High-Throughput Ingestion for Video Warehouse: Comprehensive Configuration and Effective Exploration

BAIYAN ZHANG[*] and ZEPENG LI[*], College of Computer Science and Technology, Zhejiang University, China
DONGXIANG ZHANG[†][†], The State Key Laboratory of Blockchain and Data SecurityZhejiang University, China
HUAN LI, Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security, China
TAN KIAN-LEE, School of Computing, National University of Singapore, Singapore
GANG CHEN, The State Key Laboratory of Blockchain and Data SecurityZhejiang University, China

The innovative concept of Video Extract-Transform-Load (V-ETL), recently proposed in Skyscraper, reinterprets large-scale video analytics as a data warehousing problem. In this study, we aim at enabling real-time and high-throughput ingestion of hundreds of video streams and maximizing the overall accuracy, by constructing a proper ingestion plan for each video stream. To achieve the goal, we construct a comprehensive configuration space that takes into account the configurable components in the entire ingestion pipeline, including numeric parameters and categorical options such as visual inference model selection. The new space is 1×10^7 times larger than existing approaches, rendering them as sub-optimal points in our space. To effectively explore the huge and heterogeneous configuration space, we devise an accuracy-aware search strategy based on graph embedding and reinforcement learning to establish the runtime-quality Pareto frontier. To reduce the configuration exploration cost for all video streams, we cluster video streams with similar contexts and adopt mixed integer programming to maximize the overall ingestion accuracy while ensuring the real-time ingestion requirement. In the experimental evaluation with one NVIDIA GeForce RTX 4090 GPU card, our Hippo can support real-time ingestion with 300 video streams and secures an ingestion accuracy that exceeds its competitors by more than 30%.

CCS Concepts: • **Information systems → Multimedia databases.**.

Additional Key Words and Phrases: video ingestion, optimal configuration

ACM Reference Format:
BAIYAN ZHANG, ZEPENG LI, DONGXIANG ZHANG, HUAN LI, Tan Kian-Lee, and GANG CHEN. 2025. High-Throughput Ingestion for Video Warehouse: Comprehensive Configuration and Effective Exploration. *Proc. ACM Manag. Data* 3, 3 (SIGMOD), Article 169 (June 2025), 27 pages. https://doi.org/10.1145/3725407

[*]Both authors contributed equally to this research.
[†]DONGXIANG ZHANG is the corresponding author.

Authors' Contact Information: BAIYAN ZHANG, zhangbaiyan@zju.edu.cn; ZEPENG LI, lizepeng@zju.edu.cn, College of Computer Science and Technology, Zhejiang University, China; DONGXIANG ZHANG, The State Key Laboratory of Blockchain and Data SecurityZhejiang University, China; HUAN LI, Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security, China; Tan Kian-Lee, School of Computing, National University of Singapore, Singapore; GANG CHEN, The State Key Laboratory of Blockchain and Data SecurityZhejiang University, China.

1 Introduction

The increasing reliance on video data for real-time monitoring and analysis has made large-scale video stream processing essential in a wide range of applications, including urban planning, autonomous driving, and smart city infrastructure. For instance, in the context of urban traffic management, city planners and authorities can leverage video feeds from traffic cameras to monitor traffic flow and congestion across multiple locations in real-time. By analyzing these video streams, valuable insights into traffic bottlenecks can be gained, enabling timely interventions to alleviate congestion. To address this, recent advancements have introduced the concept of Video Warehouse [18], proposed by the MIT database research group for scalable video data management and query processing. It relies on the paradigm of Video Extract-Transform-Load (V-ETL) to extract videos from the cameras, transform the unstructured video frames into structured information using visual inference models, and load the ingested results into a traditional DBMS to support diversified video queries. For examples, OTIF [4] harness multi-object tracking (MOT) techniques to yield the trajectories of all moving objects from raw videos. This step can be considered as a manner of view materialization, which stores the results of multi-object tracking query to facilitate online processing of other queries. The output of MOT can be represented as ⟨`oid`, `labels`, `interval`, `bboxes`⟩, where `interval` refers to the starting and ending frames that contain the individual moving object and `bboxes` indicates the list of bounding box information that captures the object's enclosed bounding box within each relevant video frame. With the extracted trajectories, offline indexes (e.g., inverted index, temporal index and spatial index) can be constructed to answer selection, aggregation and top-k queries with sub-second latency. Skyscraper [18] considers the MOT-based ingestion in the cloud environment with a budget constraint. By automatically tuning the parameter configurations, it allows for cheap video ingestion while adhering to throughput requirements.

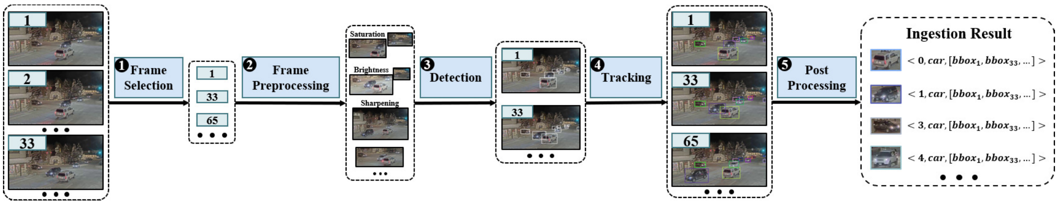
1
2
33
❶ Frame Selection
1
33
65
❷ Frame Preprocessing
Saturation
Brightness
Sharpening
❸ Detection
1
33
❹ Tracking
1
33
65
❺ Post Processing
Ingestion Result
$< 0, car, [bbox_1, bbox_{33}, ...] >$
$< 1, car, [bbox_1, bbox_{33}, ...] >$
$< 3, car, [bbox_1, bbox_{33}, ...] >$
$< 4, car, [bbox_1, bbox_{33}, ...] >$

Fig. 1. The pipeline of multi-object tracking (MOT) for video ingestion.

In this paper, we follow the V-ETL paradigm with MOT-based ingestion to transform video frames into structured object trajectories. The objective is to enable real-time and high-throughput ingestion of large-scale video streams, while maximizing the overall accuracy of the derived trajectories. We achieve the goal from the perspective of configuration space design and autonomous database tuning. In relational databases, this component is of paramount importance for performance optimization, as verified in database tuning systems like CDBTune [34], QTune [21] and UniTune [36]. In video databases, it also plays a a critical role in managing the trade-off between the ingestion time cost and result quality when dealing with large-scale video streams. Nonetheless, our observations indicate deficiencies in the configuration space design and exploration approaches employed by OTIF and Skyscraper. In the following, we elaborate these shortcomings and present our approach to support high-throughput V-ETL.

Table 1. Comparison of configuration spaces among OTIF, Skyscraper and Hippo. The column "Module" refers to the functional component of the MOT pipeline in Figure 1.

Method	Module	Parameter	Options	Space Size
OTIF	❶	*sampling_rate*	7	4375
	❷	*image_resolution*	5	
		ROI_resolution	5	
		ROI_threshold	5	
	❸	*detect_threshold*	5	
Skyscraper	❶	*sampling_rate*	4	96
	❷	*image_resolution*	2	
	❸	*detect_model_size*	3	
	❹	*track_max_loss_time*	4	
Hippo	❶	*sampling_rate*	8	5.6×10^{10}
		frame_filter_enable	2	
		frame_filter_resolution	5	
		frame_filter_threshold	5	
	❷	*image_resolution*	8	
		ROI_enable	2	
		ROI_resolution	5	
		ROI_threshold	5	
		enhancement_tools	4	
	❸	*detect_model*	10	
		detect_model_size	3	
		compression_strategies	4	
		detect_threshold	3	
	❹	*track_model*	5	
		track_max_loss_time	4	
		track_keep_threshold	3	
		track_stop_threshold	3	
		track_switch_threshold	3	
	❺	*refine_prefix-suffix*	2	
		noise_filter	4	

1. Narrow Configuration Space. Both OTIF and Skyscraper establish a configuration space for automatic parameter tuning, seeking to enhance the performance of video ingestion based on multi-object tracking. However, their configuration space is very narrow. First, as illustrated in Table 1, they focus on a limited set of parameters, such as sampling rate and input resolution, to adjust the trade-off between the time cost of ingestion and the quality of results. Other hyper-parameters with the potential to significantly influence the end-to-end performance (e.g., threshold for object ??) are overlooked. Second, they take into account numeric attributes and fail to consider categorical attributes. For example, there exist alternative object detection and tracking mechanisms proposed in the computer vision community, which have not been considered. We can create a categorical knob to incorporate these missing methods and allow tracking model selection to further enhance performance. Third, they cover a subset of modules in the pipeline, without consideration for the configuration space relevant to object tracking and post-processing stages.

Our first contribution is to devise a comprehensive configuration space (CCS). It covers the entire ingestion pipeline in Figure 1 and renders the configuration of OTIF and Skyscraper as our subspace. In other words, their optimal configurations correspond to the sub-optimal points in our new space. Moreover, our enlarged configuration space demonstrates two distinctive properties that pose challenges for OTIF and Skyscraper:

(1) **Comprehensive**. The configuration space is magnified by a factor of 10 million, necessitating the implementation of effective exploration strategies in such a huge space.
(2) **Heterogeneous.** In contrast to OTIF and Skyscraper, which only consider numeric attributes, our configuration space incorporates categorical attributes. For example, we provide 10 options for object detection models.

2. Ineffective Space Exploration. OTIF and Skyscraper adopt hill-climbing algorithm [31] for configuration space exploration. They begin with a slow but accurate configuration and iteratively update different subsets of parameters that offer a speedup. In essence, their exploration strategy is to walk along the Pareto curve to find the optimal trade-off between efficiency and accuracy. Despite the promising performance achieved in their original configuration space, they fail to work in our enlarged space for two main reasons. First, their local search methods lack backtracking mechanisms and can get stuck in poor local optima. When the search is huge, they exhibit slow convergence and it may take very long to reach optima. Second, these local search algorithms start from a candidate configuration and then iteratively move to a neighbor configuration. However, the presence of categorical parameters introduces additional challenges for neighborhood exploration. For instance, there exists a binary configuration knob in Hippo, with options {`yes`, `no`} to determine whether the frame filtering strategy is applied. It is then difficult to determine the neighborhood relationship among configuration instances with this attribute.

Addressing the shortcomings of heuristic search, we utilize reinforcement learning (RL) to explore the configuration space, which have been adopted by traditional database tuning algorithms. However, their training objective is relatively straightforward, as they are trained with single objective of minimizing query processing latency. In the scenario of video warehouse, we need to take into account both efficiency and accuracy. The dual-objective training scheme to establish the runtime-quality Pareto frontier is significantly different from these RL-based optimization approaches in relational database.

