Vision Paper

Towards Multimodal Stream Processing Systems

Uélison Jean Lopes dos Santos
TU Darmstadt and DFKI

Alessandro Ferri
TU Darmstadt and DFKI

Szilard Nistor
TU Darmstadt and DFKI

Riccardo Tommasini
INSA Lyon & University of Tartu

Carsten Binnig
TU Darmstadt, DFKI, and hessian.AI

Manisha Luthra
TU Darmstadt and DFKI

Abstract

In this paper, we present a vision for a new generation of multimodal streaming systems that embed MLLMs as first-class operators, enabling real-time query processing across multiple modalities. Achieving this is non-trivial: while recent work has integrated MLLMs into databases for multimodal queries, streaming systems require fundamentally different approaches due to their strict latency and throughput requirements. Our approach proposes novel optimizations at all levels, including logical, physical, and semantic query transformations that reduce model load to improve throughput while preserving accuracy. We demonstrate this with Saṃsāra, a prototype leveraging such optimizations to improve performance by an order of magnitude. Moreover, we discuss a research roadmap that outlines open research challenges for building a scalable and efficient multimodal stream processing systems.

Keywords

Stream Processing Systems, Multimodal Data Management, Adaptive Query Processing, Semantic Query Optimization, AI-Enhanced Data Systems, Emerging Data Management Systems, Research Challenges

1 Introduction

Success and Limitation of Streaming Systems. Stream processing systems (SPSs) have become fundamental in numerous industries, providing real-time data processing capabilities that power applications in finance [24], entertainment [5], healthcare [30], and the Internet-of-Things (IoT) [7]. These systems continuously ingest, process, and analyze data streams, enabling timely decision-making based on the most current information. However, despite their success, existing SPSs pose significant limitations that restrict broader applications. In particular, streaming systems today are designed to handle structured data but are not equipped to process diverse, unstructured, or multimodal data types, such as image streams from cameras or other modalities, including audio sensors.

Leveraging multimodal LLMs for Streaming. Recent advances in multimodal large language models (MLLMs) have demonstrated remarkable abilities to process and integrate data across multiple modalities, such as images, text, and audio, providing contextual understanding that spans these diverse data types. As such, it seems an appealing idea to use MLLMs as a building block within SPSs to natively support rich queries on modalities beyond structured data, enabling systems to process and interpret multimodal data streams. Such capabilities enable new applications of SPSs ranging from traffic monitoring using camera streams to sports analytics.

Towards Multimodal Stream Processing. We envision a new generation of SPSs that can seamlessly query across multiple modalities (video, audio, text) by embedding MLLMs as first-class operators in the query plan. Figure 1a illustrates an example query where a user aims to detect a stolen car at a toll booth using a camera stream. While current data systems [11, 20, 22, 27, 28] have already proposed integrating MLLMs for querying multimodal data, SPSs pose fundamentally different challenges. Streaming environments require extremely low latency and high throughput, and thus render a naive MLLM integration in which each MLLM call can incur up to seconds of latency highly impractical. Unlike databases, SPSs cannot tolerate long inference times, making efficient multimodal stream processing a highly non-trivial problem.

Our Vision. In this paper, we present our vision of how MLLMs should be combined with SPSs and use their capabilities to process modalities like images out-of-the-box, while satisfying high-throughput and low-latency demands. To enable such efficient multimodal streaming, we introduce the vision of a *super-optimizer*—a new class of optimizers designed specifically for multimodal stream processing. Unlike traditional query optimizers, a super-optimizer generates deeply optimized query plans tailored to one particular query and data stream. We argue that this super-optimization pays off, since streaming queries are long-running, and that the upfront effort yields high performance benefits during query execution. For this, a super-optimizer adds several novel optimization steps, e.g., we introduce a new phase in optimization called *semantic optimization*. In addition to logical and physical optimizations, semantic optimization rewrites plans based on a semantic understanding of data and queries to specialize the plan for a given scenario. For example, in a traffic monitoring scenario, understanding that cars appear one after another allows a streaming system to skip redundant frames and avoid unnecessary, expensive inference. Moreover, a super-optimizer employs techniques such as aggressive model specialization and pruning to reduce inference load.

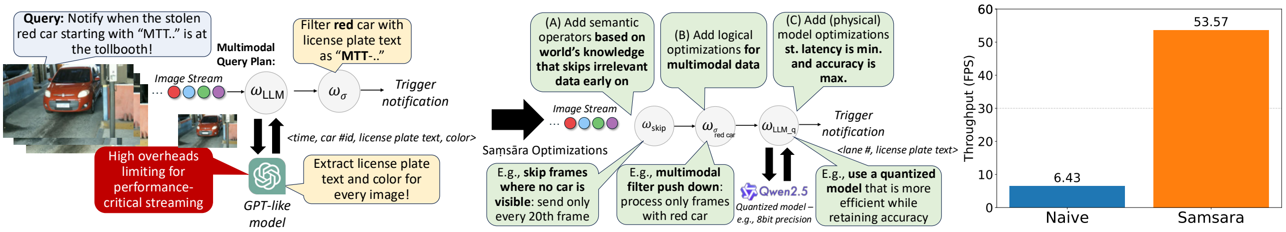
Gains and Challenges. We demonstrate these ideas in our prototype Saṃsāra[1]: a super-optimizer, which we integrated into an existing streaming system, Apache Flink [6]. In contrast to a naive evaluation in Flink without our optimizer, we can achieve substantial throughput improvements—in our example, from 6 to 53 images per second—showcasing the potential of using Saṃsāra. However, building such a *super-optimizer* to enable multimodal streaming systems that work generally for all kinds of data streams and queries is far *from trivial and requires extensive research.* In fact, building robust optimizers for classical databases has taken decades, and we believe that this paper can only be a starting point for multimodal streaming optimization powering efficient multimodal stream processing on top of MLLMs.

