Tackling the Imbalance in Video Analytics Pipelines with Hierarchical Embodied Intelligence

Wenhui Zhou, Lei Xie, Jingyi Ning, Shuyu Cao, Hao Wu, Qinghua Peng, Long Fan
State Key Laboratory for Novel Software Technology, Nanjing University, China
whzhou@smail.nju.edu.cn, {lxie, ningjy}@nju.edu.cn, {sycao, wuhao, pengqh, fanl}@smail.nju.edu.cn

***Abstract*—With the proliferation of heterogeneous software-hardware infrastructures in camera deployment, video analytics pipelines (VAPs) are increasingly burdened by spatiotemporal workload imbalance, where uneven task distributions lead to latency constraint violations and degraded quality of experience (QoE). Eliminating this imbalance is challenging due to the inherent complexity of adjusting large-scale parameters and the dynamic nature of VAP runtime environments. To this end, we propose Hier-EI, a novel scheduling framework that combines the two-phase hierarchical design with embodied intelligence, which adaptively tunes system parameters to mitigate imbalance and maintain long-term service-level objective (SLO) performance for modern VAP systems. To tackle the complexity of combinatorial decision-making, we introduce a hierarchical collaboration mechanism with macro-micro coordination that transforms the exponential search space into a linear coarse-to-fine workflow. To adapt to runtime dynamics, we present an embodied feedback mechanism that employs closed-loop feedback to converge toward real-time optimal solutions as a Markov decision process. Extensive evaluations on a real-world prototype system built on KubeEdge demonstrate that Hier-EI achieves a 3.6× improvement in latency compliance, and a 67.4% reduction in P95 latency compared with state-of-the-art scheduling methods.**

***Index Terms*—edge computing, video analytics pipelines, hierarchical scheduling, embodied intelligence**

I. Introduction

Real-time video analytics pipelines (VAPs) are increasingly shifting toward cloud-edge collaboration with heterogeneous hardware and software, powering applications like autonomous systems [1]–[3], surveillance [4]–[6], and augmented reality [7], [8]. These applications typically adopt multi-stage processing [9]–[11] to balance latency and accuracy, thereby enhancing user quality of experience (QoE) [12]–[14]. However, in real-world deployments, VAPs often encounter systemic imbalance caused by unpredictable task processing delays within end-to-end pipelines [15], [16]. These delays arise from task accumulations in heterogeneous soft/hardware, manifesting in both *spatial* and *temporal* dimensions, as illustrated in Fig. 1. Specifically, fluctuations in network conditions [17] across spatial cloud-edge systems of heterogeneous hardware (i.e., devices with different resources and positions) result in transmission bottlenecks, while throughput mismatches [18] within temporal pipelines composed of heterogeneous software (i.e., models with different complexity) trigger congestion between processing stages. *Therefore, is there an effective solution to the temporal and spatial imbalance in cloud-edge collaborative systems, enhancing long-term user QoE?*

Existing solutions to enhance user QoE focus on adjusting system "*knob*" [19] (i.e., tunable parameters like frame resolution) and they fall into three categories: i) Profiling-based methods build off/online profiles to guide knob tuning [17], [19]–[23]. However, offline profiling depends on estimation experiences and thus cannot adapt to runtime dynamics of spatiotemporal imbalance [22], [23]; online profiling incurs prohibitive overhead as configuration dimensions scale, burdening the maintenance of long-term QoE [17], [19]–[21]. ii) End-to-end learning methods leverage black-box deep neural networks to directly output all knob values [8], [24]–[26], but struggle to converge in large-scale multi-knob scenarios when tackling spatiotemporal imbalance [27], [28]. iii) Negative feedback methods provide low-cost adjustments based on latency feedback, only supporting a single knob [29]–[31] and failing to expand to multiple knobs with coupling relationship.

To simultaneously consider spatial and temporal imbalance and enhance long-term QoE performance in cloud-edge collaborative VAP, we propose **Hier-EI**, an efficient scheduling framework that integrates a two-phase hierarchical design with embodied intelligence. Our Hier-EI interacts with VAP systems as an embodied intelligence and adaptively adjust multiple system knobs with hierarchical decisions in real time, responding to the burdens of spatiotemporal imbalance and maintaining system performance.

Although the basic concept appears straightforward, fully eliminating the imbalance through this novel scheduling scheme presents two non-trivial challenges. First, achieving long-term QoE performance under spatiotemporal imbalance requires joint optimization over multiple knobs, whose combinations can easily exceed millions (e.g., over 1.1 million when using 5 knobs), leading to exponential growth in the decision space and significantly increasing the complexity of finding feasible solutions. Second, the runtime context of resources and tasks fluctuates continuously in cloud-edge VAP systems, in which the real-time optimal solutions shift dynamically, impeding timely responses and complicating the attainment of near-optimal scheduling decisions.

To handle the complexity of million-level decision space, instead of end-to-end decision-making, we introduce a hierarchical collaboration mechanism, leveraging two phases to break down exponentially combinatorial decision space into a linear coarse-to-fine process. Specifically, in the first phase, a macro-scheduler with a deep reinforcement learning (DRL) generates coarse-grained decisions (e.g., whether to increase,

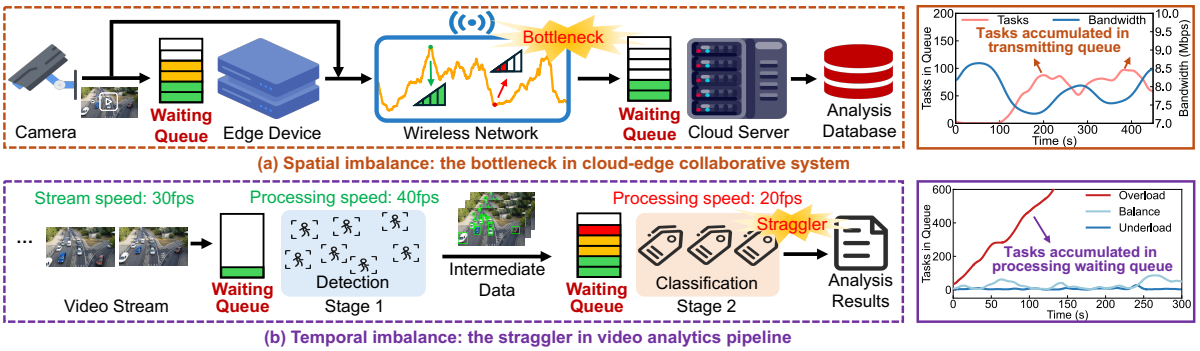


Fig. 1. The imbalance in VAPs: (a) Spatial imbalance caused by network transmission latency in heterogeneous hardware; (b) Temporal imbalance resulting from mismatched processing speeds across stages of heterogeneous software (e.g., 40fps Stage 1 vs. 20fps Stage 2).

decrease, or maintain frame resolution) for all knobs. These coarse-grained decisions provide macro-level guidance and have less complexity compared with final decisions. In the second phase, based on the coarse-grained decisions, a micro-scheduler with negative feedback (NF) methods fine-tunes knobs linearly. By assigning an independent NF method to each knob, we obtain the final fine-grained decisions (e.g., the exact frame resolution value). Meanwhile, we asynchronously coordinate the two phases with different scheduling intervals to further reduce interdependencies.