Our approach is to propose a new RL training framework to guide agents in exploring configurations that meet Pareto constraints within the configuration space. We employ a graph-driven deep learning approach to model the embedding of the configuration space, uncovering the complex relationships among different dimensional parameters of the ingestion configurations to provide a foundation for accurate configuration search. Additionally, we propose an imitation learning scheme that designs expert models to guide the training of agents, reducing the variance of agent policy. Meanwhile, the expert models constructed by the best-performing setups of agent historical trajectory do not affect the training convergence of agents.

With the trained RL-based agents, we propose an integrated system Hippo for high-throughput ingestion from hundreds of video streams, which works in three main steps. First, it uses an efficient surrogate model to extract the context of the video streams and clusters the streams with similar contexts into a group. Streams within the same cluster share a video parsing configuration, significantly reducing the use of computational resources. Second, agents trained through our reinforcement learning framework search for a set of video parsing configurations for a representative stream in each cluster, incurring a one-time cost. This set contains multiple configurations varying in accuracy and efficiency. Finally, based on the available computational resources Hippo actively sensed and the needs for real-time parsing in parallel, we employ integer linear programming to determine the optimal parsing configuration for each video stream cluster. Our system was tested under real surveillance video streams, and the experimental results show that with one NVIDIA GeForce RTX 4090 GPU card, and even with one NVIDIA GeForce RTX 3090Ti GPU, our Hippo can support real-time ingestion with 300 video streams and remarkably improve ingestion quality compared to prior methods. We release the Hippo source code at https://anonymous.4open.science/r/Hippo-B74F.

The remainder of this paper is organized as follows. We review video query optimization systems and automatic database tuning in Section 2. We formulate the ingestion problem and its objective in Section 3. The configuration space is explicitly introduced in Section 4 and we propose the system Hippo and iits space exploration strategy in Section 5. We conduct extensive experiments in Section 6 and conclude the paper in Section 7.

2 Related Work

In this section, we review video query optimization systems, with a particular emphasis on video DBMS (VDBMS) with the component of autonomous parameter tuning. We also examine autonomous DBMS tuning in the domains of relational databases.

2.1 Video Query Optimization Systems

On-the-fly Query Optimization. NoScope [15] pioneers the use of smaller, faster specialized networks to improve model inference speed. It exploits redundancy in neighboring frames, via training a difference detector for irrelevant frame filtering. Its concept of proxy model design for a better efficiency-accuracy tradeoff has been widely adopted, as seen in works like FOCUS [10], TAHOMA [2], ABAE [16], Everest [20], and FiGO [6]. For instance, TAHOMA replaces the accurate-but-expensive convolutional neural networks (CNN) with cascades of fast image classifiers, identifying Pareto-optimal cascades with varying trade-offs. Everest [20] trains a lightweight convolutional mixture density network to approximate score distributions for frames, using uncertain query processing to accelerate top-k analytics with probabilistic guarantees [7, 24]. VIVA [17] uses relational hints [33] to optimize complex video analytics queries and exploits the optimal generation of query execution plan.

Offline Ingestion and Indexing. OCUS [10] and Video-zilla [11] accelerate video query processing using an offline index. They employ lightweight CNNs to extract class labels. Subsequently, semantically similar frames are clustered and inverted index [41] is built to improve efficiency. When supporting aggregation queries, BlazeIt [14] consists of an ingestion step to randomly sample a subset of video frames [22] and annotate them using the expensive CNN models. A specialized neural networks (NN) is trained from the images as well as their labels to estimate statistics, whose variance is further reduced via the technique of control variates [9]. Both OTIF [5] and Skyscraper [18] seek to extract object tracks from unstructured videos and store them in a relational database to support online queries. In OTIF, optimization techniques, including segmentation proxy models and recurrent reduced-rate tracking, are proposed to accelerate processing.

2.2 Autonomous Tuning in Video DBMS

Chameleon [12] is the first work that addresses the significant dynamics in the impact of neural network configuration on video analytics accuracy. The work introduces a suite of heuristic techniques for periodic re-profiling. LLAMA [30] is a heterogeneous and serverless framework for auto-tuning video pipelines. It enumerates all possible configurations and dynamically runs a cost-based optimizer to assign configurations across heterogeneous hardware. OTIF [5] and Skyscraper [18], developed by the same research team, employ hill-climbing algorithms [31] to explore configurations. All these works consider a limited configuration space, and their exploration strategies fall short in the context of a vast configuration space.

2.3 Autonomous Tuning in Relational DMBS

Bayesian optimization (BO) and reinforcement learning (RL) are two types of mainstream approaches to autonomous tuning in relational database management system(RDBMS). Among the BO-based category, OtterTune [39] adopts Gaussian process as the surrogate model in BO and employs a workload mapping strategy to leverage past experiences. OnlineTune [1] employs Gaussian process to identify risky configurations in various states based on the database context, so as to ensure exploration safety. However, the performance of these BO approaches degrades in high-dimensional and complex configuration space [1, 38]. As to RL-based methods, CDBTune [34] uses the reward feedback mechanism in reinforcement learning to replace traditional regression, which makes end-to-end learning possible, and improves the efficiency of online tuning. QTune [21] employs RL to utilize integrated query vectors and database status to the database configuration adjustments. UniTune [37] considers the dependencies among the multiple agents designed for different optimization components for DBMS (e.g., data partition [32], index selection, knobs tuning, etc). It adopts Thompson Sampling with a memory buffer for agent selection and resource allocation.

3 Problem Definition

In this paper, we follow the paradigm of Video Extract-Transform-Load (V-ETL) [18], where video frames are extracted from the cameras, transformed into an intermediate format using visual inference models, and eventually loaded into a relational database to facilitate video SQL query processing. The primary research objective is to optimize the expensive V-ETL Transform step, with particular emphasis on multi-object tracking (MOT) based video ingestion, which is a key primitive in numerous video analytical pipelines [5, 18].

Definition 3.1 (MOT-based Transformation). Given a sequence of frames from a video stream, the MOT-based transformation results in the trajectories of all moving objects, each being represented as ⟨`oid, labels, interval, bboxes`⟩, where `labels` refer to the class labels associated with the detected object `oid`, `interval` indicates the range of frames containing that object, and `bboxes` denote the object's size and position at each relevant video frame.

With the ingested trajectories, offline indexes (e.g., inverted index, temporal index and spatial index) can be constructed to answer selection, aggregation and top-k queries with sub-second latency. For instance, the retrieval of video frames containing both an *ambulance* and a *firetruck* can be effortlessly achieved through a simple intersection operation between the inverted lists corresponding to these two labels. Similarly, estimating traffic flow at a timestamp can be accomplished by leveraging stabbing queries over the interval tree to obtain the number of distinct vehicles.

In our problem formulation, we address the challenge of high-throughput MOT-based transformation, where each GPU card is allocated a fixed number of video streams for real-time ingestion. The objective is to maximize the quality of the extracted video entities by configuring ingestion parameters optimally, in the meanwhile ensuring that the ingestion speed aligns with the arrival rate of streaming data.

Definition 3.2 (Configuration Space). We define the configuration space as a set of parameters $\{\mathcal{D}_k \mid k = 1, \ldots, L\}$, where L is the number of distinct parameters. Within this space, a specific pipeline configuration is represented as a vector $\mathrm{p} \in \mathbb{R}^L$, where the k-th ($1 \leq k \leq L$) component $\mathrm{p}[k]$ denotes the index of a distinct parameter from its associated dimension $\mathcal{D}_k$.

Example 3.3. We use the setting of Skyscraper as an example to explain the concept. The configuration space of Skyscraper contains four dimensions, including sampling rate $\mathcal{D}_1$={1, 5, 30, 60}, number of image tiles $\mathcal{D}_2$={1×1, 2×2}, number of historical frames used for tracking $\mathcal{D}_3$={1, 2, 3, 5} and model size $\mathcal{D}_4$={`small`, `medium`, `large`}. An instance of configuration {*sampling rate*: 5, *image titles*: 2×2, *historical frames*: 5, *model size*: `small`} corresponds to a vector $\mathrm{p} = [1, 2, 3, 0]$.

Definition 3.4 (Real-time Ingestion Constraint). Let $f(\mathrm{c}_i)$ denote the number of frames generated from video stream c_i per second. Given an MOT-based ingestion operator p with sampling rate r and processing speed fps, we denote $g(\mathrm{p}) = r \cdot fps$. The real-time ingestion constraint requires that $g(\mathrm{p}) \geq f(\mathrm{c}_i)$.

Example 3.5. For a video stream c_i producing video data at a standard frame rate of 30 fps (frames per second), the prior configuration $\mathrm{p} = [1, 2, 3, 0]$ for Skyscraper in Example 3.3 satisfies the real-time ingestion constraint as long as its processing speed reaches $\frac{30}{5} = 6$ frames per second.

Definition 3.6 (Problem Formulation). Given a collection of video data streams $C = \{\mathrm{c}_1, \ldots, \mathrm{c}_N\}$, we aim at identifying an optimal set $\mathcal{P} = \{\mathrm{p}_1, \ldots, \mathrm{p}_M\}$ for the ingestion pipeline to maximize the accuracy of ingested trajectories, without violating the real-time ingestion constraint:

$$\begin{aligned} \text{maximize} \quad & \sum\nolimits_{i=1}^{N} accu(\mathrm{p}_j, \mathrm{c}_i) \quad \text{where} \quad \mathrm{p}_j \in \mathcal{P}, \mathrm{c}_i \in C \\ \text{subject to} \quad & \sum\nolimits_{j=1}^{M} g(\mathrm{p}_j) \geq \sum\nolimits_{i=1}^{N} f(\mathrm{c}_i) \end{aligned} \tag{1}$$

Here, $accu(\mathrm{p}_j, \mathrm{c}_i)$ measures the ingestion quality (i.e., the tracking accuracy) for video stream c_i using configuration p_j for a more practical setting instead setting up uncontrollable video ingestion accuracy for all streams. Even if the two aspects are reversed, the techniques remain applicable, as the optimization algorithm is designed to build the Pareto curve balancing both accuracy and efficiency.

It is worth noting that $\sum_{i=1}^{M} g(\mathrm{p}_j) \geq \sum_{i=1}^{N} f(\mathrm{c}_i)$ is a soft constraint. If the constraint is violated, we can employ the mechanism proposed in Skyscraper, utilizing extra buffer resources to mitigate the problem. Although this post-processing procedure is beyond our research scope, we will nonetheless assess the violation rate in our experimental study, as we prefer an ingestion scheme with a lower violation rate.