Outline. The remaining paper is organized as follows. Section 2 presents our vision for multimodal stream processing,

[1] Saṃsāra (Hindi: संसार) is derived from Sanskrit, meaning “the world” reflecting our view that real-world data streams are inherently multimodal and evolves.

EDBT ’26, Tampere (Finland)

10.48786/edbt.2026.51


Query: Notify when the stolen red car starting with "MTT.." is at the tollbooth!
Multimodal Query Plan:
Image Stream
ω_{LLM}
Filter red car with license plate text as "MTT-.."
ω_{σ}
Trigger notification
<time, car #id, license plate text, color>
High overheads limiting for performance-critical streaming
GPT-like model
Extract license plate text and color for every image!
(A) Add semantic operators based on world's knowledge that skips irrelevant data early on
(B) Add logical optimizations for multimodal data
(C) Add (physical) model optimizations st. latency is min. and accuracy is max.
Image Stream
Saṃsāra Optimizations
ω_{skip}
$\omega_{\sigma_{\text{red car}}}$
$\omega_{\text{LLM\_q}}$
Trigger notification
<lane #, license plate text>
E.g., skip frames where no car is visible: send only every 20th frame
E.g., multimodal filter push down: process only frames with red car
Qwen2.5
Quantized model – e.g., 8bit precision
E.g., use a quantized model that is more efficient while retaining accuracy
Throughput (FPS)
0
10
20
30
40
50
60
6.43
53.57
Naive
Samsara

(a) Example query before and after Saṃsāra optimization.

(b) Naive vs. Saṃsāra.

Figure 1: Illustrating multimodal plan optimizations proposed by Saṃsāra in (a) with evaluation results in (b). Naively integrating MLLMs into stream processing leads to extremely low performance. With Saṃsāra and its novel optimizations for multimodal data streams, we achieve up to 9× performance improvements for this example.

highlighting the intuition behind the novel optimizations. Section 3 provides a case study that illustrates these optimizations through concrete examples and explains details for the individual optimization phases. Finally, Section 4 outlines the road ahead by discussing open challenges and future directions.

2 Vision: Multimodal Stream Processing

Our vision is to enable a new generation of stream processing systems capable of understanding and querying across multiple modalities—text, images, and audio—by integrating multimodal large language models (MLLMs) as first-class operators within query plans. However, simply integrating MLLMs into query plans as done in batch-oriented data systems such as CAESURA [28], ThalamusDB [15], Lotus [22], or DocETL [26] is infeasible in a streaming context, which demands low latency and high throughput.

A Super-Optimizer for Multimodal Streaming. To overcome these challenges, we propose Saṃsāra, a novel *super-optimizer* that aggressively transforms streaming query plans over multimodal data into efficient, low-latency, and accurate execution plans. The core idea is to generate deeply optimized query plans tailored to one particular query and data stream. Since streaming queries are long-running [1], the high upfront effort for super-optimization is tolerable, and it differs significantly from traditional database optimizers, which must produce plans quickly [21]. By being able to spending more effort offline, Saṃsāra can leverage time-consuming techniques across all optimization steps as we discuss next.

Anatomy of a Super-Optimizer. We envision Saṃsāra as a rethinking of query optimization for multimodal streaming shown in Figure 2. While some phases of optimization (logical and physical) are known from classical query optimization, a super-optimizer needs to rethink these phases and even add new phases. A key innovation is a class of *semantic optimizations*, which leverage the world knowledge embedded to significantly reduce the volume of information processed by expensive MLLMs. The main idea is that semantic optimization rewrites a plan in a manner similar to how a human expert with domain understanding would approach it. For example, by reasoning about car speed, camera frame rate, and prior vehicle positions as shown in Figure 2 (left) for the traffic monitoring scenario, the optimizer can infer which upcoming frames will contain no new vehicle and skip them entirely—eliminating unnecessary inference.

Towards Semantic Query Optimization. While humans can manually identify such opportunities, our goal is to automate and generalize this process. We propose to leverage MLLMs themselves as reasoning agents within the optimizer: extracting world knowledge, inferring latent dependencies, and suggesting data-reduction transformations for arbitrary queries and datasets. This allows the optimizer to insert operators that prune redundant data and computation before they reach expensive AI operators. The central challenge lies in determining such semantic reductions automatically, without human input. As we discuss later, the Saṃsāra optimizer therefore follows a new optimization procedure where it first uses an MLLM to understand the query and data, then it selects appropriate data reduction operators from a given catalog (e.g., frame skipping) to implement the derived optimization—essentially automating what a domain expert would design manually. Finally, it applies the selected operators iteratively in the plan.

Logical & Physical Super-Optimizations. Beyond semantic optimizations, Saṃsāra applies novel logical and physical super-optimizations as shown in Figure 2 (right). For example, as in classical query optimization, Saṃsāra logically rewrites the query plan and pushes down the filter to only process red cars. The key challenge is that this filter must be inexpensive to evaluate—without invoking an MLLM—such as by applying simple computer vision methods to detect whether the image contains enough reddish pixels in our running example. Another interesting direction is to spend significantly more time on comprehensive optimizations. During physical optimization, this enables the integration of expensive techniques such as model pruning and distillation, which can drastically reduce the parameter count of MLLMs by tailoring them to a specific data stream and query. Although pruning and distillation techniques may take minutes or hours to apply, they can be executed offline before deploying the streaming query. These combined optimizations enable Saṃsāra to efficiently execute complex multimodal queries at high performance.

