# LensDB: Compressed Learned Indexes for Traffic Video Analysis

Raj Shah
rajshah@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Karvy Mohnot
kmohnot3@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

## Abstract

Iterative video analytics incurs high storage and compute costs for city-scale surveillance. We propose a **compressed learned index** that decouples ingestion from inference by replacing per-frame object detection with a sparse embedding collection. Our pipeline integrates heuristic keyframe selection (Frame Difference, SSIM, MOG2, Flow) to filter temporal redundancy, followed by semantic encoding via a CLIP embedder and a lightweight MLP for count estimation. Crucially, by shifting exploratory queries to the latent space, we bypass the latency of video decoding from network-attached storage (NFS, which is typical for storing large amounts of video data), enabling quick preliminary analysis without video decoding cost. Evaluation on the VIRAT dataset demonstrates a 99.991% reduction in storage and sub-second query latency compared to exhaustive YOLO baselines. While the system achieves high F1 (0.963) for event detection ($T \geq 1$), we identify significant precision trade-offs in fine-grained counting ($T > 2$).

## Keywords

Compressed video indexes, Keyframe selection, CLIP embeddings, FAISS retrieval, Traffic analytics, Approximate query processing, Low-latency inference

**ACM Reference Format:**
Raj Shah and Karvy Mohnot. 2025. LensDB: Compressed Learned Indexes for Traffic Video Analysis. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (CS 8803 LRV).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 Introduction

Traffic and surveillance cameras generate vast continuous video streams, creating significant challenges for large-scale storage, processing, and query responsiveness. Even a single fixed camera can produce tens of gigabytes per day, and modern video analytics pipelines often require computing deep visual features for every frame, resulting in prohibitive compute and storage overheads. Existing systems such as BlazeIt [4], FiGO [2], VIVA [8], and NoScope [5] accelerate specific query types, but they still rely on dense per-frame features or query-specific retraining, which limits their ability to scale to city-wide deployments with strict latency and storage budgets.


*CS 8803 LRV, Atlanta, GA*

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

A central question therefore arises: Can we build a video index that stores only a compact latent representation while still supporting accurate, low-latency queries? In particular, traffic-focused analytics workloads, such as retrieving frames where the number of visible cars exceeds a threshold, should have sub-second response times and must operate within tight storage constraints. Achieving this requires rethinking the traditional "process all frames" paradigm.

In this project, we design and evaluate a compressed learned index that replaces dense, frame-by-frame processing with a combination of (1) keyframe selection, (2) CLIP-based visual embeddings, (3) FAISS similarity search, and (4) a lightweight MLP count predictor. Keyframe selection algorithms such as frame-difference, SSIM drop, MOG2 foreground estimation, and optical flow reduce 90–99% of redundant frames before embedding, dramatically lowering compute and storage requirements. The resulting index persists only the CLIP embeddings of selected frames and supports fast semantic retrieval using a text query such as "`car`".

To approximate car-count queries, we train a compact MLP on top of CLIP embeddings to predict frame-level vehicle counts. This predictor, pre-trained on COCO and fine-tuned on VIRAT, acts as a selective post-filter to improve precision without sacrificing recall. Ground-truth labels are obtained via a YOLO11x detector, which serves as a proxy oracle for evaluation.

Our findings show that the proposed index can compress a 1.4 GB video into 1.5 MB of embeddings while still supporting fast, approximate car-count queries. This storage requirement could further be reduced to 132kb by using keyframe detection techniques to remove temporal duplications. The retrieval of FAISS on CLIP embeddings achieves high recall, and the MLP count-predictor significantly improves precision at low thresholds. Although challenges remain—particularly for high car-count cases and temporal understanding—our results demonstrate that compact learned indexes are a promising direction for scaling real-time video analytics in resource-constrained environments.

## 2 Background

This section reviews the foundational concepts underlying our compressed learned index for traffic video analysis. We describe the keyframe selection techniques used to reduce redundancy, the embedding and retrieval models used to construct the index, and the dataset employed for evaluation.

### 2.1 Keyframe Selection

Traffic video streams contain substantial temporal redundancy, as consecutive frames often differ only minimally. Processing or embedding every frame significantly increases computational load and storage requirements. To address this, we evaluate several

lightweight techniques that identify informative or eventful frames while discarding redundant ones.