4 Configuration Space

Autonomous database tuning has been shown to be an effective way for performance improvement. In the context of video database, previous studies such as OTIF [4] and Skyscraper [19] have been pioneering works to leverage automatic parameter tuning to filter out suboptimal knob configurations that do not lie on the runtime-quality Pareto frontier. However, these studies were constrained by a limited configuration space. In response, this work introduces the Comprehensive Configuration Space (CCS), as illustrated in Figure 2, which extends beyond prior efforts in several critical aspects.

First, our configuration space covers the entire pipeline, including the cross-frame object association algorithm and the component of post-processing to further refine the extracted object trajectories. While OTIF and Skyscraper employ RNN-based strategies for object association, there exist multiple alternative object association strategies as proposed by the computer vision community. We incorporate cross-frame object association as a configurable knob in the configuration space, offering 5 options for algorithm selection. Furthermore, we recognize the often-neglected importance of post-processing for trajectory refinement and thus include it in the configuration space as well.

Second, we incorporate more configurable numeric parameters among different components of the ingestion pipeline. These parameters, previously unexplored by OTIF and Skyscraper, play considerable impact on the ingestion performance. For example, , within the cross-frame object

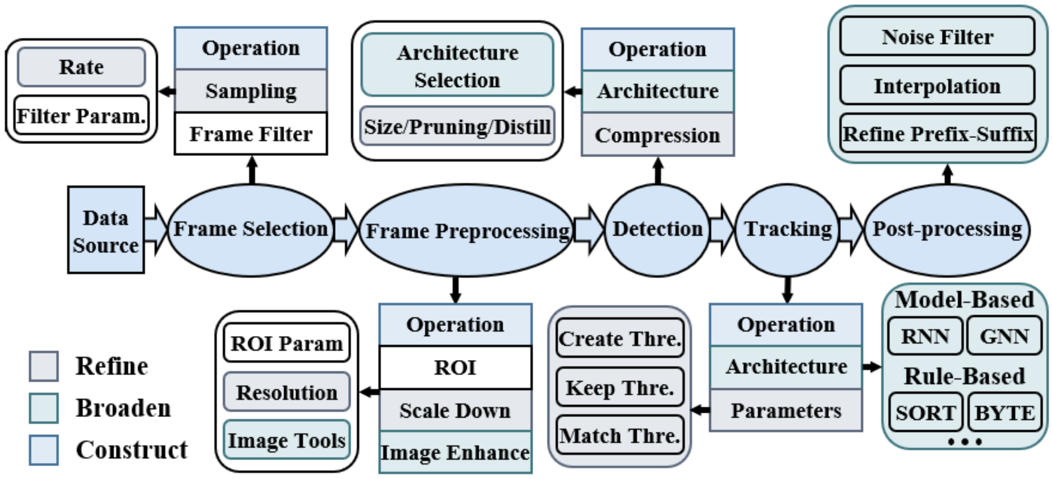

Fig. 2. Comprehensive configuration space for Hippo.

association module, multiple threshold parameters are pivotal. These include multiple threshold parameters to determine the criteria of initializing a new tracklet, filtering noisy bounding boxes of detected objects, and determining matches between two objects.

Third, besides the expansion of numeric parameters, we also enrich the configuration space with a variety of discrete design options. Within the object detection knob, we provide 10 detection models, offering distinct trade-offs between efficiency and accuracy. For instance, one-stage detectors such as YOLO family [23, 28] provide rapid detection capabilities, whereas two-stage detectors such as Faster R-CNN [29], though less efficient, ensure more robust performance.

These enhancements have allowed us to develop a configuration space vastly more extensive than that of OTIF and Skyscraper, rendering their configuration space as our subspace and providing us with more opportunities for performance optimization.

5 System overview

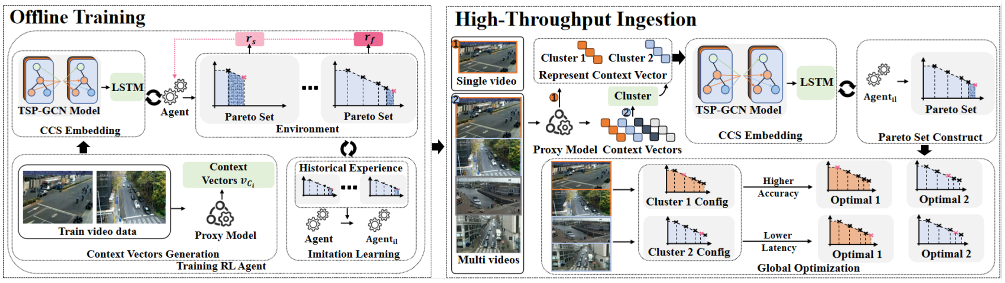

Fig. 3. The overview of the Hippo.

Based on the comprehensive configuration space (CCS) defined in the Section 4, the overview of the system Hippo is shown in Figure 3. Hippo consists of two primary components, including the RL-based offline training and high-throughput ingestion for different video streams process.

The first component focuses on training RL agent to explore the Pareto configuration set in the huge configuration space for the target video streams. The three key elements of RL are defined as following. The state is represented by the current Pareto configuration set that has been explored and context vectors extracted from video streams. To construct state, we embed the Comprehensive Configuration Space of the Pareto configuration set, introducing a Type-Stage-Performance (TSP) composite graph and leveraging Graph Convolutional Network (GCN) that captures complex interactions between parameter types, ingestion stages, and collaborative performance. The action space includes configuration changes of each video ingestion configuration in the Pareto set. Since we need to take into account ingestion efficiency and accuracy simultaneously, the reward is defined as the increase in the Area Under Curve (AUC) from the two dimensions ingestion efficiency and accuracy when a new configuration expands the Pareto set. The agent's performance is evaluated by comparing AUC of the current Pareto set, which reflects how well the agent has optimized configurations over time. To tackle the training barrier of huge configuration space and enhance training convergence, we employ imitation learning to leverage historical configuration data and integrate previous experience into the agent's training process. Specifically, we aggregate previously explored configurations to construct a Pareto set, which serves as a reference to guide the agent. Through imitation learning, the agent gradually learns to generate new configurations that either match or surpass the performance of the historical set of Pareto set, ultimately optimizing its strategy and evolving into agent with imitation learning $agent_{il}$. As the agent continues to explore the search space, more diversified parameter combinations are examined to improve performance.

With the trained RL agents, the high-throughput ingestion scheme leverage them to assign proper configurations for the incoming video streams. Ultimately, each video stream is associated with a tailored ingestion scheme with different visual models and parameters. In order to support hundreds of video streams and reduce the cost of configuration space exploration, we first cluster video streams and perform RL-based configuration search strategee only for the representative video stream in the cluster. The other streams located within the same cluster will share the same configured ingestion plan. The output of RL-agent against each representative video stream in a cluster is a set of possible Pareto configurations, with varying efficiency and accuracy. In the global optimization module, we need to take into account the constraint of global resources. For instance, the GPU memory consumption of the ingestion plans on the hundred-scale video streams cannot exceed to total memory capacity. Hippo integrates resource-awareness into the design of video ingestion workflows by estimating GPU memory resource consumption for the derived configurations in the Pareto set. The global optimization problem can be formulated as a linear programming problem and can be easily solved by a standard linear optimization solver, with the objective of maximizing the overall ingestion accuracy, under the constraints of real-time processing and GPU memory capacity.

5.1 Offline Training

5.1.1 Pareto Configuration Set Exploration. Due to the varying efficiency and effectiveness of identical configurations across different video streams, global real-time ingestion requires the selection of configurations based on a trade-off between configuration efficiency and accuracy rather than merely choosing the configuration with the highest accuracy for the video stream. We identify suitable configurations by constructing a Pareto configuration set, as the advantage of the Pareto set lies in the absence of dominance relationships among its configurations, and to construct the Pareto configuration set from the CCS, instead of heuristic algorithms such as traditional hill-climbing used by previous video database systems [4, 13, 19], we employ reinforcement learning (RL) to train an agent to search for configuration for video streams across CCS. However, the complex of machine learning operators or parameter selection and the association between them make the relationship between configuration and performance more complex. To address this, we construct a composite graph structure of Type-Stage-Performance (TSP) and use deep learning techniques to model the configurations embedding, delving deeper into the interactions between different parameter types and modules within the configuration space. Over time, the agent uses this embedding to explore a wider range of parameter combinations, uncovering potentially better configuration spaces. Searching for configuration is a one-time cost. Once the optimal configuration is identified, the system can process and ingest video streams with minimal delay. Hippo processes these streams in parallel, identifying suitable configurations for each stream as they arrive, with immediate responses provided.

5.1.2 Context Vectors Generation. In traditional database systems, context information is typically derived from structured data, which can be efficiently processed for training agents. However, in video databases, extracting context from unstructured video streams is more complex. Previous methods, such as Skyscraper [19], rely on expensive models that perform full object detection across all frames to construct context vectors. To address this challenge, we propose a more efficient feature extraction method using lightweight proxy models. These models are significantly faster than previous ones and use selective frame sampling and simplified detection processes to extract nine key features, reducing both time cost and computational load.

We define nine meta-features related to video ingestion, capturing key aspects like video texture, object motion, and object information. These features are extracted using simple, readily available tools, as shown in Table 2. Video texture information (1-3) is calculated with OpenCV, reflecting image quality and guiding configuration search for video input models. Basic object information (4-5) is obtained using YOLO [23], based on object size and count in each frame, influencing configuration search for detection models. Object motion information (6-9) is derived from ByteTrack [40], using average speed and acceleration, as well as their standard deviations, impacting the tracking and post-processing configuration.

In our approach, we increase the sampling rate while reducing resolution and applying low-cost image processing. The ByteTrack algorithm, leveraging YOLO's detection results, ensure minimal computational cost. We refer to the simplified versions as YOLO_s and ByteTrack_s. Our method takes into account effectiveness, diversity and efficiency when selecting features from video streams. In other words, we require these features to be discriminative among different video streams and cover different aspects such as scene-level and object-level. In addition, these features can be effectively extracted, without incurring expensive computation overhead. Several proxy models are typically used together, and in the actual video context vector generation process, we measured their fps during video stream pro- cessing: OpenCV achieved 105,000 fps, YOLO_s reached 16,000 fps, and ByteTrack_s achieved 45,000 fps, while the golden model from previous methods only managed 150 fps in real-world processing.