Why this is different from Video Analytics? The evolution of video analytics illustrates how structured processing and semantic reasoning can transform large-scale visual data into actionable insights. Early systems [4, 8, 12, 16, 17] pioneered query-driven optimization over unstructured data. These systems, however, were developed for batch-oriented / offline video processing. Our vision generalizes these principles beyond offline video analytics to continuous, multimodal data streams, integrating semantic reasoning directly into real-time query execution. This enables query answering with low-latency across multiple modalities as soon as data is generated.

3 Saṃsāra: A First Super-Optimizer

This section presents Saṃsāra—a first prototype super-optimizer for multimodal streaming systems. We discuss its design through

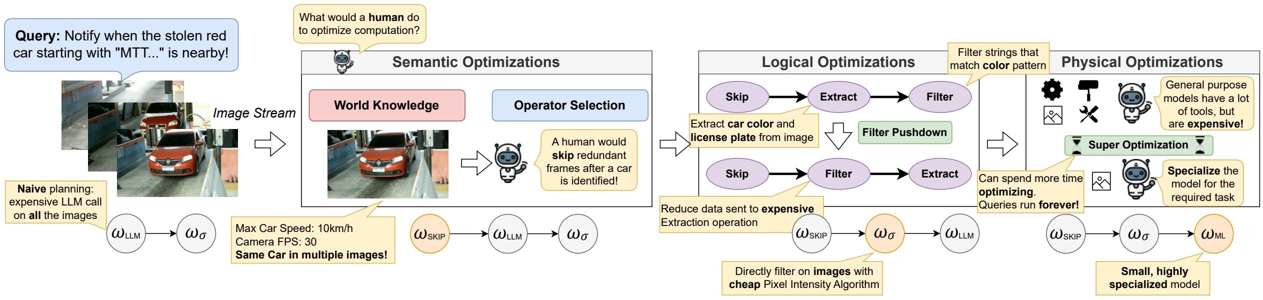

Query: Notify when the stolen red car starting with "MTT..." is nearby!
Image Stream
Naive planning: expensive LLM call on all the images
ω_{LLM}
ω_σ
What would a human do to optimize computation?
Semantic Optimizations
World Knowledge
Operator Selection
A human would skip redundant frames after a car is identified!
Max Car Speed: 10km/h
Camera FPS: 30
Same Car in multiple images!
ω_{SKIP}
ω_{LLM}
ω_σ
Logical Optimizations
Skip
Extract
Filter
Filter strings that match color pattern
Extract car color and license plate from image
Filter Pushdown
Skip
Filter
Extract
Reduce data sent to expensive Extraction operation
ω_{SKIP}
ω_σ
ω_{LLM}
Directly filter on images with cheap Pixel Intensity Algorithm
Physical Optimizations
General purpose models have a lot of tools, but are expensive!
Super Optimization
Can spend more time optimizing. Queries run forever!
Specialize the model for the required task
ω_{SKIP}
ω_σ
ω_{ML}
Small, highly specialized model

Figure 2: Overview of Saṃsāra, a super-optimizer to enable multimodal streaming systems which transforms a naive multimodal query plan into an optimized plan through a series of novel semantic, logical, and physical optimizations which aggressively optimize the plan for a particular long-running streaming query plan.

a case study that serves as a running example. Our initial evaluation extends to a broader set of queries and datasets.

3.1 Case Study: Toll Booth

We demonstrate how semantic, logical, and physical optimizations can be applied in practice using a toll booth scenario with a live camera stream. To construct the case study, we created a dataset inspired by the Linear Road Benchmark, a well-established benchmark for streaming traffic monitoring data [2]. Specifically, we incorporated images from the Rodosol-ALPR dataset [19] to generate a video stream simulating cars passing through a toll booth, with each vehicle annotated by its license plate, color, and brand. This scenario supports a variety of multimodal queries, such as filtering cars by specific vehicle attributes or counting vehicles to detect traffic patterns, and enables us to show how Saṃsāra optimizes different query types. For the purpose of the case study, we now focus on a specific query to demonstrate how Saṃsāra can optimize it using the principles of a super-optimizer. Consider the following scenario: *A car has been reported stolen, and the police want to monitor all toll booths, each equipped with cameras. The available information indicates that the car is red and its license plate begins with "MTT."* Based on this information, we can construct a naive multimodal query plan as shown in Figure 1a, which first extracts the license plate and color, and then applies filters based on the given criteria.

3.2 Super-Optimizer Design

In the following sections, we discuss using this example how Saṃsāra super-optimizes a multimodal query plan for a given data stream.

3.2.1 Semantic Optimization. Saṃsāra introduces a new optimization phase, termed *semantic query optimizations*, that leverages world knowledge and contextual reasoning to rewrite multimodal streaming query plans. The main idea is that semantic query optimizations adds query- and data-specific optimizations that require a semantic understanding of the scene. Multimodal Large language models play a central role in our framework. We leverage them throughout semantic optimization to extract semantic priors and propose valid plan rewrites.