*2.1.1 SSIM-Based Keyframe Detection.* Structural Similarity (SSIM) is a perceptual metric that compares two images using three components: luminance, contrast, and structural patterns. Let $I_t$ denote the video frame at time $t$ and $I_{t-1}$ the previous frame. We compute the SSIM drop

$$\Delta_{\text{SSIM}}(t) = 1 - \text{SSIM}(I_t, I_{t-1}),$$

where $\text{SSIM}(I_t, I_{t-1}) \in [0, 1]$ measures their perceptual similarity. A value close to 1 indicates that the two frames are nearly identical, while values closer to 0 correspond to substantial changes in illumination, structure, or object layout.

Because SSIM incorporates perceptual structure rather than raw pixel differences, a large $\Delta_{\text{SSIM}}(t)$ typically indicates a meaningful scene change, such as a vehicle entering or leaving the frame or a notable shift in lighting.

*2.1.2 Frame-Difference Keyframe Detection.* The frame-difference method identifies keyframes by measuring the raw pixel change between consecutive frames. Let $I_t$ and $I_{t-1}$ denote two grayscale video frames at times $t$ and $t-1$, respectively. We compute the mean absolute difference

$$D(t) = \frac{1}{HW} \sum_{x=1}^{W} \sum_{y=1}^{H} |I_t(x, y) - I_{t-1}(x, y)|,$$

where $H$ and $W$ are the height and width of the frame. A large value of $D(t)$ indicates substantial pixel-level change, typically caused by an object entering or leaving the scene or by abrupt illumination variation.

*2.1.3 MOG2 Background Subtraction.* The MOG2 method detects keyframes by identifying regions of foreground motion. It maintains a statistical background model for each pixel and labels pixels as *foreground* when their intensity no longer fits the learned background distribution. Applying MOG2 to each frame produces a binary mask that highlights moving objects such as vehicles or pedestrians.

To measure activity, we compute the fraction of pixels marked as foreground. Frames with a high proportion of moving pixels are interpreted as containing meaningful events and are selected as keyframes. This approach works well in traffic scenes where object motion is the primary signal of interest.

*2.1.4 Optical Flow.* Optical flow estimates the apparent motion of pixels between consecutive frames. Let $I_t$ and $I_{t-1}$ denote frames at times $t$ and $t-1$. Using dense Farnebäck flow, we compute a motion field $(u_t(x, y), v_t(x, y))$ where $u_t$ and $v_t$ represent horizontal and vertical motion at pixel $(x, y)$.

From this field, we derive a motion-magnitude score

$$F_{\text{flow}}(t) = \frac{1}{HW} \sum_{x=1}^{W} \sum_{y=1}^{H} \sqrt{u_t(x, y)^2 + v_t(x, y)^2},$$

which measures the average amount of motion between $I_t$ and $I_{t-1}$. Large values of $F_{\text{flow}}(t)$ typically indicate meaningful activity such as vehicles accelerating, turning, or entering the scene.

Unlike frame-difference methods that only capture pixel-level intensity changes, optical flow captures *directional* and *continuous* motion patterns. This makes it particularly useful for identifying subtle events like slow-moving vehicles, gradual traffic buildup, or partial occlusions that may not cause large structural or intensity changes.

## 2.2 CLIP Embeddings

CLIP (Contrastive Language-Image Pretraining) is a multimodal representation model trained on large image-text datasets. It learns to project images and natural-language descriptions into a shared embedding space, where semantically related image and text pairs lie close together. The model consists of two encoders: an image encoder (such as ViT-B/32) that maps an input image to a fixed-dimensional vector, and a text encoder that produces a corresponding embedding for a textual prompt.

## 2.3 FAISS Similarity Search

FAISS (Facebook AI Similarity Search) is a library designed for efficient similarity search and clustering over large collections of high-dimensional vectors. It provides implementations of both exact and approximate nearest neighbor (ANN) search, supporting a variety of index structures such as flat L2 search, inverted-file systems, and graph-based methods like HNSW.