To adapt to the dynamics of runtime context, we present an embodied feedback mechanism interacting with systems based on a Markov decision process (MDP) and converging toward real-time near-optimal solutions with closed-loop feedback. Since states of VAP systems evolve probabilistically based on their previous states and the latest knob adjustment, the long-term sequential knob tuning maps to a MDP formulation. Inspired by the robot control methodology [32], we perform runtime context perception in VAP systems, including resource context (i.e., system's supply, such as network bandwidth) and task context (i.e., task's demand, such as object cardinality). We then analyze these runtime contexts to extract underlying *supply-demand* relationships, providing key insights for rapid adjustments via closed-loop feedback, which in turn alters system states and triggers new interactions within the MDP cycle. Our contributions can be summarized as follows:

- We propose Hier-EI, a scheduling framework that rethinks temporal and spatial imbalance to enhance long-term user QoE in VAPs.
- We introduce a novel hierarchical collaboration mechanism with macro-micro coordination that reduces the exponential decision space to linear complexity.
- We present an embodied feedback mechanism that leverages closed-loop feedback to interact with systems and adapt dynamically to runtime variations.
- We implement a real-world prototype, and experimental results show that Hier-EI achieves a 3.6× improvement in latency compliance and a 67.4% reduction in P95 latency compared with state-of-the-art methods.

II. Preliminary

In this section, we first present the imbalance from spatial and temporal perspectives. Then, we examine how knobs and runtime contexts influence task performance, supporting the problem formulation in Section III. Preliminary experiments are conducted on our real-world system in Section IV.

A. Understanding the Imbalance

Real-world VAPs often exhibit spatiotemporal imbalance, leading to unexpected task delays. We demonstrate this from spatial and temporal perspectives.

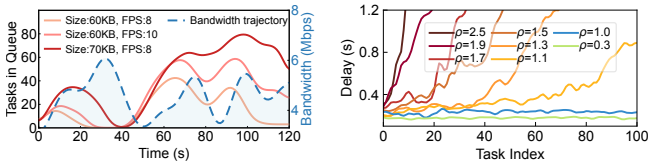
Spatial imbalance. Video streams are typically generated at edge cameras and transmitted to the cloud servers. Due to limited and fluctuating wireless bandwidth, task accumulation often occurs in transmitting queues, creating spatial imbalance between edge and cloud. As shown in Fig. 2(a), task backlog intensifies under bandwidth variations (Cellular dataset [33]), especially when data size or frame rate increases, resulting in higher transmission latency.

Temporal imbalance. Pipeline stages with mismatched throughput also induce imbalance. Fig. 2(b) shows end-to-end delays under varying throughput ratios ρ (camera frame rate vs. detection-stage execution rate). When $\rho \leq 1$, the system remains stable without congestion; however, when $\rho > 1$, task delays grow rapidly due to task accumulation in detection's waiting queue. Any stage with throughput lower than the input or upstream rate becomes a temporal bottleneck.

Summary: Spatiotemporal imbalance, caused by supply-demand mismatches in VAP systems, is a key challenge for sustaining robust QoE.

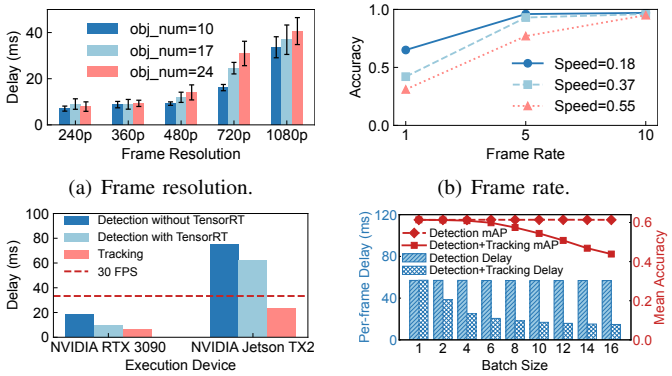
B. Knob Selection

A typical VAP consists of three basic roles: source, detection, and classification. Without loss of generality, we conceptualize VAPs using a source–detection–classification (SDC) framework, which guides the exploration and selection of optimization knobs.



(a) Spatial bottleneck. (b) Temporal straggler.

Fig. 2. Spatiotemporal imbalance: (a) Bandwidth dynamics induce spatial imbalance; (b) Throughput mismatch across stages causes temporal imbalance.



(a) Frame resolution. (b) Frame rate.

(c) Execution devices. (d) Buffer size.

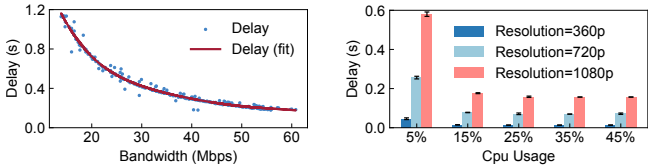
Fig. 3. Knob selection: knobs are considered from video configuration of (a) and (b), execution acceleration of (d), and distributed collaboration of (c).

Video configuration. The source in the SDC framework continuously generates video stream, in which video configuration (e.g., frame resolution, frame rate) is adjustable as control knobs. As shown in Fig. 3(a), the inference delay increases as frame resolution is improved with higher computation pressure. Besides, Fig. 3(b) shows the accuracy loss with the decrease of frame rate due to lagging responses. Therefore, adjusting video configuration like *frame resolution* and *frame rate* can help optimize task performance.

Execution acceleration. The detection in the SDC framework processes frames and directly identifies regions of interest (e.g., detect vehicles on the road). Fig. 3(c) shows that detection incurs large delays (below 30fps), even with optimization like NVIDIA TensorRT [34]. One acceleration strategy is to combine lightweight tracking [35] with heavyweight detection. Specifically, we group adjacent frames into segments, apply detection only to the first frame and tracking to subsequent frames. Fig. 3(d) demonstrates that larger buffer size of segments substantially reduces inference delays, albeit at some accuracy loss. Therefore, selecting an appropriate *buffer size* is crucial to balancing latency and accuracy.

Distributed collaboration. The classification in the SDC framework sequentially processes regions caught by detection (e.g., identify license plates on detected vehicles). Efficiency can be improved through distributed collaboration. On one hand, cloud–edge collaboration utilizes powerful cloud servers (in Fig. 3(c)) but introduces additional transmission delays. Dynamically partitioning pipeline stages between cloud and edge based on network conditions and data size can alleviate these delays. On the other hand, edge–edge collaboration distributes detected regions among multiple edge devices for parallel processing. Hence, using *pipeline partition policy* and *region allocation policy* helps reduce latency.

Complexity: Without loss of generality, we select knobs in Tab. I to eliminate imbalance. The partition point refers to the cloud-edge split of the pipeline, while region allocation involves assigning R regions to U edge devices, resulting in $\binom{R+U-1}{U-1}$ possible combinations. The overall decision space is thus expressed as $N^{\text{res}} \cdot N^{\text{fps}} \cdot N^{\text{bs}} \cdot N^{\text{pp}} \cdot N^{\text{ra}}$. For example, when $R = 18$ and $U = 3$, the total number of



(a) Network bandwidth. (b) CPU usage.

Fig. 4. Resource runtime context: dynamic states of network resource in (a) and hardware resource in (b) influence task performance.

knob combinations in Tab. I reaches 1,197,000, highlighting the need to handle high-dimensional decision complexity.

C. Runtime Context

Runtime context reflects real-time states of VAP systems. Understanding this context reveals underlying supply-demand relationships, facilitating dynamic knob adjustments. Here, we focus on resource and task context.

Resource context. Resource states of network and hardware are crucial for estimating offloading capabilities. Fig. 4(a) shows that lower bandwidth prolongs transmission delay, while Fig. 4(b) illustrates that CPU usage significantly influences processing delays. Thus, resource context perception helps effective management of hardware and network variations.

Task context. Video content characteristics also influence task performance. Fig. 3(a) indicates that increased object cardinality extends inference delays, whereas Fig. 3(b) shows object speed can degrade accuracy. Therefore, incorporating task context into scheduling decisions is necessary to adapt to dynamic environments.

Dynamics: Continuous changes in runtime context require adaptive knob adjustment to align resource supply with task demand, ensuring latency and accuracy performance.