The key challenge is to implement a general semantic optimizer in automation: how can such reasoning be systematically applied to arbitrary queries and data streams? We propose a semantic optimization procedure that takes as input (i) a data stream sample and (ii) the query plan; afterwards, Saṃsāra implements an MLLM-guided reasoning loop to identify optimizations that reduce redundant processing while preserving correctness. Figure 3 shows the stages of this process, from world-knowledge extraction to operator selection and plan rewriting, applied to the tollbooth case.

Overview of the Optimization Procedure. Our semantic optimizer uses the MLLM as a semantic reasoning engine embedded in a structured three-phase process: *(1) world-knowledge extraction*, *(2) operator selection*, and *(3) plan update*. Each phase invokes targeted LLM prompts with structured inputs—a short query description, the plan operators, and sampled data summaries—so that the model's reasoning remains grounded and verifiable.

(1) World-Knowledge Extraction. Given a query and a representative data sample from the input stream, the optimizer first invokes the MLLM to identify relevant entities, relationships, and constraints implied by both. For the tollbooth example, the query is: *"Notify when the stolen red car with plate starting with "MTT" is at the tollbooth"*. The sample consists of short video segments from a fixed camera observing the scene. From this, the MLLM extracts domain-specific priors, such as the fact that cars move approximately in a straight line through the tollbooth at bounded speeds, that empty frames frequently occur between cars, and that license plates and car colors are confined to specific spatial regions of the image. These extracted semantics extend the optimizer's reasoning context, forming a symbolic representation of the scene which guides the subsequent steps.

(2) Operator Selection. Using the extracted symbolic representation of the scene, the optimizer next invokes the MLLM to reason about which rewrites can safely reduce input volume or operator cost without affecting query correctness. In Saṃsāra, we currently implement this as a selection procedure for data reduction tools from a given tool catalog. Selecting the tools involves both *cross-frame* and *intra-frame* reasoning. In the cross-frame dimension, the MLLM infers the temporal continuity of the scene; thus, cars cannot appear or disappear instantaneously. It therefore proposes a `Skip(Amount, Condition)` operator that skips N frames after an empty detection, estimating N from metadata of the input sample, such as frame rate and maximum vehicle velocity. For example, with a frame rate of 30 FPS and $v_{max} = 30$ km/h, skipping more than three consecutive empty frames risks missing a new fast-approaching car. In the intra-frame dimension, the MLLM infers that cars predominantly appear in the lower region of the frame and that color is the only query-relevant feature. It thus proposes a `Crop(region=bottom)` operator to restrict processing spatially, and a `Downscale(resolution)` operator to reduce pixel density while preserving color fidelity. However, the MLLM explicitly rejects `Greyscale()` reduction,

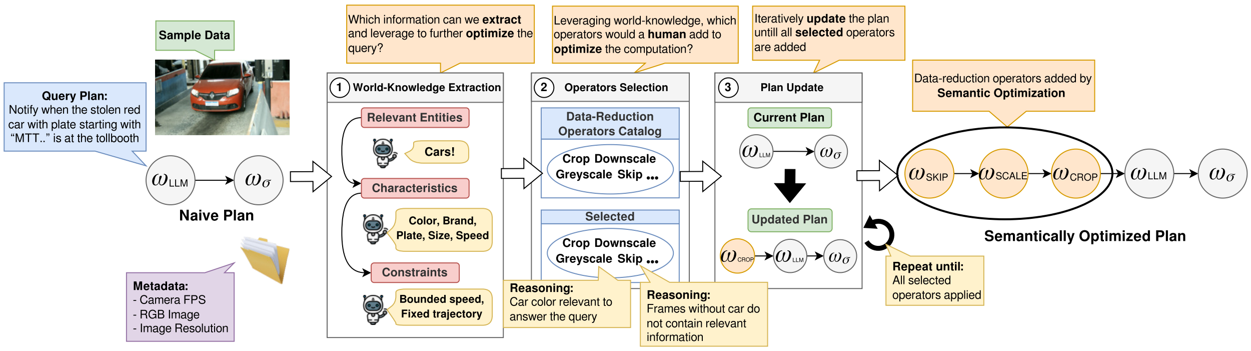

Sample Data
Query Plan:
Notify when the stolen red car with plate starting with “MTT..” is at the tollbooth
ω_{LLM}
ω_{σ}
Naive Plan
Metadata:
- Camera FPS
- RGB Image
- Image Resolution
Which information can we extract and leverage to further optimize the query?
1 World-Knowledge Extraction
Relevant Entities
Cars!
Characteristics
Color, Brand, Plate, Size, Speed
Constraints
Bounded speed, Fixed trajectory
Leveraging world-knowledge, which operators would a human add to optimize the computation?
2 Operators Selection
Data-Reduction Operators Catalog
Crop Downscale Greyscale Skip ...
Selected
Crop Downscale Greyscale Skip ...
Reasoning: Car color relevant to answer the query
Reasoning: Frames without car do not contain relevant information
Iteratively update the plan untill all selected operators are added
3 Plan Update
Current Plan
ω_{LLM}
ω_{σ}
Updated Plan
ω_{CROP}
ω_{LLM}
ω_{σ}
Repeat until: All selected operators applied
Data-reduction operators added by Semantic Optimization
ω_{SKIP}
ω_{SCALE}
ω_{CROP}
ω_{LLM}
ω_{σ}
Semantically Optimized Plan

Figure 3: Overview of the semantic optimization procedure applied to the tollbooth query. Starting from a naive plan (*Extract + Filter*), the optimizer leverages an MLLM to perform three phases: (1) *World-Knowledge Extraction*, where it identifies relevant entities (cars), their properties (color, size, speed), and real-world dynamics (limited movement through the tollbooth); (2) *Operator Selection*, where it proposes operators that reduce overhead such as `Crop`, `Downscale`, and `Skip` based on extracted knowledge and metadata; and (3) *Plan Update*, where the chosen operators are iteratively inserted to yield a semantically optimized plan (*Downscale + Crop + Extract + Filter*).

correctly reasoning that it would remove color cues critical to the query semantics. After this stage, the optimizer holds a set of validated candidate operators, each annotated with semantic preconditions.