## 2.4 MLP Count Predictor

A Multilayer Perceptron (MLP) is a feed-forward neural network composed of one or more fully connected layers with nonlinear activation functions. Given an input vector, each layer performs an affine transformation followed by a nonlinearity, allowing the network to model complex, nonlinear relationships between inputs and outputs.

## 2.5 VIRAT Dataset

The VIRAT dataset is a large-scale video benchmark designed for research on surveillance, activity recognition, and scene understanding. It contains outdoor recordings captured from fixed cameras in diverse environments such as parking lots, driveways, and building entrances. The videos exhibit real-world challenges including varying illumination, occlusions, and heterogeneous vehicle and pedestrian activity.

# 3 Design

We propose **LensDB**, a compressed learned index designed to decouple the high storage and compute cost of video ingestion from the latency requirements of interactive querying. As illustrated in Figure 1, the system architecture is bifurcated into an offline *Ingestion Pipeline* and an online *Inference Pipeline*. This separation allows us to shift the burden of frame-level analysis to the ingestion phase—performed once per video—while enabling sub-second latency for subsequent exploratory queries.

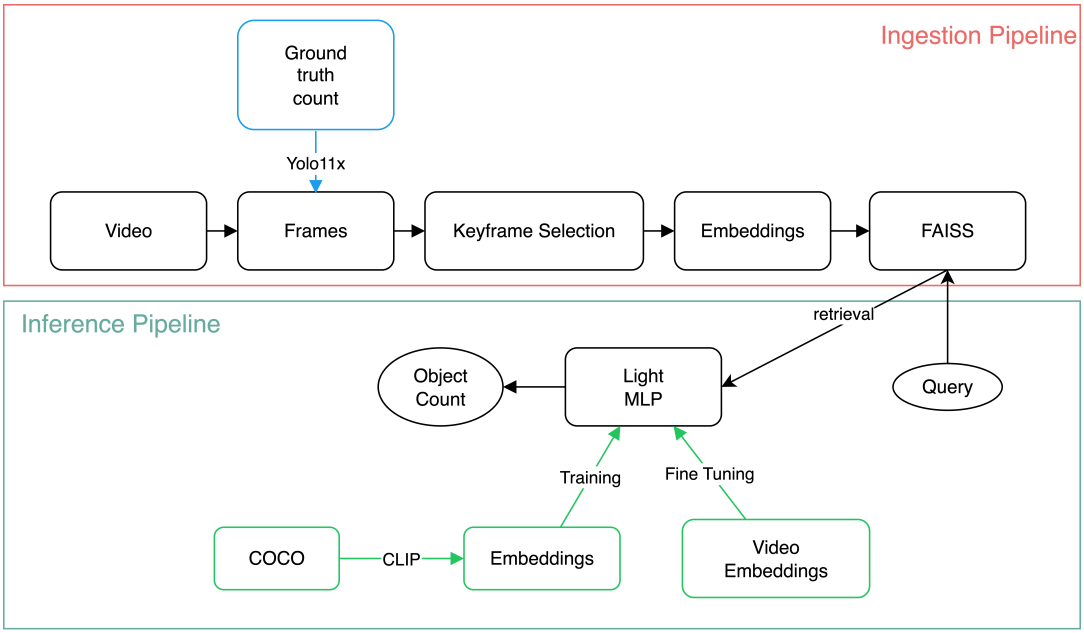


**Figure 1: System Architecture of LensDB. The Ingestion Pipeline (top) transforms raw video into a sparse index by filtering redundant frames via statistical heuristics and encoding keyframes into CLIP embeddings. The Inference Pipeline (bottom) executes approximate queries by performing similarity search in the latent space and refining results using a lightweight MLP count predictor trained on proxy ground truth.**

## 3.1 Ingestion Pipeline

The ingestion pipeline is responsible for reducing the raw video stream into a compact, searchable latent representation. This process involves four sequential stages: sampling, heuristic filtering, semantic embedding, and indexing.

*3.1.1 Frame Sampling and Pre-processing.* The system ingests video streams at a baseline sampling rate of 1 FPS. While surveillance video is typically recorded at 30 FPS, traffic analytics for congestion or occupancy rarely requires millisecond-level resolution. Downsampling to 1 FPS acts as the first layer of data reduction. Sampled frames are resized to $224 \times 224$ pixels to match the input resolution requirements of the downstream vision encoders.