Specifically, we select one minute of video from each stream to extract context features, covering the relevant information in surveillance scenarios. This fixed-length clip helps address the short-term repetitive patterns typically found in surveillance videos. The video texture and basic object information of the video clip are obtained by averaging across all frames.

5.1.3 Configuration Space Embedding. Configuration embedding is key to efficiently searching for the optimal settings for pending data. Traditional database configuration tuners create an embedding vector by concatenating parameter values across different dimensions to determine the best settings. However, this approach is not suitable for video databases for several reasons. First, video databases involve a wide variety of parameters, such as thresholds, booleans, and selections. Second, the interaction between parameters is complex, for instance, video preprocessing affects the object detector, and frame selection impacts the object tracker.In recent years, graph embedding

Table 2. Contextual feature vector for clustering.

Feature ID	Attribute Name	Model Name	Efficiency (fps)
1	Video Clarity	OpenCV	105K
2	Video Brightness		
3	Video Saturation		
4	Mean Size	YOLO_s	16K
5	Mean Number		
6	Mean Speed	ByteTrack_s	45K
7	Std Speed		
8	Mean Acceleration		
9	Std Acceleration		

models have proven to be effective across various domains, such as pattern match [42], database storage [8], query optimization [26] and social network analysis [27]. For these, this paper proposes a novel Type-Stage-Performance (TSP) composite graph $\mathcal{G}^{TSP}$ for configuration space embedding in video databases, as shown in Figure 4, introducing the first application of this approach in the context of large-scale video stream processing. It constructs three distinct graphs by representing different parameters as graph nodes and forming edges based on the parameter type, ingestion stage, and the performance of parameter collaboration. Subsequently, we employ graph convolution models to mine the representations of the TSP graph, ultimately integrating them into an embedding of the configuration.

We further detail the definition of the TSP graph and embedding model, which utilizes a set of nodes representing all parameters. The edges of these graphs model relationships from three perspectives: parameter type, stage, and collaboration, forming three graphs with shared node representations. Firstly, the graph $\mathcal{G}^T$ of different parameter types is directed, with edges built based on parameter types where boolean parameters influence all parameters within the same module; selection parameters determine the choice of algorithm-specific parameters, and threshold parameters interact with other threshold parameters within the same module. Similarly, the graph $\mathcal{G}^S$ of different parameter stages remains directed, with edges defined by the influence of parameters at different stages, such as frame gap primarily affecting the object tracker parameters and not the detector parameters. Lastly, a graph $\mathcal{G}^P$ representing the performance of parameter collaboration is undirected, with dynamically updated edge weights based on the combined performance of parameters, initially set to zero and updated through training to reflect the average historical performance of configurations.

Formally, all parameters in the CCS are initially mapped to fixed-dimensional hidden vectors $V_{node} = \{v_1^1, v_2^1, \ldots, v_{|\mathcal{D}_L|}^L\}$ representing nodes in the composite graph structure, (where v_i^j represents the embedding of the j parameter in the i dimension of the configuration vector). Three graph convolution models GCN^j (for $j \in \{S, T, P\}$) composed of a two-layer graph convolution structure are used to extract features from the nodes under different graph edges. Given the configuration vector $\mathrm{p} = [p_1, p_2, \ldots, p_L] (p_k \in [0, |\mathcal{D}_k|), 1 \le k \le L)$, the embedding list e^j extracted from different graphs can be indexed. These embeddings are then concatenated to represent the parameter embedding e_{pm} of the configuration vector. The entire configuration embedding e_c is modelled using a bidirectional long short-term memory (LSTM) network to aggregate the features of different dimensions, effectively capturing the dependencies within the parameter sequence. And to manage the relationships between parameters in the reinforcement learning model, we first use TSP graph to embed the configurations, capturing the relationships between different parameters. These configurations are then processed through a bidirectional LSTM model to extract the feature representation of the current video stream configuration space for agent. The complete computational process is outlined as follows.

$$V^j = GCN^j(\mathcal{G}^j, V_{\text{node}}) \quad \text{for } j \in \{S, T, P\} \tag{2}$$

$$\mathrm{e}^j = [\hat{v}_{p_1}^1, \hat{v}_{p_2}^2, \dots, \hat{v}_{p_L}^L]^j \quad \text{for } j \in \{S, T, P\} \tag{3}$$

$$\mathrm{e}_{pm} = \mathrm{Concat}(\mathrm{e}^S, \mathrm{e}^T, \mathrm{e}^P) \tag{4}$$

$$\mathrm{e}_c = \mathrm{LSTM}(\mathrm{e}_{pm}) \tag{5}$$

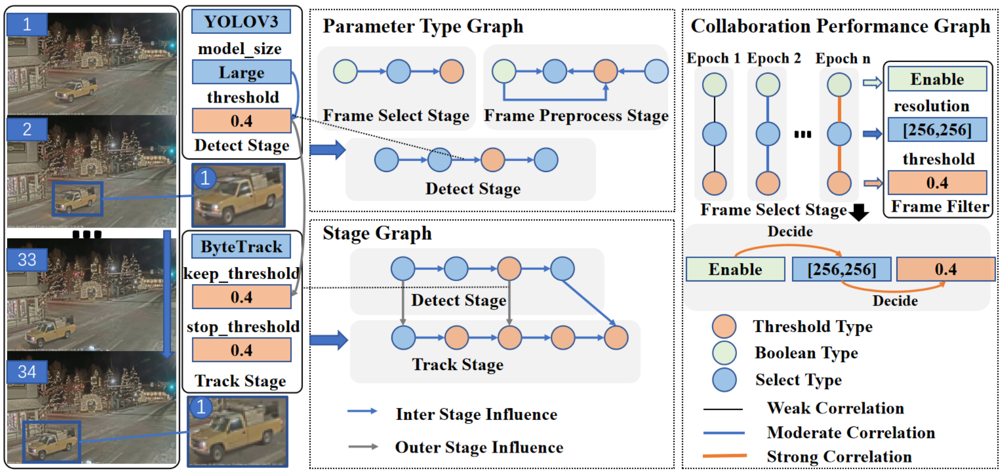

Fig. 4. Type-Stage-Performance (TSP) composite graph, each node represents a different parameter.

5.1.4 *Reinforcement Learning for Agent Training.* Reinforcement learning (RL) is an effective method for exploring complex configuration spaces. Traditional RL-based database tuning algorithms aim to generate a configuration that balances accuracy with efficiency. However, we seek to provide a Pareto configuration set that trades off performance and efficiency for each video stream cluster after clustering, optimizing ingestion accuracy under real-time constraints to tackle manually setting efficiency constraint values. We define a reinforcement learning-based configuration search framework, enabling the agent to be trained to explore the Pareto configuration set within the CCS for a video stream. In the following, we describe six key elements in reinforcement learning and demonstrate their correspondence with the search for Pareto configuration sets.

Agent: The agent can be considered as the configuration search module of the Hippo system, receiving rewards (i.e., the Pareto set $\mathscr{P}^c$ change) and states from the video stream c configuration space, and updating the policy θ to guide the generation of the new configuration for the current video stream c. It aims to achieve higher rewards by producing sets of configurations that offer better trade-off performance while satisfying Pareto optimality.

Environment: The video stream configuration space is the record that is a list of known configurations. The environment $\mathscr{E}^c$ is defined as the configuration space of the current video stream c to be searched.

State: The state primarily reflects the characteristics of the current video stream configuration space, which facilitates determining which configurations from the CSS optimize the configuration Pareto set $\mathscr{P}^c$. Formally, for a video stream c, at step t, it stores a list of configurations

$[(\mathrm{p}_1^{\mathrm{c}}, acc_1, fps_1), \ldots, (\mathrm{p}_t^{\mathrm{c}}, acc_t, fps_t)]$ characterized by accuracy and efficiency metrics. The metrics are based on the actual performance from running a randomly selected video stream. The system employs the configuration embedding modeling method introduced in the Section 5.1.3 to convert these configurations into vectors $V^{\mathrm{c}} = [(e_{c_1}^{\mathrm{c}}, acc_1, fps_1), \ldots, (e_{c_t}^{\mathrm{c}}, acc_t, fps_t)]$. Subsequently, we arrange all configurations in ascending order of efficiency and input them into a bidirectional LSTM model to derive the feature e_{cs}^t of the current video stream configuration space. Finally, we incorporate the context features e_{ct}^t of video stream as part of the environment state to form the state of the configuration space s_t.

$$\mathrm{e}_{cs}^t = \mathrm{Bi\_LSTM}(V^{\mathrm{c}}) \tag{6}$$

$$s_t = \mathrm{Concat}(\mathrm{e}_{cs}^t, \mathrm{e}_{ct}^t) \tag{7}$$

Action: The policy network of the agent provides the configuration within the comprehensive configuration space most likely to improve the Pareto set $\mathscr{P}^c$ of the video stream configuration space based on the current video stream state s_t. For instance, at step t, the action a_t is defined as the new video configuration proposed by the agent. The new video configuration a_t is then tested in the video configuration environment $\mathscr{E}^c$ to determine whether to update the Pareto set $\mathscr{P}^c$ of the video stream configuration space.

Policy: The state of video stream s_t are processed through a two-layer fully connected network, and the multiple heads of agent predict the new configuration a_t in different parameter dimensions. Different heads are fully connected layers of different sizes. The input size is the hidden layer feature size, and the output size is determined by the parameter dimension size.

$$h_t = MLP(ReLU(MLP(s_t))) \tag{8}$$

$$\hat{a}_t^i = MLP^i(h_t) \quad \text{for } i \in \{1, 2, \ldots, L\} \tag{9}$$

$$a_t = [\hat{a}_t^1, \hat{a}_t^2, \ldots, \hat{a}_t^L] \tag{10}$$

Reward: We use the Pareto set of the video stream configuration space as a reference for setting rewards. Since performance cannot be solely judged by accuracy or efficiency, we measure it using the Area Under Curve (AUC) of the Pareto front. The AUC effectively balances the trade-offs between different performance metrics. Specifically, when a new configuration a_t generated by the agent expands the AUC curve within the video stream configuration space $\mathscr{E}^c$, the agent receives a positive reward based on the increase in AUC. At the end of the interaction process, if the Pareto solution set reaches the preset maximum size MAX_I, the agent receives a terminal reward. This reward is the area under the AUC of the Pareto set multiplied by a predetermined weight factor λ greater than 1, emphasizing the importance of achieving a larger solution set. The reward function $r(s_t, a_t)$ at step t is as depicted in the below formula.

$$r(s_t, a_t) = \begin{cases} Max(AUC_{\mathscr{P}_{t+1}^c} - AUC_{\mathscr{P}_t^c}, 0) & \text{if } |\mathscr{P}_{t+1}^c| < MAX\_I \\ \lambda * AUC_{\mathscr{P}_{t+1}^c} & \text{if } |\mathscr{P}_{t+1}^c| = MAX\_I. \end{cases} \tag{11}$$

Interaction Process: The agent interacts iteratively with the environment until the maximum number of interactions or the Pareto set size is reached. In each step, the agent proposes a new configuration a_t based on the current state s_t, aiming to update the Pareto set. During training, we use a policy gradient loss to update the agent's policy. The loss function is as follows:

$$\mathcal{L}(\theta) = -\mathbb{E}_{s\sim\mathscr{P}^c, a\sim\pi_\theta}\left[R(s,a) \cdot \log \pi_\theta(a|s)\right] \tag{12}$$

$$= -\sum_{s,a} \pi_\theta(a|s) \cdot r(s,a) \cdot \log \pi_\theta(a|s), \tag{13}$$

where $\pi_\theta(a|s)$ is the probability of policy network selecting action a (i.e., the new configuration) given state s, and $r(s,a)$ is the reward function as previously defined. This formulation encourages exploration that may enhance the performance of the Pareto set.