(3) Plan Update. Finally, the optimizer integrates these operators into the query plan. Here again, the MLLM assists by reasoning about operator dependencies and insertion points. For the tollbooth query, it proposes to insert `Skip(Amount=3, Condition=no_car)` before the object detector to prune empty frames, and adds the operator `Crop(region=bottom)` before detection to restrict the spatial focus and to reduce computational overhead. The resulting plan, shown in Figure 3, demonstrates the transformation from a naive syntactic plan into a semantically optimized one that minimizes redundant processing while maintaining correctness.

Generalization to Other Queries. While we have explained the procedure for the case study query, the same reasoning loop generalizes across domains. For example, consider a sports analytics query that detects which player currently has the ball in a soccer match. The MLLM infers that ball possession transitions cannot occur instantaneously unless another player is nearby, allowing several frames to be skipped after a stable possession detection. Using the same extract–select–update framework as discussed before, the optimizer thus introduces `Skip` and `Crop` operators driven by semantic understanding rather than syntactic structure. In fact, we use this loop for all 13 queries over 2 different data streams in our initial evaluation.

Correctness of Rewrites. A core technical challenge lies in verifying the correctness of semantic rewrites. Determining whether a skip factor or downscaling level preserves query equivalence requires reasoning about both statistical fidelity and logical semantics. We propose an *empirical validation* step in which the optimizer executes both the naive and the optimized plans on a data sample and compares their outputs to estimate accuracy degradation. This feedback loop transforms Saṃsāra into a self-correcting optimizer—capable of hypothesizing, testing, and refining its own rewrites.

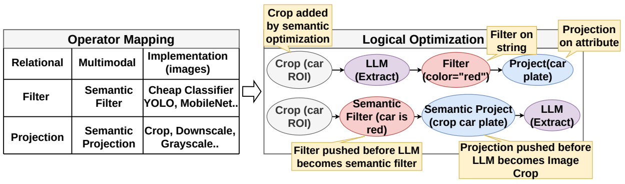


Figure 4: Logical optimization in Saṃsāra for query plans with MLLM-based operators. The upper plan shows a tollbooth query where an MLLM extracts structured attributes (car color and license plate) from images, followed by a filter on the extracted color and a projection on the license plate that cannot be directly pushed before the MLLM. To enable filter and projection pushdown, Saṃsāra rewrites structured predicates into multimodal operations. As illustrated in the lower plan, the color filter becomes a cheap classifier over images that discards irrelevant frames before costly MLLM inference, while the projection becomes a crop around the license plate to reduce the amount of data sent to the MLLM.

3.2.2 Logical Optimization. During the logical optimization phase, Saṃsāra rewrites query plans using known heuristics, such as filter pushdown and projection pushdown. For example, in our query plan for the tollbooth scenario, the operator that leverages an MLLM to extract the car color and license plate from incoming images would benefit from a filter pushdown to reduce the number of images sent to the model. To this end, Saṃsāra introduces a set of novel filter pushdown techniques that move filters normally applied after an MLLM-based operator to before the operator, by transforming structured filter predicates into the multimodal space. For example, Figure 4 shows the plan after semantic optimization of the tollbooth query that looks for a stolen red car. The MLLM-based operator is employed to extract the car color and the license plate, followed by a classic filter on strings to select license plates based on the extracted car color.

Pushing Filters from Structure to Image Domain. Unfortunately, such a filter cannot be pushed down before the MLLM-based operator, as it requires the result from the extraction (i.e., car color). To support the pushdown of such a filter, Saṃsāra translates the associated predicate from the structured domain (i.e., strings) to a filter on the image domain (or other modalities). In this case, the filter can be implemented as a similarity over

Table 1: Overview of Queries. Q1-Q9 are in the Toll Booth dataset, and Q10-Q13 on the Volleyball dataset.

Query ID	Description of Query	MLLM Tasks
Q1	Car brand recognition	Object detection
Q2	Car color recognition	Color recognition
Q3	License plate detection	Object detection, text extraction
Q4	Most popular brand & color	Color recognition, object detection, aggregation
Q5	Most popular brand	Color recognition, object detection, aggregation
Q6	Most popular color	Color recognition, aggregation
Q7	Repeated car detection	Object detection, text extraction, aggregation
Q8	Red stolen 'MTT' car	Color recognition, object detection, text extraction, filtering
Q9	Unique license plates	Object detection, text extraction, aggregation
Q10	Amount of jumping players	Action recognition, aggregation
Q11	Most offensive team	Action recognition, aggregation
Q12	Notify when someone spikes	Action recognition
Q13	3 most common actions	Action recognition, aggregation

embeddings of the image and the query or as a simpler computer vision classifier to drop frames that do not contain red cars before the MLLM invocation, as shown in Figure 4. However, naively adding such operations to drop data can result in a trade-off between accuracy and speed. To avoid this, Saṃsāra needs to be able to set confidence thresholds for multimodal filter operations in the offline phase in an optimal manner for the given query and data stream, using a sample of the data.