III. System Design

In this section, we formulate the imbalance problem as high-dimensional MDP. Building on this formulation, we introduce our Hier-EI method with efficient macro-micro collaboration.

A. Problem Formulation

In a VAP system, task T_t denotes a video segment at time t. The end-to-end latency L_t and accuracy A_t of task T_t are:

$$\begin{cases} L_t = \phi_L(\boldsymbol{s}_t, \boldsymbol{\tau}_t) \\ A_t = \phi_A(\boldsymbol{s}_t, \boldsymbol{\tau}_t) \end{cases}, \tag{1}$$

where $\boldsymbol{s}_t$ denotes runtime states, $\boldsymbol{\tau}_t$ denotes knob decisions, and functions $\phi_L(\cdot)$ and $\phi_A(\cdot)$ represent latency and accuracy

TABLE I
Selected Knobs.

Category	Knobs	Optional Values
Configuration	Resolution (N^{res})	240p,360p,480p,540p,720p,900p,1080p
	Frame rate (N^{fps})	1, 2, 3, 4, 5, 6, 7, ..., 29, 30
	Buffer size (N^{bs})	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Offloading	Partition point (N^{pp})	[cloud,cloud], [edge,cloud], [edge,edge]
	Region allocation (N^{ra})	$\binom{R+U-1}{U-1}$ combinations

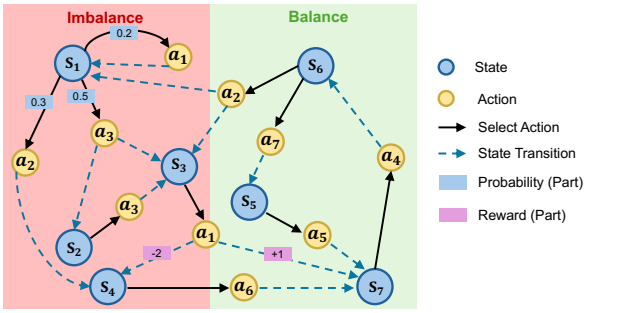


Fig. 5. High-dimensional MDP of tackling imbalance.

respectively. Since s_t fluctuates continuously in real-world environments with spatiotemporal imbalance, it is necessary to make knob decisions τ_t accordingly to maintain performance.

To address this requirement, without loss of generality, we define a *latency-first* optimization objective:

$$\max_{\tau_t} \sum_t A_t, \qquad s.t.\ L_t \leq \frac{1}{f_t}, \quad \forall t, \tag{2}$$

where f_t is the source frame rate, serving as the latency constraint. Thus, our objective is to *maximize accuracy while ensuring real-time end-to-end latency constraints.*

Knob decision τ_t has a combinatorial structure:

$$\tau_t = \left(\tau_t^{k_1}, \tau_t^{k_2}, \ldots, \tau_t^{k_n}\right) \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n, \tag{3}$$

where n is the number of knobs, and each knob k_i has a finite set of discrete values $\mathcal{X}_i = \left\{x_i^{(1)}, \ldots, x_i^{(N_i)}\right\}$ with cardinality N_i. The selected value for knob k_i at time t is denoted as $\tau_t^{k_i} = x_i^{(j_{i,t})}$, where $j_{i,t} \in 1, \ldots, N_i$ indexes the chosen value in $\mathcal{X}_i$. As n or N_i grows, the decision space size ($\prod_{i=1}^{n} N_i$) increases exponentially, posing significant complexity.

Therefore, tackling imbalance in VAP systems can be formulated as a high-dimensional MDP, as illustrated in Fig. 5. In this MDP, the agent makes sequential decisions by selecting actions from a large space of knob combinations (e.g., over 10^6 in Tab. I), based on dynamic runtime states. Through interaction with the system, it learns policies that optimize cumulative rewards reflecting imbalance levels, thereby mitigating spatiotemporal imbalance and maintaining system stability.

B. Hier-EI Overview

For real-time knob adjustment within this high-dimensional MDP, we propose **Hier-EI**, a two-phase Hierarchical scheduler with Embodied Intelligence. As shown in Fig. 6, Hier-EI comprises macro- and micro-scheduling. We first introduce each module individually, followed by two coordination mechanisms that facilitate macro–micro collaboration to reduce decision complexity and improve dynamic adaptability.

Macro-scheduling. To capture system imbalance situations, we employ an Actor-Critic DRL model for macro-scheduling. It perceives runtime context as input states and produces coarse-grained adjustment directions, providing global guidance for micro-scheduling.

Micro-scheduling. For quick and lightweight responsiveness, we employ NF methods for micro-scheduling. Each knob independently utilizes a NF method to generate fine-grained adjustments aligned with macro-level directions, ensuring timely imbalance responses.

Collaboration and optimization. A hierarchical collaboration mechanism asynchronously coordinates macro- and micro-scheduling, leveraging respective strengths and reducing complexity. Additionally, an embodied feedback mechanism dynamically converges toward optimal solutions through closed-loop interactions.

C. Knob Analysis

To effectively design our hierarchical scheduling structure, we categorize knobs from Tab. I into monotonic and non-monotonic based on their performance impacts.

Monotonic knobs. These knobs exhibit a predictable monotonic relationship with performance metrics like accuracy and latency. For example, increasing the frame resolution improves accuracy but also results in higher latency. In Tab. I, monotonic knobs include frame resolution τ^{res}, frame rate τ^{fps}, and buffer size τ^{bs}. Adjusting monotonic knobs should follow such relationships. Therefore, macro-scheduling decides adjustment direction (i.e., increase, decrease, and unchanged) in coarse-grained decisions, while micro-scheduling decides exact values in fine-grained decisions accordingly.

Non-monotonic knobs. These knobs show complex relationships with performance metrics, making it necessary to mine the interdependencies between them. In Tab. I, non-monotonic knobs include partition point τ^{pp} and regions allocation τ^{ra}. The partition point represents the split in the cloud-edge pipeline, with preceding stages offloaded to the edge and subsequent ones to the cloud. Region allocation represents

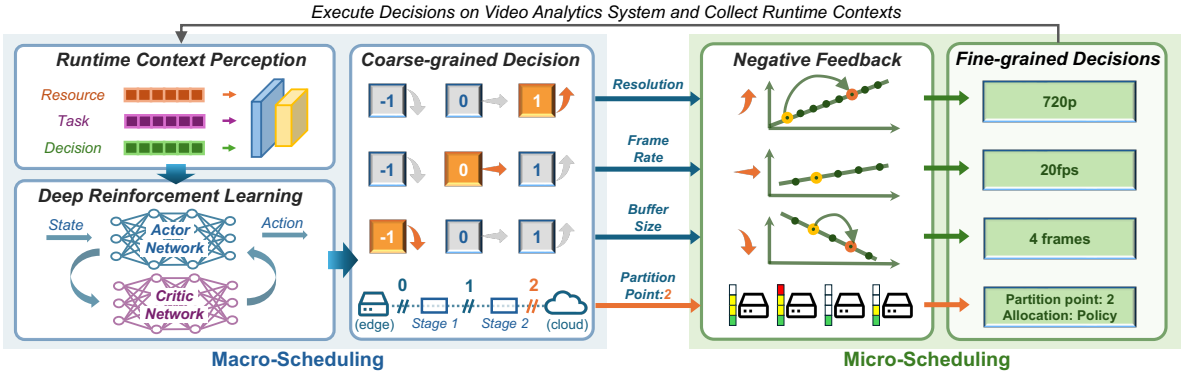


Fig. 6. The architecture overview of Hier-EI.

allocating regions on different devices in classification stage, and is based on partition point. Therefore, macro-scheduling decides partition point in coarse-grained decisions and micro-scheduling decides regions allocation in fine-grained decisions.

