Journal of Computer Science and Electrical Engineering
ISSN: 2663-1946
DOI: https://doi.org/10.61784/jcsee3059

PERFORMANCE-SCORING DRIVEN MODEL SCALING AND SCHEDULING FOR EDGE VIDEO ANALYTICS SERVING

XingTao Xu[1], JiZhe Zhang[2], HaiTao Zhang[1*]
[1]*School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China.*
[2]*Kembl Petroleum Technology Co., Ltd., Beijing 100029, China.*
Corresponding Author: Haitao Zhang, Email: zht@bupt.edu.cn

Abstract: The growing demand for real-time, multi-task video analytics at the edge has encountered challenges in resource-constrained environments, including redundant computations across tasks and poor adaptability to dynamic workloads. In this paper, we propose a performance scoring-driven framework for model scaling and scheduling in edge video analytics. The framework consists of two core modules: (1) the model performance scoring module evaluates the model performance from four dimensions—video complexity, task correlation, model performance, and system resource utilization. (2) The model scaling and scheduling module then calculates a comprehensive score based on these four evaluation metrics. Aiming at maximizing the comprehensive score, this module employs the particle swarm optimization algorithm for model scheduling and system resource allocation, and selects the optimal combination of detection models based on the current model and system states. Experimental results demonstrate that our framework outperforms state-of-the-art baselines, achieving superior performance under dynamic edge workloads.
Keywords: Model scaling and scheduling; Edge computing; Scoring metrics

1 INTRODUCTION

In recent years, the advancement of video surveillance technology and the increasing number of monitoring cameras in public spaces have made video analytics a prominent research field [1-2]. In practical applications, to reduce transmission latency and enhance data security, video analytics models are increasingly being deployed on edge devices rather than cloud servers [3-4]. However, it brings challenges such as constrained computational and storage resources and unstable network conditions, thus leading to degraded accuracy or compromised real-time performance in edge-based video analytics. Existing works primarily optimize accuracy-latency trade-off or resource allocation in edge video analytics through model scaling and scheduling, aiming to improve system quality of service (QoS). Nevertheless, these methods underperform in the multi-task video analytics scenarios due to the following reasons:

1.1 Task Dependencies Induce Redundant Computations

In such edge scenarios, multiple video analytics tasks exhibit dependencies or correlations [5]. For instance, bounding box generated from object detection may subsequently be used for recognition or object tracking. Current methods overlook these correlations, causing repeated computations across associated tasks. As a result, this redundancy wastes edge computational and storage resources, thereby reducing accuracy, increasing latency, and significantly degrading multi-task processing efficiency and system performance.

1.2 Simplified Scheduling Objectives Lack Overall Model Evaluation

Existing edge scheduling decisions are typically driven by the goal of maximizing resource utilization or optimizing the accuracy-latency trade-off. These single-objective strategies have inherent limitations. They prioritize single metric (accuracy [8-9], latency [10-11] or resource utilization [6-7]) during model scaling and scheduling, thereby failing to consider other critical metrics and give a comprehensive evaluation of the model performance and system environment. In real scenarios, sacrificing accuracy for high resource utilization or incurring delays in time-sensitive contexts could severely undermine the entire video analytics system. Furthermore, excessive pursuit of resource utilization may destabilize systems under high workloads, degrading task accuracy and failing to meet the demands of complex and dynamic video analytics scenarios.
To address these issues, we propose a performance scoring-driven model scaling and scheduling method tailored for edge video analytics serving. Our method specifically targets at two key challenges in multi-task edge video analytics serving: redundant computations and simplified scheduling objectives. For the incomplete evaluation challenge in existing approaches, we design a model performance scoring mechanism. In addition to resource utilization, the scoring mechanism incorporates input video complexity, task characteristics, and model performance metrics to generate a comprehensive evaluation score for driving model scheduling decisions. We further propose a dynamic model scaling and scheduling algorithm which employs the particle swarm optimization (PSO) algorithm [12, 13] to optimize resource allocation and model selection process.
The main contributions of our work are as follows:
(1) A model performance scoring mechanism tailored to guide scaling and scheduling decisions. To catch the key requirements and task characteristics of edge video analytics serving, we design a holistic scoring metric that integrates

(2) video complexity, task dependencies, model accuracy, and resource utilization. This metric serves as the foundation for collaborative optimization in edge environments.
(3) An elastic model scaling and scheduling method using particle swarm optimization. Using the comprehensive scores as the optimization objective, we develop a dynamic model scaling and scheduling strategy that optimizes resource allocation and model combinations through PSO. This method achieves a balance between real-time responsiveness, computational efficiency, and system stability under varying edge video analytics workloads.
(4) Implementation and validation of an edge video analytics system. We implement and evaluate our proposed method on real-world edge devices and several video analytics scenarios, demonstrating its effectiveness.

