[Vision Paper] Towards a Multimodal Stream Processing System

Uélison Jean Lopes dos Santos
TU Darmstadt and DFKI

Alessandro Ferri
TU Darmstadt

Szilard Nistor
TU Darmstadt and DFKI

Riccardo Tommasini
INSA Lyon

Carsten Binnig
TU Darmstadt and DFKI

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 more than 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.

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 [12], entertainment [3], healthcare [16], and the Internet-of-Things (IoT) [5]. 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.

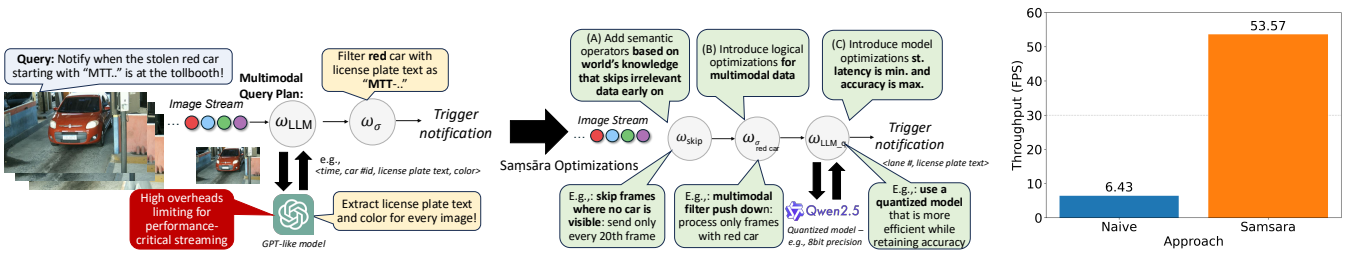
Leveraging multimodal LLMs for Streaming. Recent advances in multimodal large language models (MLLMs) have demonstrated remarkable abilities to process and integrate data across multiple modalities, such as images, text, and audio, providing contextual understanding that spans these diverse data types. As such, it seems an appealing idea to use MLLMs as a building block within stream processing systems. In fact, integrating MLLMs into stream processing opens up new possibilities to natively support rich queries on modalities beyond structured data, enabling systems to process and interpret multimodal data in real-time. Such capabilities could significantly enhance streaming systems in applications ranging from traffic monitoring using camera streams and autonomous vehicles to sports analytics and robotics.

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 station using a camera stream. While current data systems [6, 9, 11, 14, 15] have already proposed integrating MLLMs for querying multimodal data, streaming systems pose fundamentally different challenges. Streaming environments require extremely low latency and high throughput, which makes naive MLLM integration, simply by calling out to the model, infeasible. Achieving acceptable performance requires carefully optimized query execution. Unlike batch-oriented databases, streaming systems 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 streaming systems 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 this upfront effort leads to high performance benefits while running the query. For this, a super-optimizer adds several novel optimization steps. For example, we introduce a new phase in optimization called *semantic optimization* in addition to logical and physical optimizations, which optimizes plans based on a semantic understanding of data and queries to specialize the plan for a given scenario. For instance, 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: a super-optimizer, which we integrated into an existing streaming system Apache Flink [4]. In contrast to a naive evaluation in Flink without our optimizer, we can achieve orders-of-magnitude throughput improvements—in our example, from 6 to 53 images per second—showcasing the potential of super-optimizers for multimodal streaming. 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 structured data 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 remainder of the paper is organized as follows. Section 2 presents our vision for multimodal stream processing, highlighting the intuition behind the novel optimizations. Section 3 provides a case study that illustrates these optimizations through



(a) Example query before and after Saṃsāra optimization. **(b) Naive vs. optimized execution.**

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.

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 leveraging recent advances in multimodal large language models (MLLMs) as first-class operators within query plans. However, simply integrating MLLMs into query plans as done in batch-oriented systems such as CAESURA [15], Lotus [11], or DocETL [13] 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 multimodal query plans 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 over-specialization is tolerable, and it differs significantly from traditional database optimizers, which must produce plans quickly [10]. By spending more effort offline, Saṃsāra can leverage time-consuming techniques for all optimization steps and synthesize bespoke query plans. This shift from fast plan generation to deep plan synthesis opens an entirely new frontier in stream optimization.