D. Macro-Scheduling

To capture dynamic states and coordinate knob adjustment from a global shared perspective, we design macro-scheduling that generates coarse-grained decisions with an Actor-Critic DRL framework [36], as shown in Fig. 6. The DRL agent takes runtime context as its input state and outputs coarse-grained knob decisions, while a QoE-based reward guides training towards policies tackling the imbalance.

Runtime context perception. To adapt to dynamic changes, we perceive runtime context as the DRL state. As illustrated in Fig. 7, the runtime context comprises resource, task, and decision features. Resource context contains the available bandwidth C_R^{bw} between edge and cloud, gathered via system monitoring to represent resource supply. Task context contains object size C_T^{os}, object cardinality C_T^{oc}, and historical task delay C_T^{d}, collected from previously processed tasks to reflect their resource demand. Decision context records Hier-EI's past knob adjustments for frame resolution C_D^{res}, frame rate C_D^{fps}, buffer size C_D^{bs}, and partition point C_D^{pp}, capturing historical decision patterns. By jointly modeling these three types of context, the DRL agent can track real-time supply–demand fluctuations and refine scheduling decisions in dynamic environments. At time t, the state input $\boldsymbol{s}_t$ of DRL is:

$$\boldsymbol{s}_t = (C_{R,t}^{\mathrm{bw}}, C_{T,t}^{\mathrm{os}}, C_{T,t}^{\mathrm{oc}}, C_{T,t}^{\mathrm{d}}, C_{D,t}^{\mathrm{res}}, C_{D,t}^{\mathrm{fps}}, C_{D,t}^{\mathrm{bs}}, C_{D,t}^{\mathrm{pp}}). \quad (4)$$

To incorporate temporal information and stabilize decisions under imbalance, we maintain a sliding window of length l for each of the eight context features, forming a (8,l)-sized matrix of recent records. We then apply one-dimensional convolutional neural networks (1D-CNNs) to extract temporal patterns from input states. One 1D-CNN is assigned to each context category (resource, task, decision), since features within the same category share more correlations. The extracted features are concatenated and fed into DRL networks, producing scheduling actions that adapt to evolving system conditions.

Coarse-grained decisions. Instead of directly outputting full knob decisions $\boldsymbol{\tau}_t$, macro-scheduling only outputs coarse-grained decisions $\boldsymbol{\xi}_t$ to ease the burden of the DRL model. The output action $\boldsymbol{a}_t$ of the DRL model is expressed as:

$$\boldsymbol{a}_t = \boldsymbol{\xi}_t = (\xi_t^{\mathrm{res}}, \xi_t^{\mathrm{fps}}, \xi_t^{\mathrm{bs}}, \xi_t^{\mathrm{pp}}). \quad (5)$$

For monotonic knobs, DRL outputs adjustment directions $(\xi_t^{\mathrm{res}}, \xi_t^{\mathrm{fps}}, \xi_t^{\mathrm{bs}}) \in \{-1, 0, 1\}$, denoting decrease, no change, and increase. For non-monotonic knobs, DRL outputs partition point decision ξ_t^{pp}. As shown in Fig. 6, the SDC pipeline supports three partition points (0, 1, 2): in case 0, both detection and classification run on the cloud; in case 1, detection runs on the edge and classification on the cloud; in case 2, both stages run on the edge.

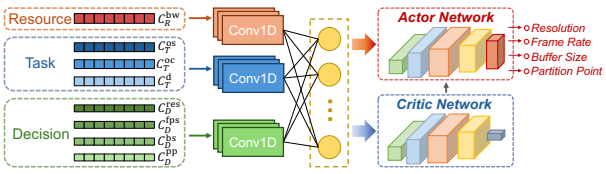


Fig. 7. Detailed architecture of macro-scheduling.

Compared with end-to-end DRLs, Hier-EI reduces action space complexity from $\prod_{i=1}^{n} N_i$ to 3^n (from about 10^6 to 81 in our knob set), which is essential for stable convergence.

QoE reward. To guide DRL toward resolving system imbalance and maintaining QoE, we design a reward aligned with the latency-first objective in Eq. (2):

$$\boldsymbol{r}_t = \begin{cases} \beta \cdot \dfrac{1}{\max(\Delta_{L,t}, \vartheta)} + A_t, & \Delta_{L,t} \geq 0, \\ \max(\gamma \cdot \Delta_{L,t}, \theta), & \Delta_{L,t} < 0, \end{cases} \quad (6)$$
$$\text{where} \qquad \Delta_{L,t} = \alpha \cdot \frac{1}{f_t} - L_t,$$

where A_t is task accuracy, f_t is the source frame rate, L_t is the end-to-end task latency, and $\Delta_{L,t}$ is the latency bias. When $\Delta_{L,t} \geq 0$, the reward is positively calculated by $\Delta_{L,t}$ and A_t due to constraint satisfaction. When $\Delta_{L,t} < 0$, a penalty is applied to discourage violations. The dynamic threshold f_t defines latency constraints, lowering f_t makes latency easier to satisfy but may reduce accuracy, encouraging the model to seek a balanced policy. Hyper parameters ($\beta = 0.3$, $\gamma = 20$, $\vartheta = 0.5$, $\theta = -2$, $\alpha = 1.6$) scale the objectives and soften constraints under concurrent execution.

During training, Hier-EI aggregates processed tasks within each scheduling interval, extracts A_t, L_t, and f_t from their records, and uses the average reward in Eq. (6) to update model weights and gradually improve decision quality.

E. Micro-Scheduling

To enable real-time knob adjustment in sudden system imbalance, a lightweight NF method is assigned to each knob for linear and low-cost value fine-tuning based on coarse-grained decisions. The fine-grained decisions are aggregated by outputs of all NF methods.

NF for monotonic knobs. Based on the monotonic performance relationship, adjusting monotonic knobs can be transformed to linear changes. Each monotonic knob k_i has an ordered value set $\mathcal{X}_i$ (e.g., resolution levels: 240p, 360p, 480p, and so on), and j_i is the index of the current knob value in $\mathcal{X}_i$. Thus, we can adjust k_i by updating j_i.

To quickly respond to system changes, we design a negative feedback policy inspired by AIMD (additive increase and multiplicative decrease [37]) to adjust index $j_{i,t}$:

$$j_{i,t} = \begin{cases} \min(j_{i,t-1} + 1, |\mathcal{X}_i|), & \xi_t^{k_i} = 1, \\ j_{i,t-1}, & \xi_t^{k_i} = 0, \\ \max(\lfloor j_{i,t-1}/2 \rfloor,\ 1), & \xi_t^{k_i} = -1, \end{cases} \quad (7)$$

where $\xi_t^{k_i}$ is the coarse-grained adjustment direction from macro-scheduling, and $|\cdot|$ denotes the cardinality of a set. This AIMD-based policy increases knob index linearly for

upward adjustments and halves it for downward adjustments, which adapts quickly to favorable conditions while responding conservatively to unfavorable conditions, balancing responsiveness and stability. The final adjustment knob value is determined by the updated index:

$$\tau_t^{k_i} = x_i^{(j_{i,t})}, \quad x_i^{(j_{i,t})} \in \mathcal{X}_i. \tag{8}$$

NF for non-monotonic knobs. The region allocation decision τ_t^{ra} is determined based on the coarse-grained partition point decision ξ_t^{pp} from macro-scheduling:

$$\tau_t^{ra} = \begin{cases} \{\sigma_t^0 = R_t\}, & \xi_t^{\mathrm{pp}} \in \{0,1\}, \\ \left\{ \sigma_t^\mu = \dfrac{\frac{1}{L_{t-1}^\mu}}{\sum_{\mu=1}^{U} \frac{1}{L_{t-1}^\mu}} \cdot R_t \right\}, & \xi_t^{\mathrm{pp}} \in \{2\}, \end{cases} \tag{9}$$

where R_t is the total number of detected regions, μ is the device index ($\mu = 0$ for cloud, $\mu = 1, \ldots, U$ for edge), and σ_t^μ denotes number of regions offloaded to device μ. When the classification stage is offloaded to the cloud ($\xi_t^{\mathrm{pp}} = 0$ or 1), all regions are processed centrally by cloud server ($\sigma_t^0 = R_t$). When classification is offloaded to the edge ($\xi_t^{\mathrm{pp}} = 2$), regions are distributed across U edge devices with a feedback-based policy. Specifically, the allocation to edge device μ is inversely proportional to its previous inference latency L_{t-1}^μ, favoring faster and lower-load devices for load balancing.