2 MOTIVATIONS

This section first introduces the technical background and characteristics of edge video analytics. Based on this, we identify the challenges of existing model scaling and scheduling methods.

2.1 Characteristics of edge video analytics

Unlike cloud-centric processing, edge video analytics serving exhibits the following characteristics:
(1) Limited Resources
Common edge devices operate under limited computational and storage resources, and this hardware disparity compels algorithm designs to balance model accuracy against resource efficiency.
(2) Latency-sensitive requirements
Applications such as autonomous driving and real-time surveillance demand millisecond-level responsiveness.
(3) Multi-task concurrency and dependency
In edge computing scenarios, multiple correlated video analytics tasks are often executed concurrently on the same edge node, sharing intermediate results or competing for limited resources.
(4) Dynamic environmental complexity
Real-time fluctuations exist in video stream quality (e.g., resolution, lighting), scene semantics (e.g., sudden changes in crowd density), and network conditions.
These inherent traits collectively define the core challenges in edge video analytics, driving research efforts toward cross-task collaboration, resource scheduling, and model optimization in dynamic edge environments.

2.2 Challenges on model scaling and scheduling

Model scaling and scheduling are pivotal for balancing computational efficiency and service quality in edge video analytics. Although prior works have improved resource utilization in static scenarios, they fall short in addressing the following challenges in edge video analytics.
Challenge 1: Redundant computations are caused by task correlations. Existing edge schedulers treat video analytics tasks (e.g., object detection, tracking) as isolated processes, ignoring their inherent dependencies. For instance, a fall detection task and a loitering detection task may process the same video frame independently, extracting overlapping features (e.g., human body bounding boxes). Existing approaches fail to reuse intermediate results across tasks, resulting in unsustainable overheads in multi-task deployments. A holistic scheduler must exploit task correlations to eliminate redundant computations while preserving accuracy.
Challenge 2: Single-objective scheduling has limitations. Most scheduling policies [5, 7, 10, 19] prioritize a single optimization metric. However, video analytics serving demands multi-dimensional optimization. As shown in Figure 1, high-accuracy models may violate latency constraints, whereas lightweight models often compromise accuracy in complex video scenarios. This underscores the need for a unified scoring metric that evaluates model performance across model accuracy, latency, and environmental adaptability.

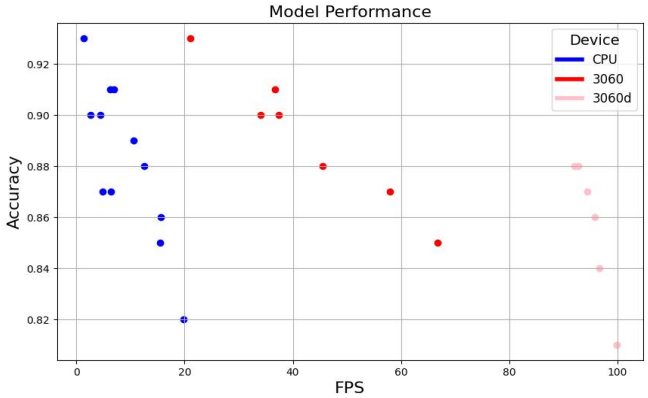


Figure 1 Accuracy-Latency Trade-off on Different Devices

These challenges collectively highlight the limitations of conventional model scaling and scheduling methods in handling task dependencies, multi-dimensional trade-offs, and environmental conditions.

2.3 Problem Formulation

In edge scenarios, quality of service (QoS) is primarily reflected in three aspects: system resource utilization, video analytics accuracy, and processing latency. Given the high real-time requirements in edge scenarios, our objective is to minimize the system latency, while ensuring system overall accuracy and resource utilization. Its formal definition is as follows:

$$min\ L_{TA} \tag{1}$$

$$\text{s.t.}\ Acc_i \geq \tau_i \quad U_{CPU}, U_{GPU} \geq U_{min} \quad M_u < M_t \tag{2}$$

Latency L_{TA} refers to the execution time from the input of a video frame to the output of the result. The resource utilization constraint is as follows: the CPU utilization U_{CPU} and GPU utilization U_{GPU} of system should exceed specified minimum thresholds U_{min}. The memory used M_u must be less than the system memory limit M_t. Additionally, we set different accuracy thresholds τ_i for different video analytic tasks, and the accuracy must be greater than the corresponding threshold.