5.1.5 Imitation Learning Strategy. Policy gradient loss is commonly used in policy-based reinforcement learning for tasks like database configuration search due to its independence from space size and good convergence. However, it has higher variance than value-based methods, and in our task, discrete dimensions and inter-parameter interactions increase this variance, leading to potential issues like local optima or unstable convergence. To reduce variance, methods like CDBTune [35] and UniTune [36] introduce a critic to stabilize training. However, in our case, the video stream configuration space has a large state space and low inter-state correlation, making it harder for the agent to determine which experiential knowledge to retain.

In such scenarios, effectively utilizing historical experience to guide policy updates is an approach that can smooth variance without increasing training difficulty. Therefore, we propose an imitation learning strategy that leverages historical configuration search data to construct an expert model for each video stream environment, exploiting the stability of the scenes to handle the extremely large search space more effectively. Specifically, for each video stream, we employ imitation learning by constructing an expert based on the historical configurations used in that environment. We aggregate all configurations$[p_1^h, \dots, p_n^h]$ historically used for the corresponding video stream c and construct a Pareto set $\hat{\mathcal{P}}^c$ based on their performance metrics. This Pareto set serves as the expert policy π_e, providing guidance to the agent during training. The imitation learning process involves aligning the agent's policy π_θ with the expert policy π_e by minimizing the Kullback-Leibler (KL) divergence between them. The imitation loss function is defined as:

$$\mathcal{L}_\mathrm{i}(\theta) = \mathbb{E}_{s\sim\hat{\mathcal{P}}^c}\left[D_{KL}(\pi_\mathrm{e}(\cdot|s)\|\pi_\theta(\cdot|s))\right] \tag{14}$$

$$= \sum_s \sum_a \pi_\mathrm{e}(a|s) \cdot \log \frac{\pi_\mathrm{e}(a|s)}{\pi_\theta(a|s)}, \tag{15}$$

where $\pi_\mathrm{e}(a|s)$ is the action probability distribution of the expert in state s, $\pi_\theta(a|s)$ is the action probability distribution under the policy θ of trained agent, and D_{KL} is the Kullback-Leibler divergence, which measures the information loss when the policy modeled by θ is used to approximate the expert policy. By minimizing $\mathcal{L}_\mathrm{i}(\theta)$, the agent learns to imitate the expert's behavior in selecting configurations. In practice, after the agent has explored each video stream environment for a certain number of episodes (e.g., ten times), we update its policy using imitation learning. The agent utilizes the historical Pareto set $\hat{\mathcal{P}}^c$ to adjust its action selection, effectively incorporating expert knowledge into its decision-making process.

5.2 High-Throughput Ingestion

5.2.1 Video Clustering. Clustering is a useful approach to reduce configuration exploration cost when facing large volumes of video streams. Video streams are clustered into multiple groups according to their similarity in specific aspects. Each group shares the same configuration and avoids the cost of assigning a unique configuration for each individual video stream. For instance, Chameleon [13] and Skyscraper [19] sample configurations and adopt the accuracy results as the key indicator to cluster video streams, i.e., videos with similar performance under the same configuration are clustered as a group. In our scenario, focusing on urban-scale surveillance video processing, clustering can effectively leverage the similarities across different surveillance video scenes to identify better configurations suited to these specific scenarios. In implementation, multiple configurations can be sampled to generate a vector of accuracy results for each video stream. Subsequently, the k-means clustering algorithm is applied on the accuracy vectors to generate the final clusters of video streams. However, the construction of the accuracy vector requires applying a golden model to estimate the accuracy. The golden model comprehensively extracts all contextual information from the video, obtaining target data (including detection and tracking results) to generate the accuracy vector without lowering the video frame rate or resolution. The process can be expensive when there are a large number of video streams and the vector dimension is high.

To address these challenges, we utilize lightweight proxy models mentioned in Section 5.1.2, which are more efficient than the costly golden model for generating accuracy metrics. Specifically, in clustering process, we apply the proxy models to extract nine types of key features, generating accuracy results for each video stream, which are then stored as vectors, normalized, and clustered using the k-means algorithm based on vector distance metrics. The feature extraction strategy prioritizes quickly obtaining comprehensive video stream information without relying on expensive object detection algorithms. We achieve this by employing proxy models (detailed in Section 5.1.2) for rapid object-level feature identification. These models balance speed and accuracy, allowing us to efficiently characterize video streams. By clustering streams with similar feature vectors, we enable streams within the same cluster to share a common video ingestion process under a collectively optimized configuration, thereby improving overall processing efficiency.

5.2.2 Global Optimization Module. After dividing the multiple video stream feeds into several clusters $C_k (k = 1, 2, \dots, K)$, the system identifies a representative video stream for each cluster and searches for a Pareto configuration set $\mathscr{P}^{C_k}$ for each cluster. Now, we need to consider the problem from the global perspective because the concurrent ingestion of video streams is resource constrained. As the number of video streams increases, Hippo adjusts the ingestion plan to prevent out-of-memory errors. It integrates resource-awareness into video ingestion workflows by estimating GPU memory usage for configurations in the Pareto set. Due to the large CCS configuration space, we focus on configurations that significantly impact GPU memory. This estimation maps configurations to the maximum GPU memory usage observed during prior training for representative configurations.

The global optimization problem can be formulated as a linear programming problem, with the objective of maximizing the overall ingestion accuracy, under the constraints of real-time processing and GPU memory capacity. As mentioned in Definition 3.6, we use efficiency as the real-time constraint and the goal is to maximize accuracy for a more practical setting, by first ensuring that all video streams can be processed in real-time. Conversely, if the two aspects are reversed, establishing an appropriate video ingestion accuracy for all streams could become challenging. However, the techniques are still applicable to this reversed scenario, as optimization algorithm is designed to construct the Pareto curve in terms of both accuracy and efficiency. Formally, given the efficiency $fps(\mathrm{p}_i^{C_k})$ of selecting the configuration $\mathrm{p}_i^{C_k}$ from the Pareto set $\mathscr{P}^{C_k}$ for the video stream cluster C_k, we can estimate the time required to process a unit time δt video. The requirement for real-time processing refers to the condition where the processing time per video unit is less than the unit time.

$$\sum_{k=1}^{K} |C_k| \cdot \delta t / fps(\mathrm{p}_i^{C_k}) < \delta t, \quad \mathrm{p}_i^{C_k} \in \mathscr{P}^{C_k}$$

Meanwhile, the GPU memory resource $res(\mathrm{p}_i^{C_k})$ for each configuration can be pre-calculated, allowing us to also determine the resource utilization for different configurations, and the total resource usage is less than the predefined GPU memory resources R_M.

$$\sum_{k=1}^{K} res(\mathrm{p}_i^{C_k}) < R_M, \quad \mathrm{p}_i^{C_k} \in \mathscr{P}^{C_k}$$

Considering the two constraints, the optimization object is to maximize estimated accuracy $acc(\mathrm{p}_i^{C_k})$.

$$\max_{\mathrm{p}_i} \sum_{k=1}^{K} |C_k| \cdot acc(\mathrm{p}_i^{C_k}) / \sum_{k=1}^{K} |C_k|$$

We utilize an off-the-shelf integer linear programming library to solve this optimization problem and obtain the optimal configuration for each cluster of video streams.

6 EVALUATION

6.1 Experimental Setup

Datasets. To construct a simulated environment with large-scale video streams, we employ a highly diverse benchmark consisting of 8 video datasets: 7 from prior works and 1 new dataset, focusing on urban-scale surveillance video processing. This benchmark is beneficial for many downstream applications, including smart transportation and urban safety. We use the Tokyo, UAV, and Warsaw datasets from MIRISs [3], the Amsterdam and Jackson datasets from BlazeIt [14], and Caldot1 and Caldot2 from OTIF [4]. We also evaluated Hippo on the Seattle dataset from online video surveillance websites in Seattle[1]. The UAV dataset consists of video captured from an aerial drone, while the other 6 datasets consist of video from fixed street-level cameras. Amsterdam captures activity at a riverside plaza, Caldot1 and Caldot2 capture highway activity, while the other 5 datasets capture city traffic junctions. In total, we acquired 1,200 video clips, each with a duration of one minute. The data was then split into training and test datasets with a ratio of 2 : 1.

Queries. We examine three types of common video queries, including selection queries, aggregation queries and event-based queries. For each type of query, we craft 3 instances, resulting in a total of 9 queries. Details of these queries are presented in Table **??**.

Performance Metrics. To evaluate video ingestion quality, we adopt three popular metrics to measure the accuracy of object tracking, including MOTA, IDF1, and HOTA [25], which are evaluated using the same RL agent. Generally speaking, MOTA leans towards evaluating object detection quality, IDF1 highlights accurate association, and HOTA is a recent metric that balances detection, association, and localization.

For selection queries and event-based queries, we use F_1-score to measure the accuracy of target video frame retrieval. We measure the quality of aggregation query processing using $Acc = 1 - MAPE = 1 - |\frac{y-\hat{y}}{y}|$, where y is the ground-truth value and $\hat{y}$ is the prediction value. For all these metrics, higher values indicate better performance.

As to efficiency, we report the total processing time to ingest the hundred-scale video streams. In addition, we measure the portion of video streams that can be processed in real time, i.e., their processing speed aligns with the incoming rate of video frames.