Fine-grained decisions. The fine-grained decisions $\boldsymbol{\tau}_t$ is:

$$\boldsymbol{\tau}_t = (\tau_t^{\mathrm{res}}, \tau_t^{\mathrm{fps}}, \tau_t^{\mathrm{bs}}, \tau_t^{\mathrm{pp}}, \tau_t^{\mathrm{ra}}), \tag{10}$$

which integrates the specific values of scheduling knobs and are executable in VAP systems. Resolution τ_t^{res}, frame rate τ_t^{fps}, and buffer size τ_t^{bs} are respectively output from monotonic feedback in Eqs. (7)–(8). Partition point τ_t^{pp} is directly from macro-scheduling ($\tau_t^{\mathrm{pp}} = \xi_t^{\mathrm{pp}}$). Region allocation τ_t^{ra} is output from non-monotonic feedback in Eq. (9).

F. Collaboration and Optimization

Based on the macro-micro scheduling structure, we design two collaborative mechanisms in Hier-EI to solve the high-dimensional MDP in real time and eliminate the spatiotemporal imbalance, as illustrated in Fig. 8.

Hierarchical collaboration mechanism. We coordinate two phases with a hierarchical asynchronous framework. The macro-scheduling (one DRL model) makes coarse-grained decisions ξ_t at a longer interval t_{ma}, whereas micro-scheduling (n NF methods) makes fine-grained decisions τ_t at a shorter interval t_{mi}. Each lightweight NF method references the latest coarse-grained decision to refine its knob. This decoupled design transforms the exponentially large decision space into a coarse-to-fine adjustment process, significantly reducing complexity in tackling severe imbalance.

Embodied feedback mechanism. To jointly optimize two hierarchical phases for dynamic adaptability, we treat Hier-EI as a unified embodied intelligence interacting with the environment. During each scheduling cycle, fine-grained decisions are performed in VAP systems to adjust knobs and guide task execution. Performance results are then used to compute DRL rewards with Eq. (6), and updated runtime context is perceived as the input states for the subsequent cycle. This closed-loop feedback loop allows continuous adaptation and convergence toward optimal scheduling in dynamic conditions.

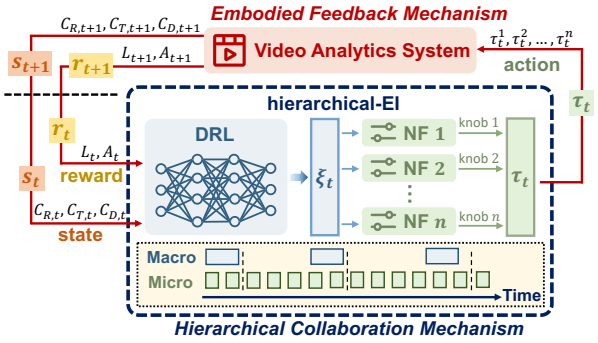


Fig. 8. Macro-micro collaboration mechanisms in Hier-EI.

IV. System Implementation

A. System Architecture

To verify our method in real-world settings, we implement a container-based prototype system[1] for VAPs, deployable across heterogeneous cloud-edge devices. Specifically, the prototype system has a five-layer architecture: (1) Basic system layer provides container orchestration in cloud–edge environments with KubeEdge [38]. (2) Intermediate interface layer facilitates deployment and communication by customizing KubeEdge plugins, including Sedna [39] and EdgeMesh [40]. (3) System supporting layer handles system interaction, providing frontend UI, backend API, and simulated datasource. (4) Collaborative scheduling layer manages fine-grained processing, scheduling, and monitoring for video stream analytics with functional components. (5) Application service layer provides a unified containerized interface for user-defined services and orchestrates them as pipeline applications.

B. Runtime Workflow

Components in collaborative scheduling layer, implemented as Docker containers [41], maintain the entire runtime workflow of VAPs, as illustrated in Fig. 9. The generator on edge devices captures video streams (e.g., RTSP) and segments them into tasks based on configuration decisions in meta information from the scheduler. The controller on each device oversees task lifecycle by routing tasks locally (local processor) or remotely (other device's controller) according to their current states. Tasks are processed by processors, stateless containers of single pipeline stages. As tasks progress through the pipeline, controllers coordinate cross-device transfers according to offloading decisions specified in meta information. Upon completion, results are sent to cloud-side distributor, which stores outputs and provides feedback (task and decision context) to the scheduler. Meanwhile, the monitor collects real-time resource metrics (e.g., hardware and network state) and updates the scheduler with the current resource context.

[1]Our prototype system is at https://dayu-autostreamer.github.io

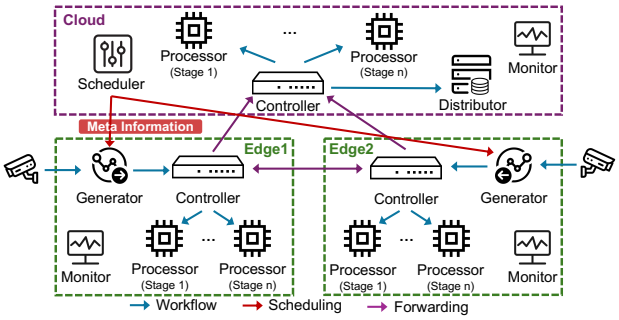
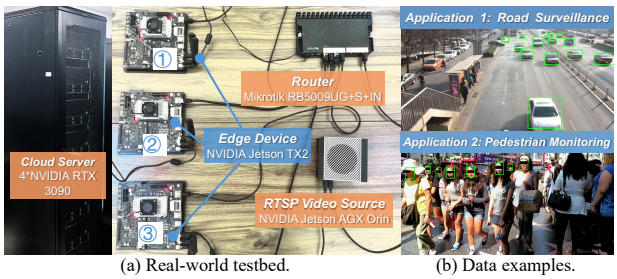


Fig. 9. The workflow of video analytics pipelines in the prototype system.



(a) Real-world testbed. (b) Data examples.

Fig. 10. Experimental setup and data examples.

The scheduler, equipped with an alternative decision-making algorithm implemented through *hook functions*, dynamically adjust knobs of data configuration and task offloading to optimize system performance.

V. Performance Evaluation

A. Methodology

Real-world testbed. We construct a real-world cloud-edge collaborative testbed, as shown in Fig. 10(a). The cloud server is equipped with four NVIDIA RTX 3090 GPUs, while the edge uses three NVIDIA Jetson TX2 [42]. An NVIDIA Jetson Orin [43] is used to push RTSP video streams. All edge devices connect via a Mikrotik router [44] over LAN. Our prototype system is deployed across this hardware setup.

Network setup. We deploy RouterOS [45] on router and replay real-world bandwidth trajectories from Belgium dataset [46] to emulate network dynamics between cloud and edge.

Scenarios setup. To explore the performance under various imbalance conditions, we define four scenarios combining network and task status: S1 (stable network, sparse objects), S2 (stable network, dense objects), S3 (unstable network, sparse objects), and S4 (unstable network, dense objects).