To solve the problem above, we adopt a model performance scoring mechanism to evaluate the performance of different models under various constraints. Then, we conduct model scaling and scheduling based on these scores. The whole framework will be introduced in the next section.

3 FRAMEWORK OVERVIEW

We propose a dynamic performance scoring-driven model scaling and scheduling method. The overall framework architecture is illustrated in Figure 2.

As inputs, the framework receives (1) raw video frames from edge cameras or unprocessed video streams, (2) characteristics of video analytics tasks, including video analytics objectives (e.g., fall detection, face recognition), priority levels, and correlation requirements between tasks, (3) model performance profiles, which involves several key metrics: accuracy, latency and memory demands of video analytics models, and (4) real-time resource snapshot, including available GPU memory, CPU utilization, and network bandwidth.

Our framework consists of three core modules: the frame processing module, the model performance scoring module, and the model scaling and scheduling module. The frame processing module first extracts and filters key frames from the input video streams for subsequent analytics. Then it calculates the video complexity based on the number of target objects. The model performance scoring module dynamically evaluates each candidate model using the input video analytics task information (e.g., priority, QoS requirements), model information (e.g., model architecture, accuracy and latency profiles), and real-time system resource status (e.g., GPU memory availability). By aggregating multi-dimensional evaluations, this module calculates the comprehensive metric $TScore$. Finally, the model scaling and scheduling module makes scheduling decisions based on $TScore$. Leveraging the Particle Swarm Optimization (PSO) algorithm, this module identifies the optimal model combinations that maximize the $TScore$, thereby ensuring efficient model performance under constraints in Eq.2.

The core of our framework is the model performance scoring mechanism and the model scaling and scheduling method based on the PSO algorithm, which will be detailed in Section 4 and Section 5.