*3.1.2 Heuristic Keyframe Selection.* To further compress the data, we employ a heuristic pre-selector to filter temporal redundancy. In surveillance footage, vast segments often contain static backgrounds or negligible motion. We implement and evaluate four statistical selectors: Frame Difference, Structural Similarity (SSIM), Background Subtraction (MOG2), and Optical Flow.

Each selector calculates a novelty score $S_t$ for the current frame $t$. To handle signal noise, we apply an Exponential Moving Average (EMA) to $S_t$. A frame is selected as a *keyframe* only if its smoothed score exceeds an adaptive threshold defined by the Median Absolute Deviation (MAD) of the recent history window. Furthermore, to prevent indefinite gaps in the index during long static periods, we enforce a "windowed keep" policy, guaranteeing at least one keyframe is retained every fixed window (e.g., 150 seconds) regardless of the novelty score. This step reduces the data volume by upto 96% before any deep learning inference occurs.

*3.1.3 Semantic Embedding.* Retained keyframes are passed through the CLIP (ViT-B/32) image encoder. Unlike traditional object detectors that output discrete bounding boxes and class labels, CLIP produces a 512-dimensional continuous semantic vector. This vector captures the global context of the scene (e.g., "parking lot," "congestion," "empty street") in a zero-shot manner. The embeddings are $L2$-normalized to facilitate cosine similarity search in the subsequent indexing stage.

*3.1.4 Index Construction.* We utilize FAISS to construct a Flat L2 index containing the normalized embeddings of all keyframes. Crucially, the index is paired with a lightweight metadata map. This map associates each indexed keyframe with the specific time range it represents (i.e., the timestamps of the redundant frames that were dropped). This allows the system to reconstruct the temporal duration of events during retrieval, despite the lossy compression.

*3.1.5 Proxy Ground Truth Generation.* To facilitate the training of our count predictor (described below), we generate ground-truth labels using a YOLO11x object detector. This detector runs on the raw frames during a one-time offline pass, acting as a "proxy oracle" to label car and person counts. These labels are used solely for supervising the MLP and evaluating system recall; they are not generated during runtime queries.

## 3.2 Inference Pipeline

The inference pipeline enables users to perform count-based queries (e.g., "Find frames with ≥ 5 cars") without accessing or decoding the raw video data from disk.

*3.2.1 Latent Similarity Search (Coarse Filter).* A user's text query is encoded using CLIP's text encoder into the shared multimodal embedding space. We perform a nearest neighbor search using the FAISS index to retrieve the top-$k$ most similar keyframe embeddings. This step acts as a coarse filter, leveraging CLIP's language-image alignment to rapidly identify frames that are semantically relevant to the query object (e.g., isolating frames containing vehicles from those containing only empty pavement).

*3.2.2 MLP Count Prediction (Fine Filter).* While CLIP captures semantic presence, it struggles to encode precise numerosity (e.g., distinguishing 5 cars from 10 cars). To resolve this, the retrieved embeddings are passed through a lightweight Multi-Layer Perceptron (MLP). The MLP architecture consists of a sequence of fully connected layers (Input 512 → Hidden Layers → Scalar Output) with ReLU activations and dropout for regularization. It maps the frozen, global semantic vector from CLIP to a scalar value representing the predicted object count. This allows LensDB to support complex predicates (e.g., $Count \geq T$) that standard semantic search cannot handle natively.

*3.2.3 Temporal Expansion.* Upon identifying positive keyframes via the MLP, the system queries the metadata map to retrieve the full set of timestamps associated with each keyframe. This effectively "decompresses" the result, presenting the user with the complete time windows where the event likely occurred, rather than just the sparse keyframes.

## 3.3 Training Strategy

The count predictor is trained using a two-stage transfer learning approach to mitigate data scarcity and domain shift.

**Stage 1: COCO Pre-training.** We first train the MLP on the COCO dataset. Since COCO contains diverse images of common objects (cars, people), this stage teaches the MLP to extract general density signals from CLIP embeddings, effectively learning "what a count looks like" in the latent space.