Beyond Filters Push-down. Furthermore, beyond filters, other operators, such as projections, can also be pushed down from after an MLLM-based operator to before, reducing the amount of data that needs to be sent to the MLLM. For example, a projection on structured data can be pushed down to the image domain by transforming it to a crop operation, which focuses on the relevant part of the image. Since in the tollbooth scenario only the car color and the license plate are required, a crop operation can be used to ignore most of the image, effectively "projecting" only the relevant parts.

Open Research Questions. These optimizations have been implemented in Saṃsāra, significantly reducing the workload of expensive plans by applying such rewrites. This translation, however, is non-trivial and hard to apply automatically. Defining a more systematic mapping from operators in the structured domain (e.g., filters, projections) to operators in the multimodal domain (crop, downscale, etc.) and automatically rewriting the plans based on this mapping, remains an open research challenge.

3.2.3 Physical Optimization. In databases, physical optimization requires selecting the most efficient algorithm to execute a logical operator. This step is important for multimodal streaming, where operators often rely on expensive AI models. Instead of always invoking a general-purpose MLLM, we can synthesize a new model tailored to the query and data at hand, enabling *super-optimization* of query execution.

Synthesizing Smaller Models. Overall, this step includes techniques such as quantization [10], pruning [9], knowledge distillation [25], and low-rank factorization [13] which are aware of query and data to compress models. In our running example, licence plate extraction from images leverages an optimally quantized MLLM. Quantization reduces the MLLM's weights and activations to 8-bit integers, halving model size and memory bandwidth.

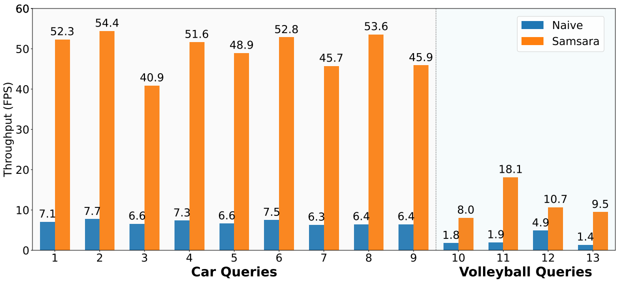


Figure 5: End-to-End gains for all(13) queries of Table 1. Saṃsāra has up to 10× speedup over naive execution

Streaming Data provides more Opportunities. Moreover, streaming data introduces new opportunities and challenges for known AI techniques such as pruning. Unlike classical AI tasks with highly diverse datasets, streaming data is often highly similar but continually evolving. We can leverage this fact for physical optimization. For example, consider model pruning. Model pruning is a technique for reducing the size of a model by removing unnecessary parameters. However, too aggressive static pruning may over-specialize and degrade performance when data characteristics change. For example, in a traffic camera scenario, data may switch from low to high density: pruning aggressively during high-traffic periods could reduce accuracy, while in low-traffic periods, it is safe. Saṃsāra addresses this with new *adaptive pruning* strategies that adjust parameters, such as pruning rates, based on data characteristics dynamically. Revisiting classical model-reduction techniques and adapting them to the uniform but evolving nature of streaming data is an interesting future direction.

Optimal Model Selection. Another key aspect of physical super-optimization is selecting an appropriate MLLM for the query and data at hand. A large MLLM can sometimes be replaced by a smaller or distilled model, balancing computational cost and accuracy. When using a smaller MLLM, optimizers must consider accuracy constraints: similar to multimodal data systems such as LOTUS [22] or Palimpzest [20], which optimize plans for batch processing, where users can specify required accuracy thresholds (e.g., 90% of the large MLLM), enabling the system to select the most efficient model that meets the accuracy threshold, we plan to apply similar ideas to stream processing. However, as data may continuously change, the model selection may need to be adapted over time as well. Building such an accuracy-guided adaptive model selection for multimodal streaming systems is another open challenge for developing a super-optimizer for them.

3.3 Initial Results

In the following, we show the initial results of using Saṃsāra for multimodal stream processing. For this, we integrated Saṃsāra with Apache Flink and executed queries without any optimizations (naive) and after optimizing plans with Saṃsāra.

Dataset and Queries. To the best of our knowledge, no benchmark currently exists for evaluating multimodal streaming queries. We therefore constructed a benchmark from existing datasets, focusing on two distinct domains that differ in complexity and data dynamics. The first dataset, *Toll Booth*, represents a static camera scenario where frames are captured from a fixed position and objects (cars) appear in predictable spatial regions. The second dataset, *Volleyball*, is derived from the Volleyball Dataset [14],

Table 2: Speedup by optimization phases of Saṃsāra over naive execution. We observe that across all queries, the minimum speedup is 2×. On average, we see around 6× speedup, and at a maximum, 10× speedup. We also observe that all phases are crucial for achieving these speedups, whereas semantic optimization is particularly important.

	Min	Avg	Max
Semantic	1.9×	4.8×	8.0×
+Logical	2.1×	7.3×	10.1×
+Physical	2.3×	7.4×	10.4×

featuring moving cameras and multiple interacting and moving objects. Across both datasets, we defined thirteen queries as shown Table 1: nine for the Toll Booth and four for Volleyball. Each query includes at least one multimodal LLM-based operator, targeting different extraction functions—such as *color recognition*, *object detection*, and *text extraction*. In addition, the queries perform streaming-level operations such as *aggregation* (e.g., counting objects over time windows) and *filtering* over attributes (e.g., detecting cars of a specific color). The benchmark does not include joins between streams, which we plan to explore in future work.

