/presentation/pakha`.

Patino, L., Cane, T., Vallee, A., and Ferryman, J. (2016). Pets 2016: Dataset and challenge. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. DOI: 10.1109/cvprw.2016.157.

Pexels (2021). Pexels - free professional videos. Available at:`https://www.pexels.com/`.

Poms, A., Crichton, W., Hanrahan, P., and Fatahalian, K. (2018a). Scanner: Efficient video analysis at scale. *ACM Transactions on Graphics*, 37(4):1–13. DOI: 10.1145/3197517.3201394.

Poms, A., Crichton, W., Hanrahan, P., and Fatahalian, K. (2018b). Scanner: Efficient video analysis at scale. *ACM Transactions on Graphics*, 37(4). DOI: 10.1145/3197517.3201394.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C., and Iyengar, S. S. (2018a). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys*, 51(5). DOI: 10.1145/3234150.

Pouyanfar, S., Yang, Y., Chen, S.-C., Shyu, M.-L., and Iyengar, S. S. (2018b). Multimedia big data analytics: A survey. *ACM Computing Surveys*, 51(1). DOI: 10.1145/3150226.

Punchihewa, A. and Bailey, D. (2020). A review of emerging video codecs: Challenges and opportunities. In *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6, Wellington, New Zealand. IEEE. DOI: 10.1109/ivcnz51579.2020.9290536.

Qin, A., Xiao, M., Wu, Y., Huang, X., and Zhang, X. (2021). Mixer: efficiently understanding and retrieving visual content at web-scale. *Proceedings of the VLDB Endowment*, 14(12):2906–2917. DOI: 10.14778/3476311.3476371.

Rahmanian, A., Ali-Eldin, A., Tesfatsion, S. K., Skubic, B., Gustafsson, H., Shenoy, P., and Elmroth, E. (2023). Ravas: Interference-aware model selection and resource allocation for live edge video analytics. In *2023 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 27–39. DOI: 10.1145/3583740.3628443.

Rahmanian, A., Amin, S., Gustafsson, H., and Ali-Eldin, A. (2024). Cvf: Cross-video filtration on the edge. In *Proceedings of the 15th ACM Multimedia Systems Conference*, MMSys '24, page 231–242, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3625468.3647627.

Ramachandra, B. and Jones, M. (2020). Street Scene: A new dataset and evaluation protocol for video anomaly detection. In *Proc. of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2569–2578. DOI: 10.1109/wacv45572.2020.9093457.

Ran, X., Chen, H., Zhu, X., Liu, Z., and Chen, J. (2018). Deepdecision: A mobile deep learning framework for edge video analytics. In *Proceedings of the IEEE Conference on Computer Communications*, pages 1421–1429. DOI: 10.1109/infocom.2018.8485905.

Ristani, E., Solera, F., Zou, R., Cucchiara, R., and Tomasi, C. (2016). Performance measures and a data set for multi-target, multi-camera tracking. In *Proc. of European Conference on Computer Vision Workshop on Benchmarking Multi-Target Tracking*, Cham, Germany. DOI: 10.1007/978-3-319-48881-3_2.

Romero, F., Hauswald, J., Partap, A., Kang, D., Zaharia, M., and Kozyrakis, C. (2022). Optimizing video analytics with declarative model relationships. *Proceedings of the VLDB Endowment*, 16(3):447–460. DOI: 10.14778/3570690.3570695.

Rosebrock, A. (2016). *Practical python and OpenCV – An introductory, example driven guide to image processing and computer vision*. Pyimagesearch, Ebook. Book.

Rosebroke, A. (2017). *Deep Learning for Computer Vision with Python*. PyImageSearch, Ebook. Book.

Ryoo, M. S., Chen, C.-C., Aggarwal, J. K., and Roy-Chowdhury, A. (2010). An overview of contest on semantic description of human activities (SDHA) 2010. In *International Conference on Pattern Recognition*, pages 270–285. Springer. DOI: 10.1007/978-3-642-17711-8_28.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4510–4520. DOI: 10.1109/cvpr.2018.00474.

School, O. M. (2018). Webcam from the oxford martin school on broad street. Available at:`https://www.youtube.com/watch?v=St7aTfoIdYQ`.

Shen, H., Han, S., Philipose, M., and Krishnamurthy, A. (2017). Fast video classification via adaptive cascading of deep models. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3646–3654. IEEE. DOI: 10.1109/cvpr.2017.236.

Shen, H., Philipose, M., Agarwal, S., and Wolman, A. (2014). MCDNN: An Execution Framework for Deep Neural Networks on Resource-Constrained Devices. In *Proc. of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, number December, pages 123–136. Available at:`https://www.microsoft.com/en-us/research/wp-content/uploads/2015/12/mcdnn_tr.pdf`.