**Stage 2: VIRAT Fine-tuning.** We subsequently fine-tune the model on a 50% split of the VIRAT subset we used for this project. This adapts the predictor to the specific camera perspectives, lighting conditions, and object scales typical of the target surveillance environment. This hybrid approach significantly outperforms training on limited video data alone.

# 4 Evaluation

We evaluate LensDB along three dimensions: (1) retrieval quality for count-based queries, (2) storage savings from keyframe selection, and (3) ingestion and query latency. We compare three families of methods:

- **YOLO**: running YOLOv11m on every frame at 1fps.
- **All-Embeddings**: embedding every sampled frame with CLIP and applying the MLP count predictor (the "Full Embedder" in figures).

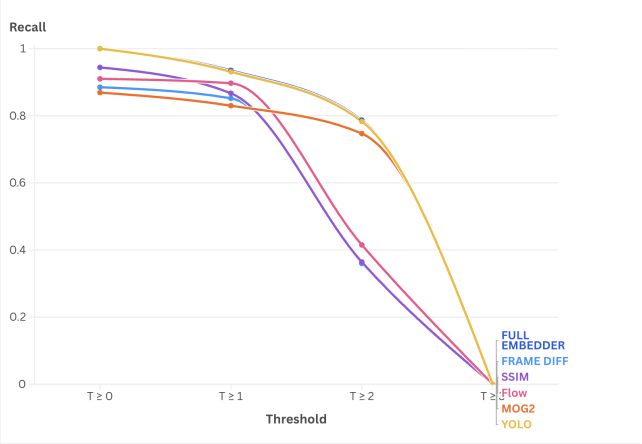


**Figure 2: Recall vs. Threshold. Recall as a function of the count threshold $T$ for the YOLO 11m, All-Embeddings, and four keyframe-based variants.**

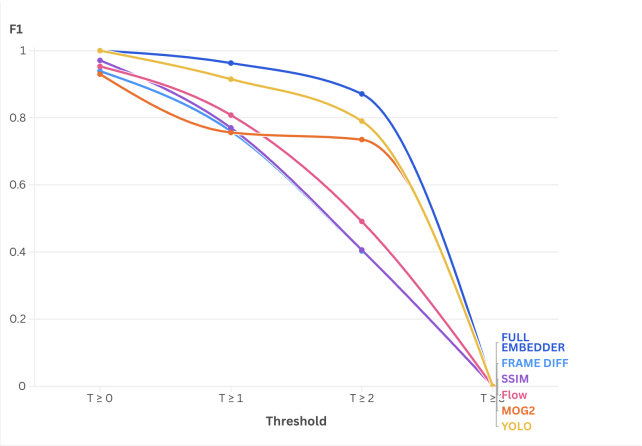


**Figure 3: F1-score vs. Threshold. F1-score for count predicates $Count \geq T$ across methods. All-Embeddings provides the highest overall F1, while keyframe variants remain competitive at low thresholds.**

- **Keyframe Variants**: four LensDB configurations that apply Frame Difference, SSIM, Optical Flow, or MOG2 to select keyframes prior to embedding.

Unless otherwise noted, we report metrics over a subset of 5 videos from VIRAT dataset sampled at 1 FPS and evaluate count predicates of the form "number of cars ≥ $T$" for thresholds $T \in \{0, 1, 2, 3\}$.

## 4.1 Accuracy and Compression

Figures 2 and 3 summarize retrieval quality as we increase the minimum count threshold. At $T \geq 1$, the All-Embeddings configuration achieves an F1-score of **0.963**, comparable to the YOLO11m (0.915) while operating entirely in the latent space. As $T$ increases, recall

**Table 1: Keyframe Compression Statistics Across Selectors**

| Selector | Total Frames | Keyframes | Avg Compression | Space Saved |
|---|---|---|---|---|
| FrameDiff | 2872 | 96 | 31.83× | 96.70% |
| SSIM | 2872 | 106 | 28.52× | 96.30% |
| Flow | 2872 | 99 | 30.73× | 96.60% |
| MOG2 | 2872 | 103 | 29.57× | 96.40% |