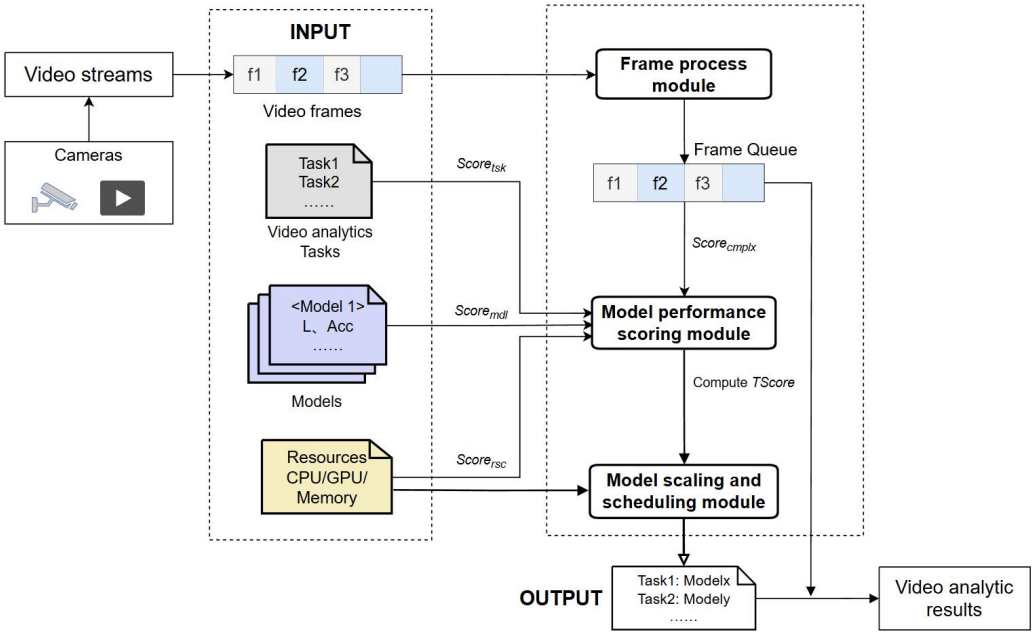


Figure 2 The Overview of Our Framework

4 MODEL PERFORMANCE SCORING

According to the optimization objectives in Section 2.3, in the edge video analytics system, the performance indicators of latency, accuracy, and system resource utilization are influenced by multiple factors. Therefore, in this section, we introduce a model performance scoring metric for edge video analytics, which integrates four critical factors: video complexity, which reflects the number of targets to be detected in the input frames and the degree of changes in the video background, task correlation, which is calculated based on the categories of video analytics tasks in the system and the feature similarity among different tasks, model performance, including the accuracy and latency of different model variants on different edge devices, and resource utilization, referring to the estimated usage of system resources (CPU, GPU, and memory) on the edge device.

4.1 Video Complexity Score

For each task, the quality of the input will indirectly affect the performance of models and thus influence the final analytics result. Frames with higher complexity require models with better performance for video analytics. Taking this impact into account, we propose a scoring metric $Score_{cmplx}$ to measure the complexity of videos, which takes into consideration factors such as the number of targets and the complexity of environment in the video. We calculate the video complexity $Complex_{obj}(vd)$ based on the number of targets in the video frame and the historical maximum number of targets. Difference function $DIFF(f_n)$ is used to calculate the difference between the current frame f_n and the previous frame f_{n-1}.
Based on the video complexity and the difference between frames. The video complexity score can be calculated as:

$$Score_{cmplx}(vd) = DIFF(f_n) \cdot \left(1 - Complex_{obj}(vd)\right) \tag{3}$$

4.2 Task Correlation Score

Usually, there exists a certain correlation in the analytical logic of video analysis tasks. For instance, both target detection and target tracking require the use of target bounding boxes in their respective process. Thus, we use a scoring metric to evaluate the correlation score $Score_{tsk}$ between tasks.
Let F_i be the feature vector of task i extracted by the analytics model. The value of $Score_{tsk}$ is calculated by defined task priority P_i and cosine similarity between two feature vectors:

$$Score_{tsk} = \sum_{i=1}^{N} \left(P_i \cdot \sum_{j=1, j\neq i}^{N} \cos\left(F_i, F_j\right) \right) \tag{4}$$

4.3 Model Performance Score

Since we focus on multi-task video analytics, each type of task has a corresponding set of models. For the same task, different models may vary in terms of accuracy and latency. To accurately quantify the performance of model, we design a model performance score $Score_{mdl}$ described in Eq.5, which integrates model accuracy and latency, aiming to identify the performance of model under the current system state.

$$Score_{mdl} = Acc_{mdl} \cdot e^{-k \cdot L_{mdl}} \tag{5}$$

where Acc_{mdl} represents the accuracy of video analytics models and L_{mdl} denotes the latency of each task. Besides, an exponential decay function is applied as a constraint to emphasize real-time requirements, with k serving as the decay coefficient.

4.4 Resource Utilization Score

Resources on edge devices are limited, so resource utilization efficiency has become one of the key factors in system optimization. Especially in multi-task scenarios, models of various tasks need to efficiently share and utilize the limited system resources. The resource utilization score $Score_{rsc}$, comprehensively takes into account the estimated resource consumption such as memory usage and the occupancy of CPU and GPU. can be obtained by:

$$Score_{rsc} = U_{Memory} \cdot U_{CPU} \cdot U_{GPU} \tag{6}$$

where U_{CPU} and U_{GPU} denote the CPU and GPU utilization of system, respectively. U_{Memory} represents the system memory usage rate, which can be calculated from the current used memory M_u and the model switching cost M_{mdl} as follows:

$$U_{Memory} = \frac{M_u + M_{mdl}}{M_t} \tag{7}$$

5 MODEL SCALING AND SCHEDULING

5.1 Problem Definition

The optimization problem in Section 2.3 has multiple optimization objectives. The process of finding the Pareto optimal solution is quite complex, and it is difficult to make real-time decisions. Combining all the scoring metrics discussed above, we define a comprehensive metric $TScore$, which integrates the key aspects of videos, tasks, models and resource factors. Then we re-define the problem of minimizing latency as the problem of maximizing $TScore$.