Sipser, A. (2020). Video ingress system for surveillance video querying. Master of engineering thesis, Massachusetts Institute of Technology. Available at:`https://dspace.mit.edu/handle/1721.1/127525`, Date: 2023-07-01.

Soomro, K., Zamir, A. R., and Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. *ArXiv preprint*. DOI: 10.48550/arXiv.1212.0402.

Spolaôr, N., Lee, H. D., Takaki, W. S. R., Ensina, L. A., Coy, C. S. R., and Wu, F. C. (2020). A systematic review on content-based video retrieval. *Engineering*

Applications of Artificial Intelligence, 90:103557. DOI: 10.1016/j.engappai.2020.103557.

Stonebraker, M., Bhargava, B., Cafarella, M., Collins, Z., McClellan, J., Sipser, A., Sun, T., Nesen, A., Solaiman, K., Mani, G., *et al*. (2020). Surveillance video querying with a human-in-the-loop. In *Proceedings of the Workshop on Human-in-the-Loop Data Analytics*, pages 14–19, Portland, OR, USA. Available at:`https://openreview.net/pdf?id=hZoEgNmzGP`, Date: 2023-07-01.

Sun, L., Wang, W., Yuan, T., Mi, L., Dai, H., Liu, Y., and Fu, X. (2024). Biswift: Bandwidth orchestrator for multi-stream video analytics on edge. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, pages 1181–1190. DOI: 10.1109/INFOCOM52122.2024.10621392.

Technologies, S. (2019). Netcamlive 2 taiwan new taipei city 720p. Available at:`https://www.youtube.com/watch?v=INR-B7FwhS8`.

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., and Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459. Available at:`https://openaccess.thecvf.com/content_cvpr_2018/html/Tran_A_Closer_Look_CVPR_2018_paper.html`.

Usman, M., Jan, M. A., He, X., and Chen, J. (2019). A survey on big multimedia data processing and management in smart cities. *ACM Computing Surveys*, 52(3):1–29. DOI: 10.1145/3323334.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2021). Scaled-yolov4: Scaling cross stage partial network. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038. IEEE. DOI: 10.1109/cvpr46437.2021.01283.

Wang, L., Lu, K., Zhang, N., Qu, X., Wang, J., Wan, J., Li, G., and Xiao, J. (2023). Shoggoth: Towards efficient edge-cloud collaborative real-time video inference via adaptive online learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. DOI: 10.1109/DAC56929.2023.10247821.

Wang, L., Qu, X., Wang, J., Li, G., Wan, J., Zhang, N., Guo, S., and Xiao, J. (2024). Gecko: Resource-efficient and accurate queries in real-time video streams at the edge. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, pages 481–490. DOI: 10.1109/INFOCOM52122.2024.10621399.

Wang, W., Gao, J., Zhang, M., Wang, S., Chen, G., Ng, T. K., Ooi, B. C., Shao, J., and Reyad, M. (2018). Rafiki: Machine learning as an analytics service system. *Proceedings of the VLDB Endowment*, 12(2):128–140. DOI: 10.48550/arxiv.1804.06087.

Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M.-C., Qi, H., Lim, J., Yang, M.-H., and Lyu, S. (2020). UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking. *Computer Vision and Image Understanding*, 193:102907. DOI: 10.1016/j.cviu.2020.102907.

Wen, Q., Zhou, J., Chen, R., Luo, Z., Tyson, G., Li, W., Wang, J., Pan, H., and Xu, Z. (2024). From limited resources to powerful insights: Empowering low-cost cameras for efficient retrospective querying. *IEEE Internet of Things Journal*, pages 1–1. DOI: 10.1109/JIOT.2024.3480089.

Wu, D., Zhang, D., Zhang, M., Zhang, R., Wang, F., and Cui, S. (2023). Ilcas: Imitation learning-based configuration-adaptive streaming for live video analytics with cross-camera collaboration. *IEEE Transactions on Mobile Computing*, pages 1–15. DOI: 10.1109/TMC.2023.3327097.

Xarchakos, I. and Koudas, N. (2019). Svq: Streaming video queries. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 2013–2016, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3299869.3320230.

Xipg (2021). Xipg video test media. Available at:`https://media.xiph.org/video/derf/`.

Xu, R., Razavi, S., and Zheng, R. (2023). Edge video analytics: A survey on applications, systems and enabling techniques. *IEEE Communications Surveys Tutorials*, 25(4):2951–2982. DOI: 10.1109/COMST.2023.3323091.

Xu, T., Botelho, L. M., and Lin, F. X. (2019). VStore: A data store for analytics on large videos. In *Proceedings of the 14th EuroSys Conference*, pages 1–17, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3302424.3303971.