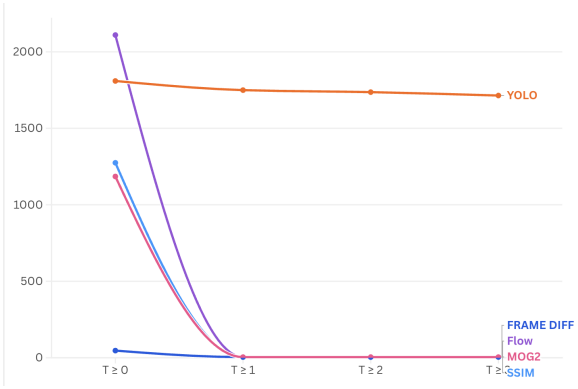


**Figure 4: Ingestion Latency per Frame. Average per-frame ingestion latency for YOLO, All-Embeddings, and keyframe variants (including heuristic computation and any embedding performed).**

and F1 decline for all methods. This loss could primarily be attributed to the under-representaiton of these classes in the training data used for the MLP (i.e., in COCO and in the videos used for fine-tuning MLP).

The keyframe variants follow a different trade-off. For presence-style queries ($T \geq 0$) and low thresholds ($T \geq 1$), Frame Difference, SSIM, Flow, and MOG2 all track the All-Embeddings curve reasonably closely, indicating that most frames satisfying these predicates are covered by the selected keyframes. As thresholds increase ($T \geq 2$), their recall and F1 decrease more noticeably. This behavior reflects the design goal: aggressive temporal filtering removes many visually similar frames, so a subset of high-count frames with limited motion or subtle appearance changes are no longer indexed. For applications where high-count events are rare but important, this trade-off may or may not be acceptable; for workloads dominated by $T \in \{0, 1\}$, the loss is modest.

Table 1 quantifies the storage benefit of these choices. All four selectors achieve between **28× and 32×** compression relative to storing embeddings for every frame, with over **96%** of potential embeddings removed. In practical terms, a 1.4 GB video that yields 2,872 sampled frames can be represented using only 96–106 keyframe embeddings, reducing both index size and downstream FAISS search cost. The keyframe variants therefore span a large portion of the accuracy–storage Pareto frontier: they give up some recall at higher thresholds in exchange for a two-order-of-magnitude reduction in stored vectors.

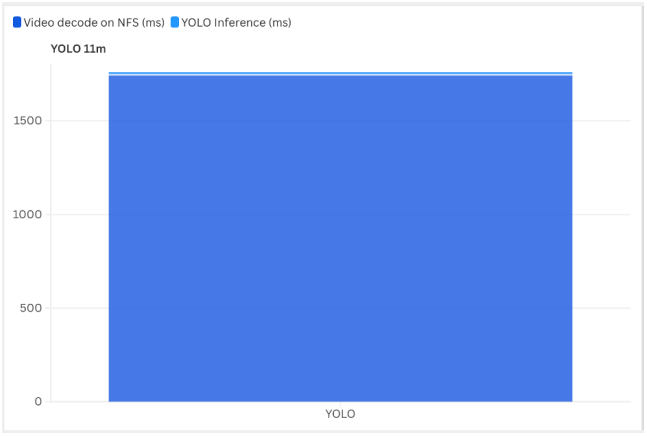


**Figure 5: YOLO Cost Breakdown. Most of the per-frame cost in the YOLO11m baseline comes from decoding video from NFS, with detector inference adding additional overhead.**

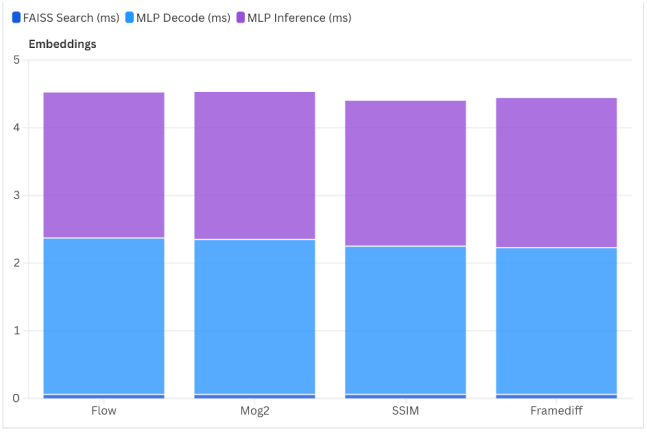


**Figure 6: LensDB Cost Breakdown. For embedding-based methods, ingestion time is dominated by MLP decoding and inference (≈4.5 ms); FAISS search contributes a negligible fraction (≈0.06 ms).**