For edge video analytic system, $TScore$ is positively correlated with the accuracy and the system resource utilization (below the threshold), and negatively correlated with the system analytic latency. Therefore, the $TScore$ can be calculated by Eq.8:

$$TScore = \alpha \cdot Score_{cmplx} \cdot Score_{tsk} \cdot Score_{mdl} + \beta \cdot Score_{rsc} \quad (8)$$

incorporates two parameters, α for analytic models and β for system resource utilization, allowing it to adapt to various edge scenarios with differing optimization goals. And the problem can be formulated as:

$$\underset{mdl}{arg\,max} \sum_{tsk=1}^{N} TScore \#(9) \quad (9)$$

Our objective is to maximize the sum of model $TScore$ across all tasks through model scaling and scheduling, which serves as the system's comprehensive score.

5.2 Working Process

According to the definitions in Section 5.1, the working process of the model scaling and scheduling module is shown in Figure 3.

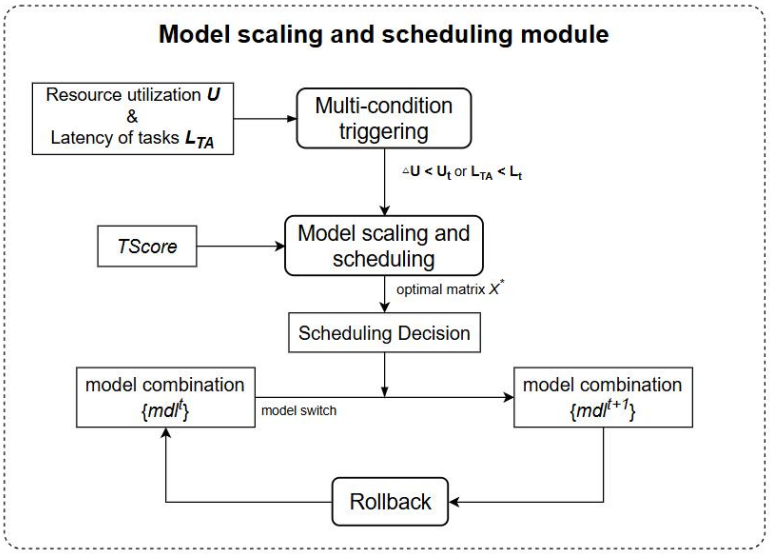


Figure 3 The Process of Model Scaling and Scheduling

As illustrated, this process consists of the following parts.

5.2.1 Multi-condition triggering.

The edge video analytic scenarios are highly dynamic, with the video complexity and system load changing in real time. In our framework, the appropriate timing for triggering the scaling and scheduling algorithm is particularly crucial. The multi-condition triggering mechanism determines when to evaluate model performance, recalculate, and perform model scaling and scheduling based on the score. In the edge scenario, two relatively important metrics are system resource utilization and the latency of video analytic tasks. The triggering conditions are:

(1) System resource utilization: The change ΔU in the utilization of CPU, GPU, or memory exceeds utilization change threshold U_τ or the utilization itself does not meet the interval value.

(2) Latency of video analytic tasks: The latency L_{TA} fluctuates and tend to be greater than threshold L_τ.

Meanwhile, we have set a triggering interval time, within which the models can only be scaled and scheduled at most once.

5.2.2 PSO-based model scaling and scheduling

The model scaling and scheduling module drive scheduling decisions based on the Particle Swarm Optimization (PSO) algorithm, aiming to maximize the system $TScore$. This section will introduce its definition and the solving process.

Particle representation. Given N video analytical tasks with M model variants per task, each particle is represented as a matrix $X_{M\times N} = (x_{ij})$, where element x_{ij} indicates the selection status of model j for video analytic task i. The swarm size is dynamically configured based on task load and problem complexity.

Velocity update mechanism. Particle velocity $V_{M\times N} = (v_{ij})$ determines search direction and step size, with element v_{ij} representing selection tendency for model x_{ij}. The velocity update rule integrates three components:

$$v_{ij}(t+1) = \omega_v \cdot v_{ij}(t) + c_1 r_1 \left[pb_{ij} - x_{ij}(t)\right] + c_2 r_2 \left[gb_{ij} - x_{ij}(t)\right] \quad (10)$$

where c_1 and c_2 are cognitive coefficient and social coefficient. r_1 and r_2 are uniform random variables for subsequent searching. Vector pb_{ij} represents the particle's historical best position, and vector gb_{ij} represents swarm's global best position, referring to the model scaling and scheduling consequence that achieves the maximum $TScore$ value in the current system state. ω_v is the adaptive inertia weight controlling global and local search balance, and it follows a linear decay scheme:

$$\omega_v = \omega_{init} - \frac{t}{T_{max}}(\omega_{init} - \omega_{end}) \tag{11}$$

The algorithm initializes with higher weight ω_{init} to enhance global exploration, and progressively decrease to ω_{end} for intensified local exploitation as iterations approach T_{max}.