Experiment 1: End-to-End Gains. We evaluated Saṃsāra across all thirteen queries, comparing the naïve query plan—where every frame is processed by a large multimodal LLM (Qwen2.5-VL [3])—against the optimized query plans that include semantic, logical, and physical optimizations. Throughput was measured in frames per second (FPS), while accuracy was computed at the query-result level. Across all queries, Saṃsāra achieved significant end-to-end speedups. In particular, the queries Q1-Q9 on the Tool Both data, which are less complex to optimize, show higher benefits than the Volleyball queries (Q10-Q13). However, all queries show significant throughput improvements, with the best case reaching up to 10× *higher FPS*. On average, optimizations allowed several queries to reach or surpass real-time performance, turning previously infeasible pipelines into practical streaming deployments. Accuracy losses caused by model specialization and operator reordering were minimal, with a mean accuracy drop of *7%* relative to the baseline (naive execution). Although semantic optimization may introduce errors through region cropping or frame skipping, our analysis indicates that the observed accuracy drop is primarily due to incorrect recognition by specialized models rather than to the optimization process. This demonstrates that Saṃsāra can substantially improve performance without sacrificing correctness.

Experiment 2: Ablation Study. To quantify the contribution of each optimization phase, we performed an ablation study summarizing the *minimum*, *average*, and *maximum* FPS improvements across all thirteen queries. The results in Table 2 confirm that each optimization phase—*semantic*, *logical*, and *physical*—makes a meaningful contribution to overall performance, depending on the query and data characteristics. Semantic optimizations produced the highest average and maximum gains by enabling early data reduction. Logical optimizations were most beneficial in cases where *filter pushdown* could be applied and efficiently realized via low-cost operators. Physical optimizations further improved performance by selecting or adapting a lightweight model (i.e., the *YOLOv8[29]* object detector instead of the full-scale multimodal LLM (Qwen2.5-VL)).

4 Road Ahead

This section outlines our research plan to evolve Saṃsāra from a proof-of-concept to a principled, high-performance super optimizer for multimodal streaming systems.

Super-Optimization of Multimodal Streaming. While we have presented a first prototype of Saṃsāra and demonstrated its promise of super-optimization for enabling multimodal streaming systems, many research questions remain open in all stages of the optimization. A core step is to transform semantic optimization—leveraging common-sense knowledge captured by MLLMs—into a theory-backed, system-enforced capability that enhances continuous query plans by semantic rewrites. Furthermore, also all other optimization phases (logical and physical) yield open questions including a general rewrite engine which also include how rewrites such as filter-push-down can be implemented in a general manner as discussed before. An additional promising direction is *adaptive model specialization*, where long-running queries with stable logic but evolving data (e.g., a fixed camera feed) allow lightweight retraining or pruning to yield faster and smaller models tailored to given workloads. All these directions aim to evolve Saṃsāra into an autonomous, correctness-aware planner for multimodal streams.

Multimodal Streaming System Techniques. While super-optimization is a key aspect of enabling the use of MLLMs for rich and efficient querying, integrating MLLMs into streaming systems opens many orthogonal directions for further exploration. Beyond optimization, other important aspects include *semantic caching* to reduce redundant MLLM calls or even use caches for similar queries. Moreover, we have integrated MLLMs as user-defined map operators into the query execution, but there are other opportunities, such as integrating MLLMs into more *streaming operators* for multimodal data like joins over image streams to fuse streams across sources (e.g., merge two camera streams). A promising starting point for exploring such operators is the rich body of work on video data analytics [16, 17, 23]. However, lifting these techniques into SPSs introduces new challenges: continuous query semantics (windows and state), strict end-to-end latency constraints under backpressure, and the need to optimize plans jointly with continuously evolving multimodal streams. Finally, extending these ideas beyond vision to other modalities, such as audio or unstructured sensor data, presents further challenges. Although some techniques will transfer, new challenges will emerge across all layers—from optimization to operator design and beyond.

Language Extensions and Novel Benchmarks. Integrating multimodality at the query language level is essential for declarative and efficient multimodal analytics. Future research will also focus on *language extensions* that enrich continuous query semantics with multimodal predicates forming the foundation for the super-optimization stack above. Finally, advancing this field requires robust *benchmarks* and evaluation frameworks that provide annotated multimodal datasets. While benchmarks started emerging for multimodal databases, such as Sembench [18], they are oriented towards query answering against static datasets. In particular, they contain large corpora of annotated unstructured data (such as images or documents) that are not part of a continuously evolving scenario. Thus, they struggle to capture dimensions that are crucial to stream processing, such as temporal continuity of the scene and low-latency, high throughput processing.

Acknowledgments

This work was supported by the LOEWE Spitzenprofessur programme (III 5-519/05.00.003-(0005)), etaGPT project under grant number 03EN4107, Athene Young Investigator Programme, and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy (EXC-3057/1 "Reasonable Artificial Intelligence", Project No. 533677015). We also thank DFKI, hessian.AI and French Research Agency under grant agreement nr. ANR-22-CE23-0001 Polyflow.

Artifacts

The query plans and data used in this paper are available at: https://github.com/DataManagementLab/Samsara

References

[1] Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, et al. 2015. The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of VLDB* 8, 12 (2015), 1792–1803.

[2] Arvind Arasu, Mitch Cherniack, Eduardo F. Galvez, David Maier, Anurag Maskey, Esther Ryvkina, Michael Stonebraker, and Richard Tibbetts. 2004. Linear Road: A Stream Data Management Benchmark. In *Proceedings of VLDB 2004, Toronto, Canada, August 31 - September 3 2004*. Morgan Kaufmann, 480–491. doi:10.1016/B978-012088469-8.50044-9

[3] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923* (2025).