Hier-EI setup. We implement Hier-EI within the scheduler (see Section IV). Macro-scheduling uses a Soft Actor-Critic (SAC) [49] implemented in *PyTorch*, chosen for its entropy-maximizing balance between exploration and exploitation. Each network in SAC has three hidden layers with 256 neurons. 1D-CNNs in runtime context extraction are composed of three Conv-1D layers (kernel size 3, output size 64). The sliding window length l is 8. Scheduling intervals t_{ma} and t_{mi} are respectively set to 3s and 1s. The DRL model is trained for 20,000 steps and then applied directly in various applications.

VAP applications and datasets. As listed in Tab. II, we evaluate two applications: (1) A single-stage road surveillance application using 331-minute video from UA-DETRAC dataset [47] with ground truth annotations. (2) A two-stage pedestrian monitoring application using 275-minute YouTube video [48], with ground truth generated via the golden configuration [19]. Fig. 10(b) shows dataset examples. All services are optimized with NVIDIA TensorRT [34].

TABLE II
VAP applications in evaluation.

Application	Pipeline	Dataset
Road Surveillance	[Car Detection]	UA-DETRAC [47]
Pedestrian Monitoring	[Face Detection, Gender Classification]	YouTube videos [48]

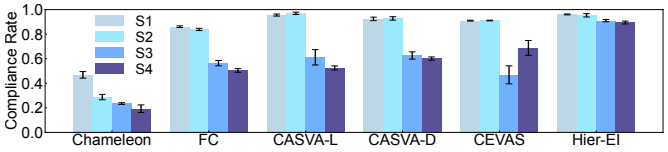
Baselines. We reproduce the following state-of-the-art baselines: (1) Chameleon [19]. Use online profiling to periodically adjust video configurations based on runtime accuracy estimation. (2) FC [31]. Use AIMD-based negative feedback to adjust frame resolution based on delay bias. (3) CASVA [24]. Use end-to-end DRL to select video configurations with latency-first (CASVA-L) and delivery-first (CASVA-D) modes. (4) CEVAS [50]: build profiles offline and solve convex optimization online to decide partition point.

Evaluation metrics. We use the following metrics to assess: (1) Latency compliance. Proportion of tasks meeting real-time latency constraints (derived from source frame rate), reflecting long-term load balance. (2) P95 latency. 95th-percentile end-to-end latency, characterizing SLO performance under extreme imbalance. (3) Average accuracy. Mean average precision (mAP) over detected regions, indicating overall quality.

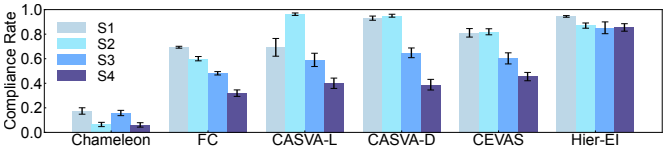
B. Overall Performance

The overall performance is summarized in Fig. 11–13. Each method is tested five times per scenario for robustness. For road surveillance application, Fig. 11(a) shows that Hier-EI consistently achieves over 90% latency compliance across all scenarios, outperforming baselines by 25.18%-247.14%, with an average improvement of 75.29%. In terms of accuracy in Fig. 12(a), Hier-EI is second only to FC with a 7.96% gap, but still delivers a 54.11% average improvement over other baselines. This confirms its ability to mitigate imbalance in real time while maintaining high QoE. For pedestrian monitoring application, Hier-EI performs even better. As shown in Fig. 11(b), it exceeds baseline latency compliance by 36.12%-858.60%, with an average improvement of 212.05%. In Fig. 12(b), Hier-EI ranks just behind Chameleon in average accuracy but achieves a notable 57.25% average improvement over other methods. Given the higher pipeline complexity in this application, these results highlight the robustness of our method. Fig. 13 shows the P95 latency across over 6,000 task records per scenario. Hier-EI achieves a tightly bounded delay distribution, with an average P95 latency of 306 ms, which is 56.26% lower than the baselines, indicating effective control over tail latency and system imbalance.

Hier-EI also delivers consistent performance across all scenarios, especially in extreme cases. The scenarios become

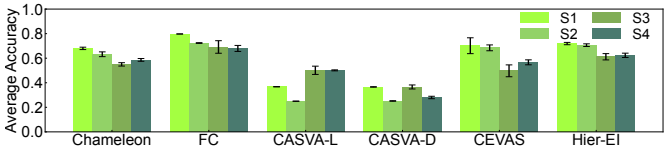


(a) Road surveillance application.

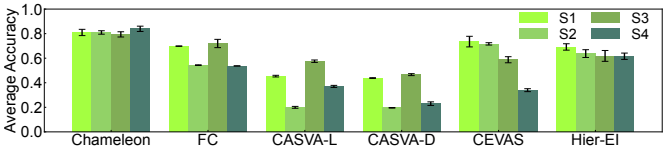


(b) Pedestrian monitoring application.

Fig. 11. Latency compliance of tasks.

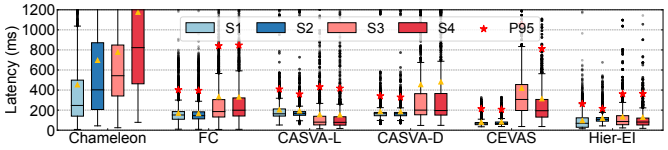


(a) Road surveillance application.

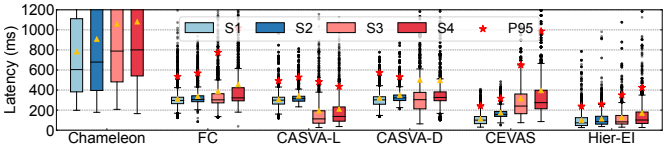


(b) Pedestrian monitoring application.

Fig. 12. Average accuracy of tasks.



(a) Road surveillance application.



(b) Pedestrian monitoring application.

Fig. 13. P95 latency in task latency distribution.

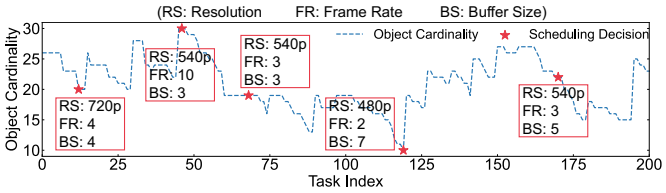


Fig. 14. Scheduling adaptive to task context.

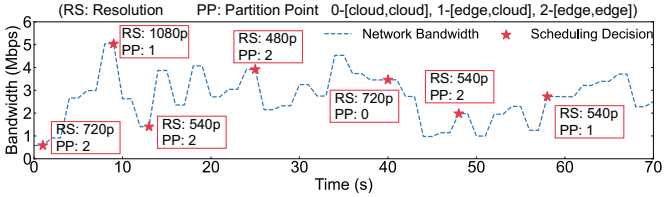


Fig. 15. Scheduling adaptive to resource context.

increasingly challenging from S1 to S4. As shown in Fig. 11, Hier-EI maintains remarkable stability, with a low coefficient of variation (CV) of 4.59% in latency compliance across all scenarios and applications. In the most challenging scenario (S4), Hier-EI improves latency compliance by 358.42% and reduces P95 latency by 67.43%, demonstrating strong adaptability and effective SLO fulfillment in VAPs.

C. Adaptive Scheduling

We evaluate the adaptability of Hier-EI by aligning scheduling decisions with dynamic task and resource conditions over time, as shown in Fig. 14–15. In Fig. 14, Hier-EI dynamically adjusts knobs in response to variations in object cardinality. For instance, it reduces frame resolution during high object density (from index 12 to 46) to lower computational load. Similarly, it decreases frame rate and increases buffer size as object motion slows (from index 68 to 119) to accelerate processing speed. These adaptations stem from macro-scheduling's convolutional layers, which extract latent patterns from runtime context.