Fitness Function. The optimization objective is to maximize the composite performance score $TScore$. Therefore, the fitness function is:

$$Fitness(X_{M\times N}) = \sum_{tsk=1}^{N} TScore \tag{12}$$

Scheduling Process. First, the PSO algorithm initializes the initial positions and velocities of particles in the particle swarm based on the analytical task and model information, and sets a series of parameters such as learning factors c and the number of iterations T_{max}.

Then, according to the model performance scoring module, the fitness values of each particle are calculated respectively. Then, through comparison, the historical optimal position of each particle and the historical optimal position of the swarm are updated. If the maximum number of iterations T_{max} is reached or the difference between two consecutive fitness values is less than the threshold, the loop terminates and the optimal position is output; otherwise, it continues to calculate the new fitness value for a new round.

Finally, the algorithm output matrix $X^*_{M\times N}$ based on the global best position. This matrix corresponds to the currently optimal model combination to be scheduled.

Once the PSO algorithm generates the model scheduling decision, the module preloads the selected models for each task and gradually redirects the input video frames from the current models to the new selected models in a data-driven manner until this process is complete.

5.2.3 Rollback

The rollback mechanism is designed to retain a snapshot of models before scheduling. When the scheduled models combination encounter an anomaly and fail to meet the requirements of edge video analytics, such as a prolonged decline in $TScore$, the system will automatically rollback to the previous models combination, and wait for the next time interval to perform new scheduling.

6 EXPERIMENTS

6.1 Implementation

Video analytic tasks. To verify the performance of the proposed framework, we selected common abnormal behavior detection tasks: intrusion detection [14], fall detection [15], loitering detection [16], and crowd detection [17]. We use $TScore$ as evaluation metrics. In intrusion detection task, we selected mAP50 as the accuracy metric. In fall detection task, we selected precision as the accuracy metric. In loitering detection task, we selected the Identity F1 score (IDF) as the accuracy metric. The IDF can be calculated by the Identity Precision (IDP) and Identity Recall (IDR) metrics:

$$IDF = \frac{2 \cdot IDP \cdot IDR}{IDP + IDR} \tag{13}$$

In crowd detection task, we selected the mean squared error (MSE) as the accuracy metric.

The abnormal behavior detection in four tasks was implemented based on the YOLOv8 model. YOLOv8 is an advanced object detection model that integrates an optimized neural network architecture, dynamic input adaptation, and multi-scale training techniques. For intrusion detection task, we used YOLOv8 to detect intruding pedestrians within the set target area. For fall detection task, we used YOLOv8 to recognize human postures in the frame and make fall judgments. For loitering detection task, we used YOLOv8 and DeepSORT [18] to locate and track pedestrians in the video, and determined loitering based on the movement trajectories and residence time. For crowd detection task, we used YOLOv8 for crowd counting and calculated the crowd density in the frame accordingly.

When training these video analytic models, we set the learning rate to 0.001, the batch size to 4, and the training epochs to be between 300 and 500. In the multi-condition triggering mechanism proposed in Section 5.2.1, we set the utilization change threshold U_τ and the latency threshold L_τ to 20% and 150 ms, respectively.

Workload. We selected three types of datasets as the workloads for video analytic tasks. For intrusion detection and fall detection task, we selected the Le2i Fall detection Dataset as the workload. This dataset was captured by real surveillance cameras and contains videos of various fall behaviors as well as video frames without pedestrians. For loitering detection and crowd detection task, we selected the MOT16 and MOT20 datasets [20] as the workloads. The MOT16 dataset includes pedestrian tracking in different scenarios, while the MOT20 dataset has a denser pedestrian scenario.

6.2 Baselines

We compared our method with the following frameworks:

(1) Static Model Deployment: a basic method where video analytic models are statically deployed on the experimental platform without dynamic adjustments based on real-time operating conditions.

(2) CerberusDet [5]: a unified multi-dataset object detection framework, employs an innovative architecture design and advanced feature fusion techniques to fully exploit the complementary characteristics across datasets, thereby enhancing detection accuracy.