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

[5] Netflix Technology Blog. 2018. Keystone Real-time Stream Processing Platform. https://t.ly/61XZJ. [Online; accessed 10-07-2025].

[6] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache Flink™: Stream and Batch Processing in a Single Engine. *IEEE Data Eng. Bull.* 38, 4 (2015), 28–38. http://sites.computer.org/debull/A15dec/p28.pdf

[7] Ankit Chaudhary, Steffen Zeuch, and Volker Markl. 2020. Governor: Operator Placement for a Unified Fog-Cloud Environment. In *EDBT*. EDBT.

[8] Yueting Chen, Xiaohui Yu, and Nick Koudas. 2020. Tqvs: Temporal queries over video streams in action. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2737–2740.

[9] Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*. PMLR, 10323–10337.

[10] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: Accurate Post-training Compression for Generative Pretrained Transformers. *arXiv preprint arXiv:2210.17323* (2022).

[11] Google. [n. d.]. Introduction to AI and ML in BigQuery. https://cloud.google.com/bigquery/docs/bqml-introduction?hl=en Accessed on 8.10.2025.

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

[13] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[14] Mostafa S Ibrahim, Srikanth Muralidharan, Zhiwei Deng, Arash Vahdat, and Greg Mori. 2016. A hierarchical deep temporal model for group activity recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1971–1980.

[15] Saehan Jo and Immanuel Trummer. 2024. ThalamusDB: Approximate Query Processing on Multi-Modal Data. *Proc. ACM Manag. Data* 2, 3 (2024), 186.

[16] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics. *Proceedings of VLDB* 13, 4 (2019), 533–546. doi:10.14778/3372716.3372725

[17] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Deep CNN-Based Queries over Video Streams at Scale. *Proceedings of VLDB* 10, 11 (2017), 1586–1597. doi:10.14778/3137628.3137664

[18] Jiale Lao, Andreas Zimmerer, Olga Ovcharenko, Tianji Cong, Matthew Russo, Gerardo Vitagliano, Michael Cochez, Fatma Özcan, Gautam Gupta, Thibaud Hottelier, H. V. Jagadish, Kris Kissel, Sebastian Schelter, Andreas Kipf, and Immanuel Trummer. 2025. SemBench: A Benchmark for Semantic Query Processing Engines. arXiv:2511.01716 [cs.DB] https://arxiv.org/abs/2511.01716

[19] Rayson Laroca, Everton V. Cardoso, Diego Rafael Lucio, Valter Estevam, and David Menotti. 2022. On the Cross-dataset Generalization in License Plate Recognition. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2022, Volume 5: VISAPP, Online Streaming, February 6-8, 2022*, Giovanni Maria Farinella, Petia Radeva, and Kadi Bouatouch (Eds.). SCITEPRESS, 166–178. doi:10.5220/0010846800003124

[20] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, et al. 2025. Palimpzest: Optimizing ai-powered analytics with declarative query processing. In *Proceedings of the Conference on Innovative Database Research (CIDR)*. 2.

[21] Ryan Marcus. 2023. Learned Query Superoptimization. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB 2023), Vancouver, Canada, August 28 - September 1, 2023 (CEUR Workshop Proceedings, Vol. 3462)*. CEUR-WS.org. https://ceur-ws.org/Vol-3462/AIDB5.pdf

[22] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators and Their Optimization: Enabling LLM-Based Data Processing with Accuracy Guarantees in LOTUS. *Proceedings of VLDB* 18, 11 (Sept. 2025), 4171–4184. doi:10.14778/3749646.3749685

[23] Matthew Russo, Tatsunori Hashimoto, Daniel Kang, Yi Sun, and Matei Zaharia. 2023. InQuest: Accelerating Aggregation Queries on Unstructured Streams of Data. In *Proceedings of VLDB*, Vol. 16. 2897–2910. doi:10.14778/3611479.3611496

[24] Mohammad Sadoghi, Martin Labrecque, Harsh Singh, Warren Shum, and Hans-Arno Jacobsen. 2010. Efficient event processing through reconfigurable hardware for algorithmic trading. *Proceedings of VLDB* 3, 1–2 (sep 2010), 1525–1528. doi:10.14778/1920841.1921029

[25] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[26] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. *Proceedings of VLDB* 18, 9 (2025), 3035–3048.

[27] Snowflake. [n. d.]. AI Functions as UDFs. https://docs.snowflake.com/en/sql-reference/functions/ai_filter Accessed on 13.02.2026.

[28] Matthias Urban and Carsten Binnig. 2024. CAESURA: Language Models as Multi-Modal Query Planners. In *14th Conference on Innovative Data Systems Research, CIDR 2024, Chaminade, HI, USA, January 14-17, 2024*. www.cidrdb.org. https://vldb.org/cidrdb/2024/caesura-language-models-as-multi-modal-query-planners.html

[29] Rejin Varghese and M Sambath. 2024. Yolov8: A novel object detection algorithm with enhanced performance and robustness. In *2024 International conference on advances in data engineering and intelligent computing systems (ADICS)*. IEEE, 1–6.

[30] Di Wang, Elke A. Rundensteiner, Han Wang, and Richard T. Ellison. 2010. Active complex event processing: applications in real-time health care. *Proceedings of VLDB* 3, 1–2 (sep 2010), 1545–1548. doi:10.14778/1920841.1921034