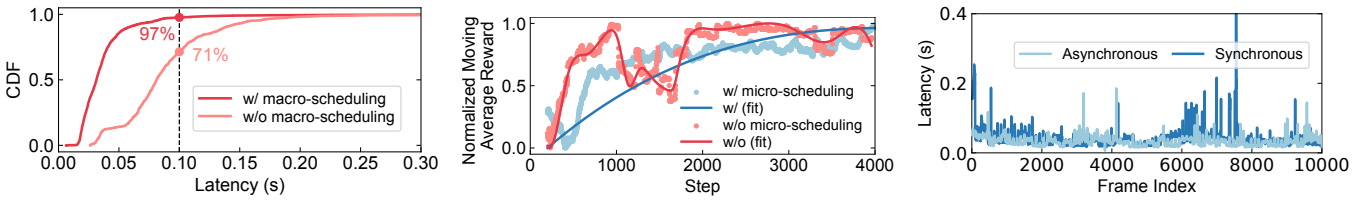
Fig. 15 demonstrates adaptability to network bandwidth. Beyond similar features displayed in Fig. 14, Hier-EI also exhibits predictive behaviors. For example, at 25th second, despite high bandwidth, it makes a conservative decision, accounting for potential instability. This foresight is enabled by the sliding window in macro-scheduling, which captures temporal context and supports trend prediction.

D. Scheduling Overhead

Scheduling methods introduce additional overhead, as shown in Tab. III. For Hier-EI, the overhead is calculated as a weighted average of macro- and micro-scheduling. We can observe that online profiling methods (e.g., Chameleon) impose considerable runtime overhead. Learning-based methods (e.g., CASVA and CEVAS) are more resource-intensive than feedback-based methods (e.g., FC), due to their broader data processing requirements. Our Hier-EI, which combines learning-based and feedback-based strategies, achieves an av-

TABLE III
THE SCHEDULING OVERHEAD OF DIFFERENT METHODS.

Methods	Scheduling Overhead	
	Road Surveillance	Pedestrian Monitoring
Chameleon	4959.63 ms ± 570.05 ms	6852.78 ms ± 806.77 ms
FC	0.50 ms ± 0.12 ms	0.49 ms ± 0.11 ms
CASVA-L	3.58 ms ± 0.72 ms	3.80 ms ± 0.64 ms
CASVA-D	3.81 ms ± 0.98 ms	3.74 ms ± 0.76 ms
CEVAS	3.09 ms ± 0.74 ms	3.18 ms ± 0.73 ms
Hier-EI (macro)	2.28 ms ± 0.80 ms	2.55 ms ± 0.64 ms
Hier-EI (micro)	0.89 ms ± 0.31 ms	0.85 ms ± 0.66 ms
Hier-EI	1.66 ms ± 0.58 ms	1.70 ms ± 0.87 ms



(a) Macro-scheduling module. (b) Micro-scheduling module. (c) Asynchronous collaboration mechanism.

Fig. 16. Ablation study results.

erage overhead of only 1 ms (second only to FC), satisfying real-time processing requirements.

E. Ablation Study

We conduct an ablation study to evaluate the effects of functional modules in Hier-EI.

Macro-scheduling module. To evaluate the contribution of macro-scheduling, we remove DRL and allow each NF method to adjust its knob independently. As shown in Fig. 16(a), this leads to a 26% drop in latency compliance at the 0.1-second threshold, demonstrating the importance of the global guidance from macro-scheduling for coordinated knob adjustment.

Micro-scheduling module. To verify the effectiveness of the micro-scheduling, we configure the DRL model to directly produce end-to-end decisions across all knobs. Fig. 16(b) shows that this approach fails to converge, as reflected in unstable normalized rewards during training. This underscores the need for micro-scheduling to manage the complexity of high-dimensional decision spaces.

Asynchronous collaboration mechanism. We compare asynchronous and synchronous modes. As shown in Fig. 16(c), the synchronous mechanism exhibits greater latency fluctuations and poorer adaptability to sudden changes, as micro-scheduling must wait for macro-level updates, reducing real-time responsiveness.

VI. Related Work

Profiling-based methods. Existing profiling-based methods involve offline and online profiling. VideoStorm [23] and VideoEdge [22] utilize offline profiling to generate resource-quality profiles and guide online resource allocation. Chameleon [19] and LLAMA [21] employ online profiling in a time window for dynamic configuration selection. AWStream [17] and ALERT [20] combine offline and online profiling, where offline profiling learns performance across conditions and online profiling updates with real-time data. However, offline profiles fail to reflect dynamic environments, and online profiling introduces additional delays. As knob number increases, search space expands exponentially, leading to unacceptable costs. Our Hier-EI employs embodied feedback mechanism to adapt dynamically and avoid profiling costs.

End-to-end learning methods. With the advent of deep learning, several methods have applied learning-based frameworks to optimize video analytics. CASVA [24] and CuttleFish [8] use DRL to tune video configurations under fluctuating network conditions. Magic-Pipe [26] employs DRL model to adjust resource allocation and microservice parameters. Similarly, Zhang *et al.* [27] dynamically adjusts video batch size, while Xu *et al.* [28] decides the number of sampled frames. However, the exponential growth of output space challenges convergence in purely end-to-end learning methods. Our Hier-EI utilizes a hierarchical collaboration mechanism to break down the output space, aiding faster convergence.

Negative feedback methods. Negative feedback offers an intuitive strategy for real-time adjustments. DDS [29] employs a feedback loop from the server to the camera to improve video analysis quality. ELF [30] allocates region proposals across edge devices based on device performance and proposal complexity. Nevertheless, NF is typically limited to adjusting a single knob. Even with independent NF methods for each knob, lack of global coordination can lead to poor performance. Our Hier-EI assigns an independent NF method to each knob in micro-scheduling and uses a unified DRL model to guide their collaborative fine-grained outputs.

VII. Conclusion

In this paper, we propose Hier-EI, an adaptive cloud-edge scheduler for VAPs that combines DRL and NF to tackle the imbalance from spatial and temporal perspectives. Through taking a hierarchical collaboration mechanism and an embodied feedback mechanism, Hier-EI reduces decision complexity and improve dynamics adaptability, maintaining long-term QoE. We also construct a scalable real-world prototype system based on KubeEdge, compatible with different scheduling methods and applications. Evaluations on our prototype system show that Hier-EI effectively improves latency compliance ratio by 3.6× and reduces P95 latency by 67.4% compared with SOTA baselines. We have provided public access to the code of Hier-EI at https://dayu-autostreamer.github.io/hier-ei.

Acknowledgment

This work is supported in part by National Key Research and Development Program of China under Grant 2022YFB3303900; National Natural Science Foundation of China under Grant No. 92467202; Key Projects of Jiangsu Provincial Basic Research Program under Grant No. BK20243040; National Natural Science Foundation of China under Grant No. 62272216, 62502196, 62441225; Basic Research Program of Jiangsu under Grant No. BK20251207; Postdoctoral Fellowship Program of CPSF under Grant No. GZB20250389; China Postdoctoral Science Foundation under Grant No. 2025M771535; and Collaborative Innovation Center of Novel Software Technology and Industrialization. Lei Xie is the corresponding author.

REFERENCES

[1] S. Afzal, S. Ghani, M. M. Hittawe, S. F. Rashid, O. M. Knio, M. Hadwiger, and I. Hoteit, "Visualization and visual analytics approaches for image and video datasets: A survey," *ACM Transactions on Interactive Intelligent Systems*, vol. 13, no. 1, pp. 1–41, 2023.

[2] Y. Kim, C. Oh, J. Hwang, W. Kim, S. Oh, Y. Lee, H. Sharma, A. Yazdanbakhsh, and J. Park, "Dacapo: Accelerating continuous learning in autonomous systems for video analytics," in *Proc. of the 51st ACM/IEEE ISCA*, 2024, pp. 1246–1261.