(3) Learning-Based Query Scheduling and Resource Allocation (LQSRA) [6]: an adaptive mobile-edge-coordinated RL -based framework to handle unpredictable query arrivals and fluctuant available resources.

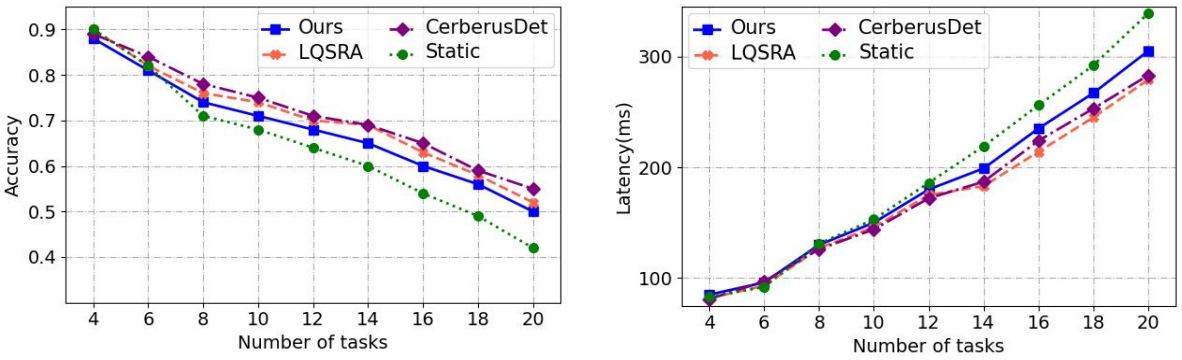
6.3 Performance Evaluation

We evaluate our approach based on accuracy and latency, and compute the $TScore$ accordingly. Experimental results are presented in Table 1 and Figure 3, respectively.
In terms of accuracy, our method achieves a 6.0% improvement compared to the static method but decreases by 6.7% and 5.4% compared to CerberusDet and LQSRA, respectively. As for latency, our method reduces latency by 5.0% compared to static deployment but incurs 1.9% and 5.2% latency increases compared to CerberusDet and LQSR. The static deployment method lacks any scheduling optimization, leading to poor accuracy in multi-task edge video analytic. CerberusDet enhances video analytics accuracy for highly similar tasks. However, for dissimilar tasks (e.g., object detection and object tracking), interference between multiple tasks reduces its average detection accuracy and increase latency of both tasks. LQSRA employs the PPO method for task scheduling and resource allocation, addressing resource shortages in high-load edge scenarios by selecting lightweight models.
During scheduling, our method incorporates accuracy as part of the optimization objective (Eq 5), performing model scaling and scheduling by maximizing the comprehensive $TScore$. Although our approach does not have advantages in single metrics like accuracy or latency, there are improvements of 41.9%, 17.3%, and 11.4% compared with static method, CerberusDet, and LQSRA, respectively. The $TScore$ calculation results demonstrate that it achieves the highest $TScore$ by optimizing both model performance and resource utilization of video analytic system.

Table 1 Experiment Results

Framework	Task	Accuracy	Average accuracy	Latency(ms)	$TScore$
Ours	Intrusion	0.73	0.70	262	0.88
	Fall	0.70			
	Loitering	0.67			
	Crowd	0.69			
Static	Intrusion	0.69	0.66	276	0.62
	Fall	0.68			
	Loitering	0.63			
	Crowd	0.63			
CerberusDet	Intrusion	0.79	0.75	257	0.75
	Fall	0.77			
	Loitering	0.71			
	Crowd	0.74			
LQSRA	Intrusion	0.75	0.74	249	0.79
	Fall	0.74			
	Loitering	0.72			
	Crowd	0.73			

As shown in Figure 4, in multi-task scenarios (with the number of tasks > 4), compared to CerberusDet and LQSRA, both accuracy and latency fluctuations of our method remain within 10%. The static method achieves lower video analytic latency under fewer tasks. As the number of tasks grows, it experiences significant latency fluctuations due to the lack of multi-task optimization. Compared to our approach, CerberusDet and LQSRA focus on single optimization objectives, thus demonstrating certain advantages in accuracy or latency.