Comparison Approaches. We choose OTIF [4] and Skyscraper [19], both from MIT database group, as two competitive baselines for efficient video ingestion. These two works share a similar idea of utilizing configuration space for parameter optimization and present heuristic search algorithms to find (sub-)optimal settings. Additionally, we also compare our approach with UniTune [36], which was originally developed for autonomous tuning in relational database. We modify the optimization

[1]https://web.seattle.gov/Travelers

Table 4. Performance of video ingestion with the setting of 100 video streams on GeForce RTX 4090.

Method	HOTA	MOTA	IDF1	Total Time
OTIF	0.56	0.45	0.63	33.95s
Skyscraper	0.53	0.45	0.63	34.66s
UniTune	0.55	0.37	0.58	38.21s
Hippo	**0.65**	**0.59**	**0.72**	**23.69*s***

objectives of UniTune to take into account the accuracy and efficiency of video data processing simultaneously.

Implementation Details. To train the configuration explorer in Hippo, we set the batch size to 32, i.e., each batch contains 32 video clips. For each batch, we generate the Pareto configuration set and calculate the losses for reinforcement and imitation learning. The fully connected layer and LSTM network are configured with 128 dimensions in the hidden layer. The model is trained for 100 epochs, with a learning rate of 0.01. All experiments are conducted on a server equipped with an Intel(R) Core(TM) i9-10980XE CPU (3.00GHz) and one NVIDIA GeForce RTX 4090 GPU, which offers up to 82.6 TFLOPS of FP32 performance, 1,008 GB/s memory bandwidth, and 16,384 CUDA cores. Additionally, we evaluate the performance on a server with the same CPU and one NVIDIA GeForce RTX 3090Ti GPU, which provides up to 40.0 TFLOPS of FP32 performance, 1,008 GB/s memory bandwidth, and 10,496 CUDA cores, to verify its performance under resource-constrained conditions.

6.2 Performance of Video Ingestion

In this experiment, we examine the performance of real-time video ingestion against a large number of video streams. Each video stream is simulated by a 1-minute video clip whose FPS is 30. Hence, each stream consists of 1800 video frames. As to performance metrics, we adopt the ingestion speed (measured by the total time cost to process all video frames) and the quality of video ingestion using three popular metrics (HOTA, MOTA and IDF1). We present the ingestion quality and speed for 100 video streams in Table 4. Among the baselines, OTIF and Skyscraper exhibit comparable performance in both speed and quality, as they adopt similar search strategies to explore the configuration space. However, OTIF operates over a larger search space, increasing the likelihood of finding better configurations. As a result, it achieves slightly better HOTA scores and marginally higher speed than Skyscraper. UniTune exhibits inferior performance, as it was originally designed for autonomous tuning in RDBMS with a focus on efficiency. Despite the customization with two agents to balance accuracy and efficiency in video ingestion, the dual-objective task remains challenging for UniTune. Moreover, its optimization strategy appears less effective in exploring the vast configuration space in our scenario. In contrast, Hippo surpasses all other methods across all performance metrics. Its comprehensive configuration space and efficient exploration strategy allow it to dominate. Compared with OTIF, Hippo improves its HOTA by over 20% and MOTA by 30%, with significantly lower amount of processing time (23.69*s* v.s. 33.95s). Additionally, we evaluate the same performance metrics in resource-constrained environments in Table 5. Hippo shows greater advantages in resource-constrained environments, achieving shorter processing time (24.78*s* v.s. 35.85s) and better performance. The results show that with inferior computing resources, Hippo demonstrates even greater performance superiority than the baselines.

Table 5. Performance of video ingestion with the setting of 100 video streams on GeForce RTX 3090Ti.

Method	HOTA	MOTA	IDF1	Total Time
OTIF	0.56	0.43	0.55	35.85s
Skyscraper	0.49	0.43	0.59	37.72s
UniTune	0.51	0.37	0.58	45.34s
Hippo	**0.64**	**0.59**	**0.71**	**24.78*s***

6.3 Performance of Query Processing Accuracy

We use the extracted trajectories of moving objects to support selection queries (Q_1–Q_3), aggregation queries (Q_4-Q_6) and event-based queries (Q_7-Q_9). The ground truth of traditional queries (Q_1-Q_6) is directly taken from the dataset provided in previous works, and those of event-based queries (Q_7-Q_9) are obtained using expensive multi-object tracking (MOT) models, such as YOLO and ByteTrack, which are then manually verified to ensure accuracy. These queries can be answered instantly on the structured semantic information, with sub-second latency even in a very large video database [4]. Thus, we only report the accuracy of 100 video streams query processing in Table 6 along with resource-constrained environments in Table 7. Generally speaking, higher ingestion quality can lead to better query processing accuracy. We can see that Hippo outperforms the other baselines in all query scenarios. The accuracy of event-based queries (i.e., $Q_7 - Q_9$) is relatively low because we need to predict the turning direction from the extracted trajectories. This step is simply performed with a heuristic rule via the trajectory angle, which is not sufficiently robust. The prediction error grows when the size of object is large, such as trucks and buses.

Table 6. Accuracy of video query processing with the setting of 100 video streams on GeForce RTX 4090.

Method	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
OTIF	0.88	0.19	0.42	0.69	0.85	0.55	0.41	0.07	0.09
Skyscraper	0.81	0.28	0.48	0.50	0.83	0.56	0.41	0.09	0.10
UniTune	0.88	0.24	0.59	0.73	0.89	0.81	0.40	0.07	0.12
Hippo	**0.92**	**0.30**	**0.79**	**0.79**	**0.90**	**0.85**	**0.41**	**0.25**	**0.24**

Table 7. Accuracy of video query processing with the setting of 100 video streams on GeForce RTX 3090Ti.

Method	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
OTIF	0.86	0.19	0.42	0.68	0.81	0.56	0.38	0.06	0.09
Skyscraper	0.80	0.25	0.46	0.47	0.81	0.49	0.41	0.09	0.10
UniTune	0.85	0.19	0.53	0.63	0.83	0.76	0.39	0.07	0.11
Hippo	**0.90**	**0.31**	**0.71**	**0.75**	**0.83**	**0.87**	**0.40**	**0.22**	**0.19**

6.4 Scalability Analysis

In the scalability analysis, we evaluate system performance as the number of video streams increases while utilizing consistent hardware, specifically a single NVIDIA GeForce RTX 4090 GPU. We aim to investigate whether Hippo can dynamically adjust the video ingestion configuration according to the varying volumes of video streams, thus matching the pace of incoming video frame data. We report the ratio of video streams without processing delay in Table 8, the ingestion quality measured by the HOTA of extracted object trajectories in Table 9 and the video streams throughput of ingestion per 1-minute in Table 10.

Table 8. Ratio of video streams without processing delay.

Method	100	150	200	250	300
OTIF	**100.0%**	**100.0%**	**100.0%**	**100.0%**	62.3%
Skyscraper	**100.0%**	**100.0%**	**100.0%**	**100.0%**	31.7%
UniTune	**100.0%**	**100.0%**	68.0%	30.0%	31.7%
Hippo	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**

Table 9. Quality of ingestion w.r.t. increasing video streams.

Method	100	150	200	250	300
OTIF	0.56	0.56	0.56	0.54	0.45
Skyscraper	0.53	0.54	0.49	0.40	0.40
UniTune	0.55	0.52	0.48	0.48	0.43
Hippo	**0.65**	**0.62**	**0.61**	**0.61**	**0.60**

The findings reveal that UniTune lacks flexibility in handling varying number of video streams. Its inability to optimally balance accuracy and efficiency often results in sub-optimal configurations for certain streams, leading to processing delay. OTIF and Skyscraper also fail to guarantee effective configurations when handling up to 300 video streams. In contrast, Hippo can trade accuracy for efficiency when facing increasing number of video streams. It slightly compromises ingestion quality to ensure real-time processing. This strategy effectively eliminates delays, even when managing up to 300 video streams, and secures an ingestion accuracy that exceeds its competitors by more than 30% (0.60 vs. 0.45). Its comprehensive space offers substantial potential for performance enhancements, and its exploration strategy effectively identifies promising configurations for individual video streams. Moreover, the effectiveness of clustering plays a crucial role in maintaining performance as the number of video streams increases. By dynamically adjusting to the unique characteristics of each video stream scenario, clustering effectively leverages the similarities across different scenarios, contributing to the maintenance of performance stability as the data scales from 100 to 300 video streams. Additionally, applying the previous method (OTIF [4]) at such large scales is prohibitively expensive. For instance, processing one hour of traffic data from 300 cameras using OTIF would cost over \$5 on AWS[2].With better ingestion, the actual cost could be reduced by 40% (\$5 vs. \$3).

Due to the distribution shifts based on time of the day and weather condition, We also consider longer videos from the Seattle dataset, which include two-hour-long footage capturing the continuous transition of scenes from day to night, which provide significant variation in lighting conditions and density of moving objects, and introduce a parameter W to control the interval for re-clustering. In other words, every W minutes, we extract new contextual features and reassign a

[2]E.g., using 50 p3.2xlarge instances, each of which currently costs 3.06 USD/h.

new configuration. The results shown in Table 11 indicate that, compared to the original setting without re-clustering, re-clustering every 30 minutes improves HOTA by over 10% and MOTA by 14%, alleviating the issue of distribution shift.

Table 10. Throughput of ingestion w.r.t. increasing video streams.

Method	100	150	200	250	300
OTIF	100	150	200	250	187
Skyscraper	100	150	200	250	95
UniTune	100	150	136	75	95
Hippo	**100**	**150**	**200**	**250**	**300**

Table 11. Performance of video ingestion with the setting of W minutes in video streams.

W	HOTA	MOTA	IDF1
120 (Origin setting)	0.56	0.49	0.64
30	**0.61**	**0.56**	**0.69**

Table 12 presents the ILP processing times for different video streams. The results show that as the number of streams increases, the ILP approach incurs minimal overhead (0.47s) and remains stable with increasing number of video streams.

Table 12. Time cost of integer linear programming optimization.

Method	100	150	200	250	300
ILP_Efficiency	0.47s	0.47s	0.46s	0.46s	0.45s

6.5 Effect of Search Space Exploration

In the experiment, we perform an in-depth analysis on the explored configuration space, with the goal of providing explainable clues to justify the superiority of Hippo over its competitors. The experiment is conducted on a 5-minute clip. In Figure 5, we plot the Pareto frontiers between ingestion quality and efficiency for Skyscraper, OTIF, and Hippo. Two main observations arise from the experimental results. First, when aiming for high ingestion accuracy, the efficiency of all three algorithms converges in the upper-right region of the figure, as they favor lower sampling rates and higher frame resolutions—strategies that preserve more original data but demand more processing time. Second, when accuracy degradation is acceptable, Hippo delivers remarkably greater efficiency than Skyscraper and OTIF. At an accuracy level of 0.5, Hippo is more than three times as efficient. This is because Hippo is associated with many more configurable knobs to control the trade-off between ingestion quality and efficiency. For example, the configurations P_1 and P_2 cause high efficiency, because their sampling rate is set to 256, i.e., Hippo processes only 1/256 of the total frames.