[3] C. Hu, R. Lu, and D. Wang, "Feva: A federated video analytics architecture for networked smart cameras," *IEEE Network*, vol. 35, no. 6, pp. 163–170, 2022.

[4] S. Wang, S. Yang, and C. Zhao, "Surveiledge: Real-time video query based on collaborative cloud-edge deep learning," in *Proc. of the 39th IEEE INFOCOM*, 2020, pp. 2519–2528.

[5] G. Grassi, K. Jamieson, P. Bahl, and G. Pau, "Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments," in *Proc. of the 2nd ACM SEC*, 2017, pp. 1–14.

[6] G. H. Apostolo, P. Bauszat, V. Nigade, H. E. Bal, and L. Wang, "Live video analytics as a service," in *Proc. of the 2nd EuroMLSys*, 2022, pp. 37–44.

[7] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *Proc. of the 37th IEEE INFOCOM*, 2018, pp. 1421–1429.

[8] N. Chen, S. Quan, S. Zhang, Z. Qian, Y. Jin, J. Wu, W. Li, and S. Lu, "Cuttlefish: Neural configuration adaptation for video analysis in live augmented reality," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 830–841, 2020.

[9] S. Paul, K. Rao, G. Coviello, M. Sankaradas, Y. C. Hu, and S. T. Chakradhar, "Camtuner: Adaptive video analytics pipelines via real-time automated camera parameter tuning," *IEEE Transactions on Mobile Computing*, 2025.

[10] Y. Lu, S. Jiang, T. Cao, and Y. Shu, "Turbo: Opportunistic enhancement for edge video analytics," in *Proc. of the 20th ACM SenSys*, 2022, pp. 263–276.

[11] L. Han, Z. Zhou, and Z. Li, "Pantheon: Preemptible multi-dnn inference on mobile edge gpus," in *Proc. of the 22nd ACM MobiSys*, 2024, pp. 465–478.

[12] D. Xu, A. Zhou, G. Wang, H. Zhang, X. Li, J. Pei, and H. Ma, "Tutti: coupling 5g ran and mobile edge computing for latency-critical video analytics," in *Proc. of the 28th ACM MobiCom*, 2022, pp. 729–742.

[13] G. Zhang, J. Zhang, Y. Liu, H. Hu, J. Y. Lee, and V. Aggarwal, "Adaptive video streaming with automatic quality-of-experience optimization," *IEEE Transactions on Mobile Computing*, vol. 22, no. 8, pp. 4456–4470, 2022.

[14] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 339–350, 2013.

[15] J. Wang and M. Balazinska, "Deluceva: Delta-based neural network inference for fast video analytics," in *Proc. of the 32nd SSDBM*, 2020, pp. 1–12.

[16] T. Tan and G. Cao, "Deep learning on mobile devices through neural processing units and edge computing," in *Proc. of the 41st IEEE INFOCOM*, 2022, pp. 1209–1218.

[17] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *Proc. of the 32nd ACM SIGCOMM*, 2018, pp. 236–252.

[18] M. Salehe, Z. Hu, S. H. Mortazavi, I. Mohomed, and T. Capes, "Videopipe: Building video stream processing pipelines at the edge," in *Proc. of the 20th ACM Middleware*, 2019, pp. 43–49.

[19] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proc. of the 32nd ACM SIGCOMM*, 2018, pp. 253–266.

[20] C. Wan, M. Santriaji, E. Rogers, H. Hoffmann, M. Maire, and S. Lu, "{ALERT}: Accurate learning for energy and timeliness," in *Proc. of the 31st USENIX ATC*, 2020, pp. 353–369.

[21] F. Romero, M. Zhao, N. J. Yadwadkar, and C. Kozyrakis, "Llama: A heterogeneous & serverless framework for auto-tuning video analytics pipelines," in *Proc. of the 12th ACM SoCC*, 2021, pp. 1–17.

[22] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, "Videoedge: Processing camera streams using hierarchical clusters," in *Proc. of the 3rd ACM SEC*, 2018, pp. 115–131.

[23] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *Proc. of the 14th USENIX NSDI*, 2017, pp. 377–392.

[24] M. Zhang, F. Wang, and J. Liu, "Casva: Configuration-adaptive streaming for live video analytics," in *Proc. of the 41st IEEE INFOCOM*, 2022, pp. 2168–2177.

[25] R. Zhang, Y. Zhou, F. Wang, and Z. Wang, "Maxim: Drl-based cross-camera streaming configuration for real-time video analytics," in *Proc. of the 31st IEEE ICME*, 2022, pp. 01–06.

[26] G. Coviello, Y. Yang, K. Rao, and S. Chakradhar, "Magic-pipe: Self-optimizing video analytics pipelines," in *Proc. of the 22nd International Middleware Conference*, 2021, pp. 79–90.

[27] L. Zhang, Y. Zhang, X. Wu, F. Wang, L. Cui, Z. Wang, and J. Liu, "Batch adaptative streaming for video analytics," in *Proc. of the 41st IEEE INFOCOM*, 2022, pp. 2158–2167.

[28] M. Xu, X. Zhang, Y. Liu, G. Huang, X. Liu, and F. X. Lin, "Approximate query service on autonomous iot cameras," in *Proc. of the 18th ACM MobiSys*, 2020, pp. 191–205.

[29] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *Proc. of the 34th ACM SIGCOMM*, 2020, pp. 557–570.

[30] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang, "Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading," in *Proc. of the 27th ACM MobiCom*, 2021, pp. 201–214.

[31] V. Nigade, R. Winder, H. Bal, and L. Wang, "Better never than late: Timely edge video analytics over the air," in *Proc. of the 19th ACM SenSys*, 2021, pp. 426–432.

[32] D. Howard, A. E. Eiben, D. F. Kennedy, J.-B. Mouret, P. Valencia, and D. Winkler, "Evolving embodied intelligence from materials to machines," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 12–19, 2019.

[33] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: A 4g lte dataset with channel and context metrics," in *Proc. of the 9th ACM MMSys*, 2018, pp. 460–465.

[34] "NVIDIA TensorRT," https://developer.nvidia.com/tensorrt, 2024.

[35] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the 7th IJCAI*, vol. 2, 1981, pp. 674–679.

[36] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[37] M. Vojnovic, J.-Y. Le Boudec, and C. Boutremans, "Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times," in *Proc. of the 19th IEEE INFOCOM*, vol. 3, 2000, pp. 1303–1312.

[38] "KubeEdge," https://kubeedge.io/, 2024.

[39] "Sedna," https://sedna.readthedocs.io/, 2021.

[40] "EdgeMesh," https://edgemesh.netlify.app/, 2021.

[41] "Docker," https://www.docker.com/, 2024.

[42] "NVIDIA Jetson TX2," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/, 2024.

[43] "NVIDIA Jetson AGX Orin," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/, 2024.

[44] "MikroTik RB5009UG+S+IN," https://mikrotik.com/product/rb5009ug_s_in, 2024.

[45] "MikroTik RouterOS," https://help.mikrotik.com/docs/spaces/ROS/pages/328059/RouterOS, 2024.

[46] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. Rondao Alface, T. Bostoen, and F. De Turck, "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[47] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "Ua-detrac: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, vol. 193, p. 102907, 2020.

[48] "YouTube," https://youtube.com, 2024.

[49] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. of the 37th ICML*, 2018, pp. 1861–1870.

[50] M. Zhang, F. Wang, Y. Zhu, J. Liu, and Z. Wang, "Towards cloud-edge collaborative online video analytics with fine-grained serverless pipelines," in *Proc. of the 12th ACM MM*, 2021, pp. 80–93.