## 4.2 Latency and Overhead

Figure 4 reports end-to-end ingestion latency. The YOLO11m is the slowest configuration, averaging **1809.4 ms** per frame. As shown in Figure 5, this cost is dominated by decoding video from NFS, with detector inference adding additional delay. This confirms that running a full detector over all frames is impractical for interactive workloads or large-scale analytics.

The All-Embeddings configuration reduces ingestion latency to **43.9 ms** per frame, a ∼ 41× speedup over the YOLO baseline. Microbenchmarks in Figure 6 indicates that MLP inference accounts for roughly 4.5 ms, while FAISS index operations add only 0.06 ms. Once embeddings are materialized, query-time costs are dominated by the MLP and are effectively independent of whether we index all frames or a small subset.

The keyframe variants introduce an additional dimension to this trade-off. When run over all frames, the CPU-side computation required for SSIM, Flow, or MOG2 can exceed the benefits of not embedding all the frames on the GPU. However, these heuristics are applied only once during ingestion, and for use cases where storage is a primary constraint, their cost can be amortized over many subsequent queries. At higher thresholds ($T \geq 1$), the variants generate far fewer embeddings, and their effective per-frame cost drops because the CLIP+MLP path is invoked only on selected keyframes.

Overall, the three families of methods occupy distinct operating points: YOLO provides the strongest supervision at the highest cost; All-Embeddings offers near-oracle quality with two orders of magnitude lower latency; and the keyframe variants further reduce storage by nearly 30× while preserving most of the utility for low-threshold, presence-style queries.

# 5 Relevant Work

## 5.1 Video Query Acceleration Systems

Several systems aim to reduce the cost of querying long video streams. BlazeIt [4] accelerates aggregate queries by training proxy models that filter candidate frames before invoking heavyweight detectors. NoScope [5] similarly uses specialized models to rapidly answer binary predicates such as object presence. FiGO [2] and VIVA [8] extend these ideas with optimized pipelines and benchmark frameworks. While these approaches achieve substantial speedups, they often require per-query supervision, multiple learned models, or dense feature computation. In contrast, our system relies on lightweight keyframe selection and general-purpose embeddings rather than training task-specific proxies.

## 5.2 Semantic Embedding Models

Multimodal embedding models such as CLIP [7] learn joint representations of text and images using contrastive training on large-scale datasets. These models enable zero-shot classification and semantic similarity search without domain-specific supervision. Such representations have been widely adopted for retrieval, indexing, and dataset exploration. Our work leverages CLIP as a universal visual encoding mechanism, allowing text queries to be matched directly against keyframe embeddings.

## 5.3 Nearest Neighbor Search

Efficient retrieval of high-dimensional vectors has been heavily studied in the context of large-scale machine learning and recommendation systems. FAISS [3] provides optimized implementations of exact and approximate nearest neighbor search, including inverted-file indexes, product quantization, and GPU-accelerated search. These tools form the basis for scalable vector search systems. Our system employs FAISS to enable fast similarity search over CLIP embeddings corresponding to selected keyframes.

## 5.4 Temporal Redundancy and Keyframe Methods

Classical video-processing techniques exploit temporal redundancy to reduce per-frame computation. Frame differencing, optical flow (e.g., Farnebäck flow [1]), and background subtraction (MOG2 [9]) identify frames containing significant visual or motion changes. These methods have been widely used in video compression, surveillance, and streaming analytics. Our work incorporates these techniques as alternative keyframe selectors to study how motion- and appearance-based heuristics influence retrieval effectiveness. The VIRAT dataset [6], a standard benchmark for surveillance video analysis, provides the long, continuous sequences used in our evaluation.