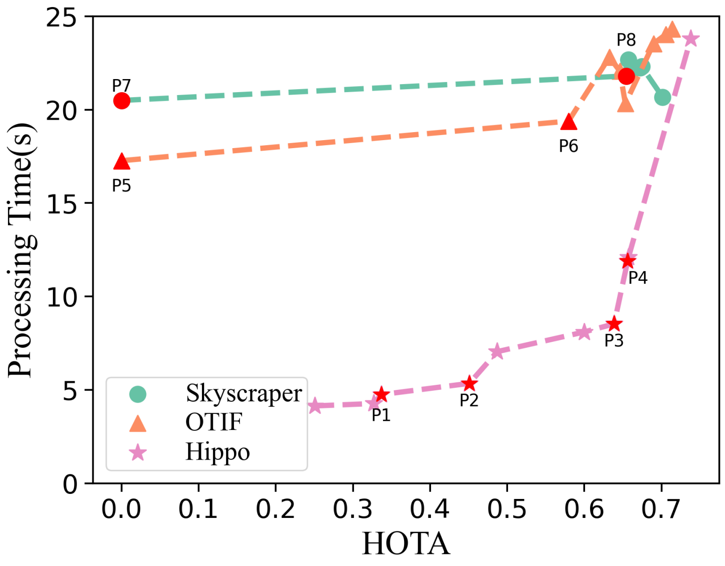

Fig. 5. Pareto frontiers discovered by different algorithms.

We also explored how parameter variations across different configurations affect the trade-off between ingestion quality and efficiency. A detailed parameter-level analysis reveals the key factors underlying these changes, as shown in Figure 5. In Hippo's Pareto curve, accuracy improves from P_1 to P_2 without a significant increase in processing time. This is primarily due to an increase in image resolution, enabling more precise object detection. Then, with a proper detection threshold to remove candidate bounding boxes with low confidence, the object tracking accuracy is further enhanced. From P_3 to P_4, ingestion accuracy continues to improve, driven by the use of a more computationally intensive detection model and adjustments to two tracking-related thresholds. In the Pareto frontiers for OTIF, accuracy rises significantly from P_5 to P_6 due to the increase in resolution and the decrease in video frame sampling rate. As a result, more detailed semantic information of the video is obtained during processing. The increase in detection threshold also leads to more accurate object detection results. As to the performance of Skyscraper from P_7 to P_8, reducing the sample rate improves tracking accuracy because more detailed information can be captured. While the track max loss time also contributes to increasing tracking accuracy. The smaller parameter space used in previous method (i.e.,OTIF [4]) indeed has a positive impact on the results, particularly because it focuses on common parameters for video ingestion, and tuning within this smaller space can still yield effective performance. The key parameters and their variations that contributed to these accuracy gains can be observed from Table 13.

6.6 Efficiency on Search Space Exploration

Besides inference time, we assess the efficiency of configuration space exploration, defined as the total time required to assign a configuration for each video stream. It is important to note that this exploration occurs offline, and the associated cost is incurred only once. As shown in Figure 6, Hippo consumes much higher search space exploration time because it involves additional offline training cost. Training the reinforcement learning agents takes 26 hours, but the exploration phase itself is efficient, taking less than 1.5 hours. In contrast, OTIF and Skyscraper use a hill-climbing algorithm for exploration. While they avoid training time, their search stages are more time-consuming,

Table 13. Key configurations parameters and their variation for performance trade-off in Figure 5.

	Key Parameters	Variation
$P_1 \rightarrow P_2$ (Hippo)	image_resolution	[128,128]→[256,256]
	detect_threshold	0.2→0.4
$P_3 \rightarrow P_4$ (Hippo)	detect_model_size	large→medium
	track_stop_threshold	0.4→0.3
	track_switch_threshold	0.7→0.6
$P_5 \rightarrow P_6$ (OTIF)	sample_rate	64→32
	image_resolution	[128,128]→[256,256]
	detect_threshold	0.2→0.4
$P_7 \rightarrow P_8$ (Skyscraper)	sample_rate	64→32
	track_max_loss_time	32→64

requiring approximately 3.5 and 10 hours, respectively. To further investigate the training efficiency of Hippo in a smaller configuration space, we construct a new baseline Hippo$_s$, which integrates OTIF's configuration space and Hippo's search exploration strategy. We can see that the total time for training and exploration becomes comparable to OTIF and Skyscraper. The accuracy of Hippo$_s$ will be discussed in the subsequent ablation study to reflect the effect of our proposed comprehensive configuration space.

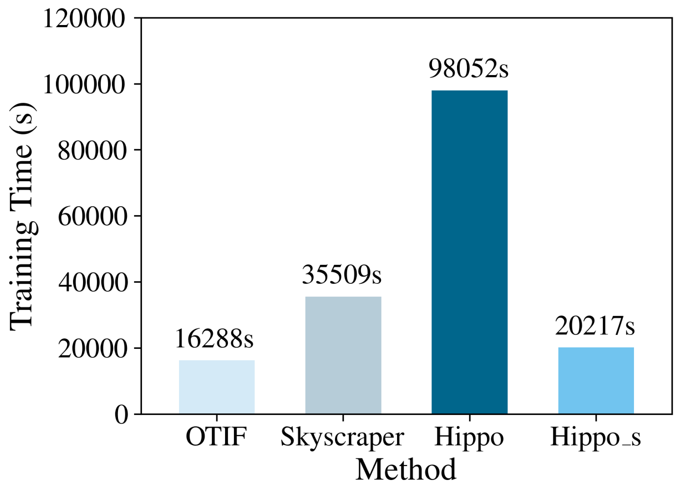

Fig. 6. Offline configuration space exploration time.

6.7 Effect of Clustering

The performance bottleneck in clustering lies in extracting context vector from video streams.Skyscraper relies on an expensive multi-object tracking model to generate accuracy vectors under different configuration settings. Hippo adopts a more efficient approach by employing lightweight models to extract more diversified contextual features. OTIF and UniTune are not included in the comparison, as they utilize the same clustering mechanism as Skyscraper in our implementation. As shown in Figure 7, the time required for feature generation increases with the number of video streams. When there are 300 video, Skyscraper consumes over an hour. In contrast, Hippo uses lightweight proxy models to extract visual context, improving the efficiency of feature generation for clustering by a factor of around 150.

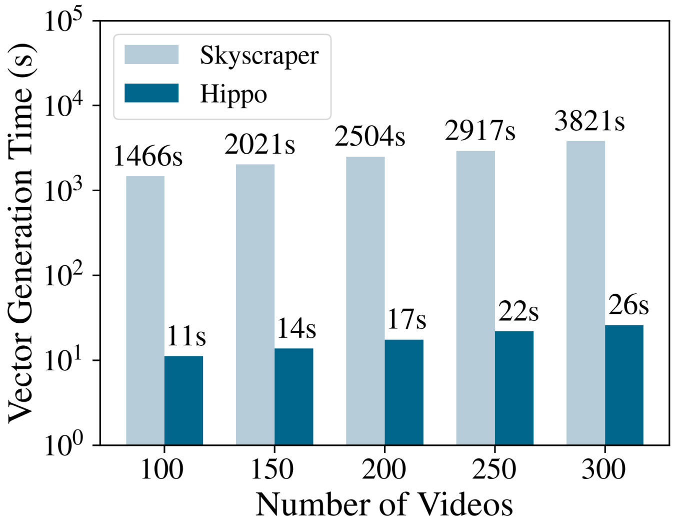

Fig. 7. Time cost to generate features for clustering.

We also evaluate the effectiveness of the generated feature vectors for clustering. When the contextual features of two video streams are similar, we expect their assigned configurations to yield comparable performance, reflected in their proximity on the Pareto curve. To quantify this, we select an anchor feature f_a from video stream v_a and randomly sample two other features, f_1 from v_1 and f_2 from v_2. We then compute the pairwise distances in feature space, $D_f(f_a, f_1)$ and $D_f(f_a, f_2)$. Without loss of generality, we assume $D_f(f_a, f_1) < D_f(f_a, f_2)$. The clustering procedure essentially relies on the proximity between features. A positive match occurs if the normalized distance D_p in the performance space, including the two dimensions of ingestion efficiency and accuracy, satisfies $D_p(v_a, v_1) < D_p(v_a, v_2)$. As shown in Figure 8, even though Hippo uses proxy models to extract contextual features, its quality slightly surpasses that of Skyscraper. This advantage arises from Hippo's use of more diverse visual features, providing stronger cues for clustering.

6.8 Ablation Study

In the ablation study, we evaluate the impact of Hippo's comprehensive configuration space and search exploration strategy. We create a baseline model, Hippo$_s$, by reducing Hippo's configuration space to match that of OTIF and applying our search strategy to this smaller space. As shown in Table 14, Hippo$_s$ performs worse than the original version, verifying that the larger configuration space offers greater potential for performance optimization through a wider variety of parameter combinations, providing more opportunities to discover parameters and certain parameter combinations that positively benefit the current video stream. However, Hippo$_s$ still outperforms OTIF, demonstrating the superior effectiveness of our search strategy compared to OTIF's hill-climbing

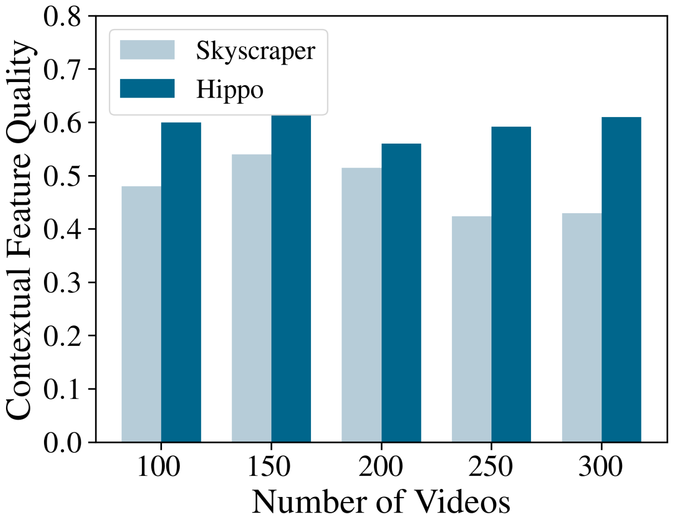

Fig. 8. Effectiveness of feature generation.

approach. Furthermore, when we removed imitation learning (IL) from our DRL agent, we observed significant performance degradation. This is because, after removing IL, we can no longer use previously accumulated experience configurations. Instead, we have to search for trade-off configurations from scratch. Given the vast search space, this easily leads to getting trapped in local optima.