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. One key innovation is a new 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 a deep understanding of the domain 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.

The Challenge of Semantic 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 LLM 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.

Bespoke Logical & Physical Optimizations. Beyond semantic optimizations, Saṃsāra applies novel logical and physical optimizations tailored to multimodal workloads as shown in Figure 2 (right). For example, it can logically rewrite the query plan and push down the filter for only processing red cars. The challenge here is that this filter must be cheaper to evaluate (i.e., without an MLLM), e.g., by using simple computer vision algorithms to detect that the image contains sufficient areas of red-ish pixels in our running example. Another interesting direction is that we can afford 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 size of MLLMs by tailoring them to data and one particular query. Although these techniques may take minutes or hours to apply, they can be executed offline before deploying the streaming query. These combined optimizations allow Saṃsāra to execute complex multimodal queries efficiently and at scale.

3 Saṃsāra: A First Super-Optimizer

This section presents our super-optimizer prototype: Saṃsāra. We show its design and functionality through a detailed case study that serves as a running example throughout this section.

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. Our initial evaluation further extends to a broader set of queries and datasets. To construct the case study, we created a dataset inspired by the Linear Road Benchmark, a well-established

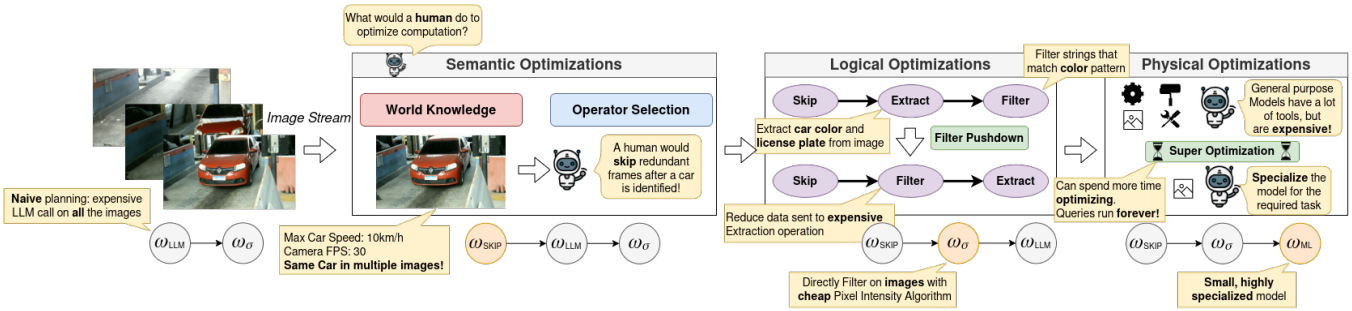


Figure 2: Overview of Saṃsāra, a super-optimizer to enable multimodal streaming systems which transforms a naive multimodal query plan into an optimized plan through a series of novel semantic, logical, and physical optimizations which aggressively optimize the plan for a particular long-running streaming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

benchmark for streaming traffic monitoring data [2]. Specifically, we incorporated images from the Rodosol-ALPR dataset [8] 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 by specific vehicle attributes or counting vehicles to detect traffic patterns, and shows how Saṃsāra optimizes different query types. For the purpose of the case study, we now focus on a specific query to demonstrate how it can be optimized 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 the naive multimodal query plan 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 the design of Saṃsāra –our first super-optimizer.

3.2.1 Semantic Query Optimization. Our vision introduces a new optimization phase, termed *semantic query optimizations*, which uses world knowledge and contextual reasoning to rewrite multimodal streaming query plans in a human-like yet systematic fashion. Large language models (LLMs) play a central role in our framework. We leverage them throughout semantic optimization—to extract semantic priors, and to propose valid plan rewrites. This differs fundamentally from existing multimodal query systems, such as CAESURA [15], which primarily employ LLMs to translate natural-language queries into executable plans. In contrast, we harness the reasoning capabilities of LLMs *after* a plan has already been constructed, to optimize it semantically rather than syntactically.