(a) Comparison of accuracy (b) Comparison of latency

Figure 4 Comparison of Accuracy and Latency under Multi-Task Scenario

7 CONCLUSION

To address the challenges of redundant computations and single-objective scheduling in edge video analytics, we propose a model scaling and scheduling framework driven by a comprehensive performance metric $TScore$ and PSO-based optimization. The $TScore$ holistically evaluates video complexity, task dependencies, model accuracy-latency trade-offs, and resource utilization, enabling adaptive model selection. Experimental results on edge devices demonstrate that our framework achieves the best $TScore$ performance. The elastic scheduling mechanism ensures real-time responsiveness and robust quality of service in scenarios like fall detection and crowd monitoring. The current framework requires manual calibration of $TScore$ parameters for specific scenarios. Future directions include (1) self-adaptive parameter tuning via online meta-learning to enhance plug-and-play adaptability and (2) hardware-aware lightweight scheduling through neural architecture search (NAS)-enhanced PSO for heterogeneous platforms.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Bogdoll D, Nitsche M, Zöllner J M. Anomaly detection in autonomous driving: A survey. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 4488-4499.
[2] Tay N C, Connie T, Ong T S, et al. A review of abnormal behavior detection in activities of daily living. IEEE Access, 2023, 11: 5069-5088.
[3] Zhang Q, Sun H, Wu X, et al. Edge video analytics for public safety: A review. Proceedings of the IEEE, 2019, 107(8): 1675-1696.
[4] Xu R, Razavi S, Zheng R. Edge video analytics: A survey on applications, systems and enabling techniques. IEEE Communications Surveys & Tutorials, 2023, 25(4): 2951-2982.
[5] Tolstykh I, Chernyshov M, Kuprashevich M. CerberusDet: Unified Multi-Dataset Object Detection. arXiv preprint arXiv:2407.12632, 2024.
[6] Lin J, Yang P, Wu W, et al. Learning-based query scheduling and resource allocation for low-latency mobile-edge video analytics. IEEE Internet of Things Journal, 2023, 11(3): 4872-4887.
[7] Ma H, Li R, Zhang X, et al. Reliability-aware online scheduling for DNN inference tasks in mobile-edge computing. IEEE Internet of Things Journal, 2023, 10(13): 11453-11464.
[8] Wang C, Yang P, Hou J, et al. Dependence-aware multi-task scheduling for edge video analytics with accuracy guarantee. IEEE Internet of Things Journal, 2024, 11: 26970-26983.
[9] Ahmad S, Guan H, Friedman B D, et al. Proteus: A high-throughput inference-serving system with accuracy scaling. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2024(1): 318-334.
[10] Yang Y, Shen H, Tian H. Scheduling workflow tasks with unknown task execution time by combining machine-learning and greedy-optimization. IEEE Transactions on Services Computing, 2024, 17(3): 1181-1195.
[11] Zhou X, Liang W, Yan K, et al. Edge-enabled two-stage scheduling based on deep reinforcement learning for internet of everything. IEEE Internet of Things Journal, 2022, 10(4): 3295-3304.
[12] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, 1995, 4: 1942-1948.
[13] Jain M, Saihjpal V, Singh N, et al. An overview of variants and advancements of PSO algorithm. Applied Sciences, 2022, 12(17): 8392.
[14] Mei X, Zhou X, Xu F, et al. Human intrusion detection in static hazardous areas at construction sites: Deep learning–based method. Journal of Construction Engineering and Management, 2023, 149(1): 04022142.
[15] Chen B, Chen C, Hu J, et al. Computer vision and machine learning-based gait pattern recognition for flat fall prediction. Sensors, 2022, 22(20): 7960.

[16] Huang T, Han Q, Min W, et al. Loitering detection based on pedestrian activity area classification. Applied Sciences, 2019, 9(9): 1866.
[17] Song Q, Wang C, Jiang Z, et al. Rethinking counting and localization in crowds: A purely point-based framework. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021: 3365-3374.
[18] Wojke N, Bewley A, Paulus D. Simple online and realtime tracking with a deep association metric. In: Proceedings of the IEEE International Conference on Image Processing, 2017: 3645-3649.
[19] Shao J, Zhang X, Zhang J. Task-oriented communication for edge video analytics. IEEE Transactions on Wireless Communications, 2023, 23(5): 4141-4154.
[20] Dendorfer P, Osep A, Milan A, et al. Motchallenge: A benchmark for single-camera multiple target tracking. International Journal of Computer Vision, 2021, 129: 845-881.