Table 14. Performance of video ingestion with the setting of 100 video streams.

Method	HOTA	MOTA	IDF1	Total Time
Hippo	**0.65**	**0.59**	**0.72**	**23.69***s*
Hippo$_s$	0.58	0.51	0.66	33.94s
OTIF	0.56	0.43	0.62	33.95s
Hippo w/o IL	0.43	0.30	0.47	38.87s

7 Conclusion

In this paper, we presented Hippo as a high-throughput video ingestion system that supports real-time ingestion of hundreds of video streams with high quality. To achieve the goal, we proposed a comprehensive configuration space that takes into account the entire ingestion pipeline's configurable components and devised an effective RL-based exploration strategy. Experimental results showed that Hippo outperformed prior approaches in terms of both efficiency and accuracy.

Acknowledgments

The work is supported by the National Key Research and Development Project of China (2022YFF0902000), the Fundamental Research Funds for the Central Universities (226-2024-00145226-2024-00216), the Major Research Program of Zhejiang Provincial Natural Science Foundation (No. LD24F020015) and NSFC Grant No. 62402420

References

[1] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. 2017. Automatic Database Management System Tuning Through Large-scale Machine Learning. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, Semih Salihoglu, Wenchao Zhou, Rada Chirkova, Jun Yang, and Dan Suciu (Eds.). ACM, 1009–1024. doi:10.1145/3035918.3064029

[2] Michael R. Anderson, Michael J. Cafarella, Germán Ros, and Thomas F. Wenisch. 2019. Physical Representation-Based Predicate Optimization for a Visual Analytics Database. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 1466–1477. doi:10.1109/ICDE.2019.00132

[3] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael J. Cafarella, Tim Kraska, and Sam Madden. 2020. MIRIS: Fast Object Track Queries in Video. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1907–1921. doi:10.1145/3318464.3389692

[4] Favyen Bastani and Samuel Madden. 2022. OTIF: Efficient Tracker Pre-processing over Large Video Datasets. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 2091–2104. doi:10.1145/3514221.3517835

[5] Favyen Bastani and Samuel Madden. 2022. OTIF: Efficient Tracker Pre-processing over Large Video Datasets. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 2091–2104. doi:10.1145/3514221.3517835

[6] Jiashen Cao, Karan Sarkar, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. 2022. FiGO: Fine-Grained Query Optimization in Video Analytics. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 559–572. doi:10.1145/3514221.3517857

[7] Lu Chen, Yunjun Gao, Ziquan Fang, Xiaoye Miao, Christian S Jensen, and Chenjuan Guo. 2019. Real-time distributed co-movement pattern detection on streaming trajectories. *Proceedings of the VLDB Endowment* 12, 10 (2019), 1208–1220.

[8] Pranjal Gupta, Amine Mhedhbi, and Semih Salihoglu. 2021. Columnar storage and list-based processing for graph database management systems. *arXiv preprint arXiv:2103.02284* (2021).

[9] John Michael Hammersley and David Christopher Handscomb. 1964. General principles of the Monte Carlo method. In *Monte Carlo Methods*. Springer, 50–75.

[10] 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.

[11] Bo Hu, Peizhen Guo, and Wenjun Hu. 2022. Video-zilla: An Indexing Layer for Large-Scale Video Analytics. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1905–1919. doi:10.1145/3514221.3517840

[12] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodík, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*, Sergey Gorinsky and János Tapolcai (Eds.). ACM, 253–266. doi:10.1145/3230543.3230574

[13] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodík, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*, Sergey Gorinsky and János Tapolcai (Eds.). ACM, 253–266. doi:10.1145/3230543.3230574

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

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

[16] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, Yi Sun, and Matei Zaharia. 2021. Accelerating Approximate Aggregation Queries with Expensive Predicates. *Proc. VLDB Endow.* 14, 11 (2021), 2341–2354. doi:10.14778/3476249.3476285

[17] Daniel Kang, Francisco Romero, Peter D. Bailis, Christos Kozyrakis, and Matei Zaharia. 2022. VIVA: An End-to-End System for Interactive Video Analytics. In *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*. www.cidrdb.org. https://www.cidrdb.org/cidr2022/papers/p75-kang.pdf

[18] Ferdinand Kossmann, Ziniu Wu, Eugenie Lai, Nesime Tatbul, Lei Cao, Tim Kraska, and Sam Madden. 2023. Extract-Transform-Load for Video Streams. *Proc. VLDB Endow.* 16, 9 (2023), 2302–2315.

[19] Ferdinand Kossmann, Ziniu Wu, Eugenie Lai, Nesime Tatbul, Lei Cao, Tim Kraska, and Sam Madden. 2023. Extract-Transform-Load for Video Streams. *Proc. VLDB Endow.* 16, 9 (2023), 2302–2315. https://www.vldb.org/pvldb/vol16/p2302-kossmann.pdf

[20] Ziliang Lai, Chenxia Han, Chris Liu, Pengfei Zhang, Eric Lo, and Ben Kao. 2021. Top-K Deep Video Analytics: A Probabilistic Approach. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1037–1050. doi:10.1145/3448016.3452786

[21] Guoliang Li, Xuanhe Zhou, Shifu Li, and Bo Gao. 2019. QTune: A Query-Aware Database Tuning System with Deep Reinforcement Learning. *Proc. VLDB Endow.* 12, 12 (2019), 2118–2130. doi:10.14778/3352063.3352129

[22] Zepeng Li, Dongxiang Zhang, Sai Wu, Mingli Song, and Gang Chen. 2024. Sampling-resilient multi-object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 3297–3305.

[23] Mona Lisa and Hew Bot. 2017. *My Research Software.* doi:10.5281/zenodo.1234

[24] Yuqing Liu, Lizhen Wang, Peizhong Yang, and Lihua Zhou. 2024. Mining Interpretable Regional Co-location Patterns Based on Urban Functional Region Division. *Data Science and Engineering* (2024), 1–22.

[25] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip H. S. Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. 2021. HOTA: A Higher Order Metric for Evaluating Multi-object Tracking. *Int. J. Comput. Vis.* 129, 2 (2021), 548–578. doi:10.1007/s11263-020-01375-2

[26] Limin Ma and Ken Q Pu. 2024. Accelerating Relational Keyword Queries With Embedded Predictive Neural Networks. In *2024 IEEE International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 84–89.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[28] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. *CoRR* abs/1804.02767 (2018). arXiv:1804.02767 http://arxiv.org/abs/1804.02767

[29] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 91–99. https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

[30] Francisco Romero, Mark Zhao, Neeraja J. Yadwadkar, and Christos Kozyrakis. 2021. Llama: A Heterogeneous & Serverless Framework for Auto-Tuning Video Analytics Pipelines. In *SoCC '21: ACM Symposium on Cloud Computing, Seattle, WA, USA, November 1 - 4, 2021*, Carlo Curino, Georgia Koutrika, and Ravi Netravali (Eds.). ACM, 1–17. doi:10.1145/3472883.3486972

[31] Stuart J Russell and Peter Norvig. 2010. *Artificial intelligence a modern approach.* London.

[32] Ziyang Xiao and Dongxiang Zhang. 2023. A deep reinforcement learning agent for geometry online tutoring. *Knowledge and Information Systems* 65, 4 (2023), 1611–1625.

[33] Yanchao Xu, Dongxiang Zhang, Shuhao Zhang, Sai Wu, Zexu Feng, and Gang Chen. 2024. Predictive and near-optimal sampling for view materialization in video databases. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–27.

[34] Ji Zhang, Yu Liu, Ke Zhou, Guoliang Li, Zhili Xiao, Bin Cheng, Jiashu Xing, Yangtao Wang, Tianheng Cheng, Li Liu, Minwei Ran, and Zekang Li. 2019. An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 415–432. doi:10.1145/3299869.3300085

[35] Ji Zhang, Yu Liu, Ke Zhou, Guoliang Li, Zhili Xiao, Bin Cheng, Jiashu Xing, Yangtao Wang, Tianheng Cheng, Li Liu, Minwei Ran, and Zekang Li. 2019. An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 415–432. doi:10.1145/3299869.3300085

[36] Xinyi Zhang, Zhuo Chang, Hong Wu, Yang Li, Jia Chen, Jian Tan, Feifei Li, and Bin Cui. 2023. A Unified and Efficient Coordinating Framework for Autonomous DBMS Tuning. *Proc. ACM Manag. Data* 1, 2 (2023), 186:1–186:26. doi:10.1145/3589331

[37] Xinyi Zhang, Zhuo Chang, Hong Wu, Yang Li, Jia Chen, Jian Tan, Feifei Li, and Bin Cui. 2023. A Unified and Efficient Coordinating Framework for Autonomous DBMS Tuning. *Proc. ACM Manag. Data* 1, 2 (2023), 186:1–186:26. doi:10.1145/3589331

[38] Xinyi Zhang, Hong Wu, Zhuo Chang, Shuowei Jin, Jian Tan, Feifei Li, Tieying Zhang, and Bin Cui. 2021. ResTune: Resource Oriented Tuning Boosted by Meta-Learning for Cloud Databases. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 2102–2114. doi:10.1145/3448016.3457291

[39] Xinyi Zhang, Hong Wu, Yang Li, Jian Tan, Feifei Li, and Bin Cui. 2022. Towards Dynamic and Safe Configuration Tuning for Cloud Databases. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 631–645. doi:10.1145/3514221.3526176

[40] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2022. ByteTrack: Multi-object Tracking by Associating Every Detection Box. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXII (Lecture Notes in Computer Science, Vol. 13682)*, Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 1–21. doi:10.1007/978-3-031-20047-2_1

[41] Silin Zhou, Chengrui Huang, Yuntao Wen, and Lisi Chen. 2025. Feature Enhanced Spatial–Temporal Trajectory Similarity Computation. *Data Science and Engineering* 10, 1 (2025), 1–11.

[42] Lei Zou, Lei Chen, M Tamer Özsu, and Dongyan Zhao. 2012. Answering pattern match queries in large graph databases via graph embedding. *The VLDB journal* 21 (2012), 97–120.

Received October 2024; revised January 2025; accepted February 2025