The key challenge is 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; then, it uses an LLM-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. Naively prompting an LLM to "improve" a query plan typically fails, since it lacks structured context and domain constraints. Instead, our optimizer uses the LLM 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, the optimizer first invokes the LLM 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 LLM 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 guiding the subsequent steps.

(2) Operator Selection. Using the extracted symbolic representation of the scene, the optimizer next invokes the LLM 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 of data reduction tools from a given catalog of tools. Selecting the tools involves both *cross-frame* and *intra-frame* reasoning. In the cross-frame dimension, the LLM 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 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 LLM 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 LLM explicitly rejects `Greyscale()` reduction, correctly reasoning that it would remove color cues critical to the query semantics. At this stage, the optimizer holds a set of validated candidate operators, each annotated with semantic preconditions.

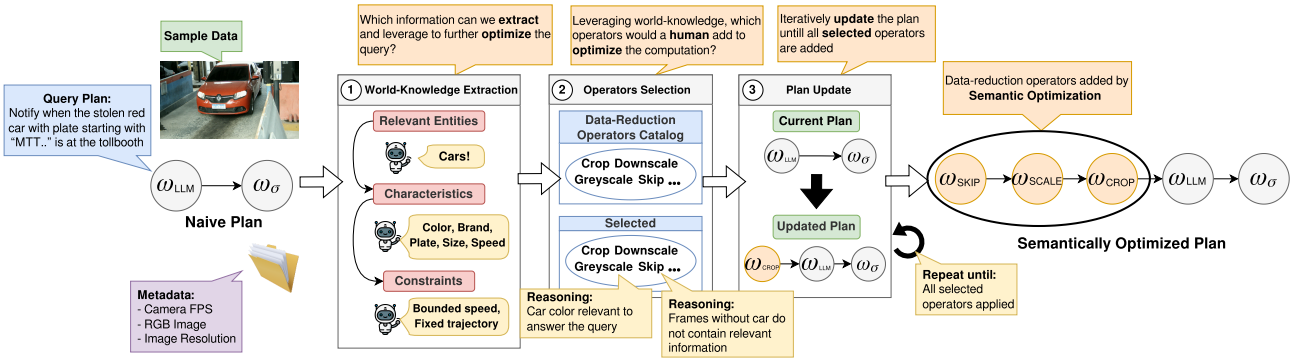


Figure 3: Overview of the semantic optimization procedure applied to the tollbooth query. Starting from a naive plan (*Extract + Filter*), the optimizer leverages an LLM to perform three reasoning phases: (1) *World-Knowledge Extraction*, where it identifies relevant entities (cars), their properties (color, size, speed), and real-world dynamics (limited movement through the tollbooth); (2) *Operator Selection*, where it proposes semantically motivated operators 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 (*Crop + Extract + Filter*). The figure shows how LLM-guided reasoning bridges data semantics and query intent to automatically produce more efficient execution plans.

(3) Plan Update. Finally, the optimizer integrates these operators into the query plan. Here again, the LLM 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, and in fact, we use this loop for all 13 queries over 2 different data streams in our initial evaluation. Consider a sports analytics query detecting which player currently possesses the ball in a soccer match. The LLM 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, the optimizer thus introduces `Skip` and `Crop` operators driven by semantic understanding rather than syntactic structure.

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 employ 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 the optimizer into a self-correcting agent—capable of hypothesizing, testing, and refining its own rewrites through LLM-guided reasoning. In contrast to systems like CAESURA [15] that use LLMs to construct executable plans from natural language, our approach establishes a new paradigm of *semantic optimization*, where LLMs reason about meaning to improve efficiency without altering intent.

3.2.2 Logical Optimization. Logical optimization in traditional databases rewrites query plans using heuristics such as filter pushdown or projection pushdown to improve efficiency while preserving correctness. In Saṃsāra, we aim to achieve a similar effect for multimodal streaming plans, where operators act on images, text, or audio instead of structured tuples.