Yadav, P. (2019). High-performance complex event processing framework to detect event patterns over video streams. In *Proceedings of the 20th International Middleware Conference Doctoral Symposium*, pages 47–50, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3366624.3368169.

Yadav, P. (2021). *Query-aware adaptive windowing for spatiotemporal complex video event processing for internet of multimedia things*. PhD thesis, University of Galway, Ireland.

Yadav, P. and Curry, E. (2019a). VEKG: Video event knowledge graph to represent video streams for complex event pattern matching. In *Proceedings of the IEEE First International Conference on Graph Computing*, pages 13–20, Laguna Hills, CA, USA. IEEE. DOI: 10.1109/GC46384.2019.00011.

Yadav, P. and Curry, E. (2019b). VidCEP: Complex event processing framework to detect spatiotemporal patterns in video streams. In *Proceedings of the IEEE International Conference on Big Data*, pages 2513–2522, New York, NY, USA. Association for Computing Machinery. DOI: 10.1109/BigData47090.2019.9006018.

Yadav, P., Salwala, D., and Curry, E. (2021). Vid-win: Fast video event matching with query-aware windowing at the edge for the internet of multimedia things. *IEEE Internet of Things Journal*. DOI: 10.1109/jiot.2021.3075336.

Yadav, P., Salwala, D., Das, D. P., and Curry, E. (2020). Knowledge graph driven approach to represent video streams for spatiotemporal event pattern matching in complex event processing. *International Journal of Semantic Computing*, 14(03):423–455. DOI: 10.1142/s1793351x20500051.

Yang, K., Liu, J., Yang, D., Wang, H., Sun, P., Zhang,

Y., Liu, Y., and Song, L. (2023). A novel efficient multi-view traffic-related object detection framework. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. DOI: 10.1109/ICASSP49357.2023.10095027.

Yang, P., Lyu, F., Wu, W., Zhang, N., Yu, L., and Shen, X. S. (2020). Edge Coordinated Query Configuration for Low-Latency and Accurate Video Analytics. *IEEE Transactions on Industrial Informatics*, 16(7):4855–4864. DOI: 10.1109/tii.2019.2949347.

Yang, Z., Wang, Z., Huang, Y., Lu, Y., Li, C., and Wang, X. S. (2022). Optimizing machine learning inference queries with correlative proxy models. *Proceedings of the VLDB Endowment*, 15(10):2032–2044. DOI: 10.14778/3547305.3547310.

Yi, S., Hao, Z., Zhang, Q. Q., Zhang, Q. Q., Shi, W., and Li, Q. (2017). LAVEA: Latency-Aware video analytics on edge computing platform. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 2573–2574, New York, NY, USA. Association for Computing Machinery. DOI: 10.1109/icdcs.2017.182.

Yun, K., Honorio, J., Chattopadhyay, D., Berg, T. L., and Samaras, D. (2012). Two-person interaction detection using body-pose features and multiple instance learning. In *Proc. of IEEE Computer Vision and Pattern Recognition Conference*, pages 28–35. DOI: 10.1109/cvprw.2012.6239234.

Zhang, H., Ananthanarayanan, G., Bodik, P., Philipose, M., Bahl, P., and Freedman, M. J. (2017). Live video analytics at scale with approximation and delay-tolerance. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation*, pages 377–392, Boston, MA. USENIX Association. Available at:`https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zhang`.

Zhang, Q., Sun, H., Wu, X., and Zhong, H. (2019). Edge video analytics for public safety: A review. *Proceedings of the IEEE*, 107(8):1675–1696. DOI: 10.1109/jproc.2019.2925910.

Zhang, R.-X., Li, C., Wu, C., Huang, T., and Sun, L. (2023). Owl: A pre-and post-processing framework for video analytics in low-light surroundings. In *IEEE Conference on Computer Communications*, pages 1–10. DOI: 10.1109/INFOCOM53939.2023.10229059.

Zhang, Y. and Kumar, A. (2019). Panorama: a data system for unbounded vocabulary querying over video. *Proceedings of the VLDB Endowment*, 13(4):477–491. DOI: 10.14778/3372716.3372721.

Zhang, Y., Zhang, X., Ananthanarayanan, G., Iyer, A., Shu, Y., Bahl, V., Mao, Z. M., and Chowdhury, M. (2024). Vulcan: Automatic query planning for live ML analytics. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1385–1402, Santa Clara, CA. USENIX Association. Available at:`https://www.usenix.org/conference/nsdi24/presentation/zhang-yiwen`.

Zhen, L., Hu, P., Wang, X., and Peng, D. (2019). Deep supervised cross-modal retrieval. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10394–10403. DOI: 10.1109/cvpr.2019.01064.