# 6 Limitations and Future Work

While our prototype demonstrates that lightweight keyframe selection and embedding-based retrieval can significantly reduce storage and computation, several limitations remain.

First, the quality of retrieval is constrained by the semantic resolution of CLIP embeddings. CLIP does not encode fine-grained quantitative attributes such as object counts or precise spatial relationships, which limits its ability to answer queries requiring numerical accuracy. Incorporating specialized counting models or region-level embeddings is a potential extension.

Second, our keyframe selectors rely on simple heuristics based on frame differences, motion magnitude, or background subtraction. These methods can fail in scenes with low motion, gradual lighting changes, or persistent occlusions. A promising direction is to explore learning-based or adaptive thresholding techniques that adjust sensitivity based on scene dynamics.

Third, retrieval performance depends on the density and distribution of selected keyframes. Sparse sampling may miss short-lived events, while aggressive sampling reduces the benefits of compression. Future work could consider dynamic sampling policies that optimize keyframe density based on query patterns or content characteristics.

Finally, our evaluation focuses on a single dataset and a limited set of queries. Broader experimentation across diverse environments, query types, and embedding models would strengthen the generality of our findings. An additional avenue is to integrate temporal reasoning or sequence-level embeddings to better capture activity context beyond individual frames.

# 7 Conclusion

This work presents a lightweight pipeline for video retrieval that combines simple keyframe selection techniques with general-purpose visual embeddings to reduce storage and computation while supporting semantic, text-driven queries. By evaluating multiple motion- and appearance-based keyframe selectors and indexing the resulting frames using CLIP and FAISS, we show that substantial compression is achievable with limited degradation in retrieval quality. Our results highlight the trade-off between keyframe density and semantic recall, as well as the complementary strengths of different selectors across varied scene conditions.

More broadly, our approach demonstrates that effective video exploration does not require dense frame processing or task-specific learned models. Instead, a small number of representative frames, paired with strong pre-trained embeddings, can offer a practical and scalable alternative for interactive video analytics. We hope these

findings motivate further work on lightweight, embedding-centric pipelines for large-scale video systems.

## Division Of Work

Karvy led the initial analysis of CLIP embeddings (including PCA-based exploration) and designed and implemented the keyframe selection methods, evaluating multiple heuristic detectors (Frame Difference, SSIM, MOG2, and Optical Flow). Raj implemented the end-to-end ingestion and inference pipelines, including the YOLO-based oracle runs and the CLIP+MLP training pipeline for count prediction. Both authors jointly integrated the keyframe detectors into the full LensDB pipeline, conducted experimental analysis on PACE, and collaborated closely on project scoping, system design, and writing of the report.

Generative AI was used to gain familiarity with new ideas and exploring the literature. We used generative AI tools for editing and language suggestions; all technical design and analysis are our own.

## References

[1] Gunnar Farnebäck. 2003. Two-Frame Motion Estimation Based on Polynomial Expansion. In *Proceedings of the Scandinavian Conference on Image Analysis (SCIA)*. Springer, 363–370.

[2] Kevin Hsieh, Yale Song, Balaji Vijayaraghavan, and Shivaram Venkataraman. 2018. FiGO: Fast Inference for Object Queries in Video Analytics. In *Proceedings of the ACM Symposium on Cloud Computing*. 1–15.

[3] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale Similarity Search with GPUs. *IEEE Transactions on Big Data* (2019).

[4] Daniel Kang, Peter Bailis, and Matei Zaharia. 2020. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-based Video Analytics. In *Proceedings of the VLDB Endowment*, Vol. 13. 533–546.

[5] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. In *Proceedings of the VLDB Endowment*, Vol. 10. 1586–1597.

[6] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jonggun Lee, Sandip Mukherjee, JK Aggarwal, Hyungtae Lee, and Larry Davis. 2011. A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 3153–3160.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, et al. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[8] Deva Ramanan, Oncel Tuzel, and Pedro Felzenszwalb. 2011. VIVA: A Video Analytics Benchmark. In *2011 IEEE Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 212–219.

[9] Zoran Zivkovic. 2004. Improved Adaptive Gaussian Mixture Model for Background Subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, Vol. 2. IEEE, 28–31.