The challenge is that standard rewrite rules do not exist for multimodal operators. To address this, we leverage known relational optimization techniques and map them to multimodal data. For example, in our running scenario, a filter selecting frames with red cars—originally applied after an expensive MLLM-based *Extract* operator—can be pushed earlier in the plan. Unlike structured data, this requires transforming the filter from a predicate on extracted attributes into a low-cost operation on the raw images. If this transformation is too expensive, the early filter could increase overall cost, which naturally connects logical optimization with physical optimization, as algorithm selection impacts efficiency.

Saṃsāra leverages LLMs to reason about low-overhead operators' instantiation. For instance, the red-car filter allows for either a similarity search over embeddings or a simpler computer vision classifier. Other relational rules, such as projection or aggregation pushdown, can also be adapted: projection pushdown can correspond to cropping only the relevant parts of an image needed for downstream extraction, such as the license plate and car color.

The logical optimization is threefold as shown in Figure 4. First, *Data Model Reconciliation* provides a relational interpretation of the data modality. For images, pixels are mapped to tuples with schema (row_id, column_id, red, green, blue), with row and column forming a composite primary key. Second, *Operation Mapping* translates multimodal operators to relational analogs: *Crop* aligns with projection, while *Downscale* corresponds to aggregation. Third, *Optimization Rule Mapping* applies relational rewrite rules: for example, the *Crop* operator can be pushed before *Downscale*, mirroring filter pushdown. Similarly, selection predicates on multiple attributes (e.g., color and license plate) can be split, pushing the less expensive filter earlier to reduce the workload of downstream operators.

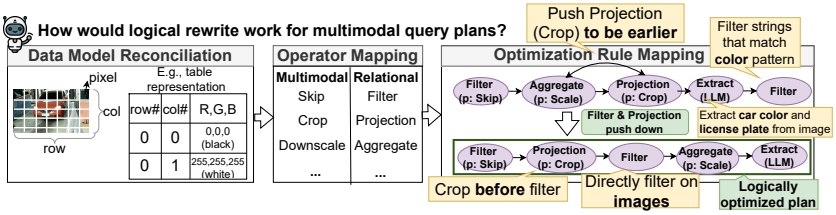


Figure 4: Overview of logical optimiseation in Saṃsāra. It starts with a semantically optimized plan with `Skip`, `Scale`, `Crop`, `MLLM`, and `Filter` operators to optimize it analogously to the relational counterparts, i.e., (1) Data model reconciliation that maps images to tuples, (2) Operator mapping that maps multimodal operators to relational and (3) Optimization rule mapping that applies logical rewrite rules. The resulting logically optimiseed plan is highlighted on the right.

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

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

These optimizations have been implemented in Saṃsāra, significantly reducing the workload of expensive plans by plan rewrites. Defining a systematic mapping from relational rewrites to multimodal operators—and specifying how operators after rewrite (e.g., filters and projections) can be efficiently realized on unstructured data—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 select a bespoke model or synthesize one tailored to the query and data at hand, allowing for *super-optimization* of the query execution. Such a step also includes reducing the computation and memory requirements of MLLM models. Techniques such as quantization, pruning, knowledge distillation, and low-rank factorization are employed to compress models while preserving functionality. In our running example, text extraction leverages a quantised MLLM, while YOLO pre-filters frames containing cars. Quantization reduced the MLLM's weights and activations to 8-bit integers, halving model size and memory bandwidth, while YOLO removed irrelevant frames early in the pipeline, improving inference speed and overall FPS.

Streaming data, however, 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 continuously evolving. We can make use of this fact for physical optimization. For example, let us look at model pruning.

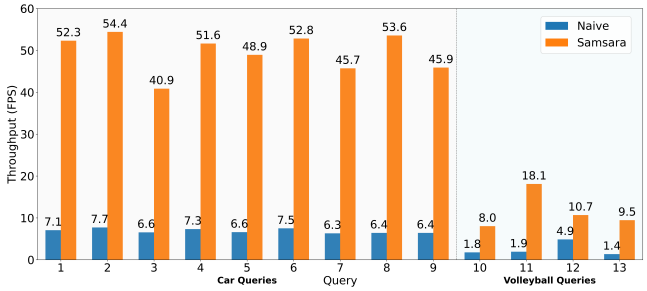


Figure 5: End-to-End gains for all(13) queries cf Table 1. Saṃsāra has up to 10× speedup over naive execution

Model pruning is a technique to reduce the size and computational cost of the model by removing unnecessary parameters. Highly similar parameters, too aggressive static pruning, may over-specialize and degrade performance when data characteristics change. For example, data from a traffic camera scenario may switch from low to high-density traffic: pruning aggressively in 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 streaming 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.

Another key aspect of physical optimization is selecting the appropriate algorithm. A large MLLM can sometimes be replaced by a smaller or distilled model, trading off computational cost against accuracy. Optimizers must consider accuracy constraints: similar to multimodal data systems such as LOTUS or Palimpzest [9], which optimize plans for batch processing, where users to specify required relative accuracy (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 multi-modal streaming systems is another open challenge for developing a super-optimizer for multimodal streaming systems.

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 [7], 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)—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 while 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). These results demonstrate 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* object detector instead of the full-scale multimodal LLM (Qwen2.5-VL)).

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

Table 2: Speedup by optimization phases of Saṃsāra over naive execution. We observe that across all queries, the minimum speedup is at least 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, and that semantic optimization is particularly important.

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, many research questions remain open. The next step is to transform semantic optimization—leveraging common-sense knowledge captured by LLMs—into a theory-backed, system-enforced capability that enhances continuous query plans without compromising declarativity. Key challenges include developing formal underpinnings for semantic rewrites with uncertainty-aware semantics, adapting classic planning problems such as cardinality estimation and plan enumeration to multimodal settings, and extending operator synthesis with specification-by-contract and cost/latency-aware search. 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 Systems. 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). Finally, extending these ideas to other modalities such as audio or other non-structured sensor data is an exciting next step. 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, uncertainty-aware constructs, and semantic annotations, forming the foundation for the super-optimization stack above. Finally, advancing this field requires robust *benchmarks* and evaluation frameworks that provide annotated multimodal datasets and measure both throughput and semantic correctness, ensuring reproducible and comparable progress.

Artifacts

The source code 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. *PVLDB* 8, 12 (2015), 1792–1803.

[2] Arvind Arasu, Mitch Cherniack, Eduardo Galvez, David Maier, Anurag S Maskey, Esther Ryvkina, Michael Stonebraker, and Richard Tibbetts. 2004. Linear road: a stream data management benchmark. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. 480–491.

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

[4] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache flink: Stream and batch processing in a single engine. *The Bulletin of the Technical Committee on Data Engineering* 38, 4 (2015), 28–38.

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

[6] 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.

[7] 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.

[8] Rayson Laroca, Everton V Cardoso, Diego R Lucio, Valter Estevam, and David Menotti. 2022. On the cross-dataset generalization in license plate recognition. *arXiv preprint arXiv:2201.00267* (2022).

[9] 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.

[10] Ryan Marcus. 2023. Learned Query Superoptimization. In *5th International Workshop on Applied AI for Database Systems and Applications (AIDB) colocated with VLDB 2023*.

[11] 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. *Proc. VLDB Endow.* 18, 11 (Sept. 2025), 4171–4184. doi:10.14778/3749646.3749685

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

[13] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G Parameswaran, and Eugene Wu. 2024. Docetl: Agentic query rewriting and evaluation for complex document processing. *arXiv preprint arXiv:2410.12189* (2024).

[14] Snowflake. [n. d.]. AI Functions as UDFs. https://docs.snowflake.com/en/sql-reference/functions/ai_filter Accessed on 8.10.2025.

[15] Matthias Urban and Carsten Binnig. 2023. CAESURA: language models as multi-modal query planners. *arXiv preprint arXiv:2308.03424* (2023).

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