Edge-assisted Video Object Detection with Adaptive Resource Allocation and Model Selection

Xuejian Chi, Xiaoxi Zhang, *Member, IEEE*, Yifei Zou, *Member, IEEE*, Xingze Wu, Zhipeng Cai, *Fellow, IEEE*, Falko Dressler, *Fellow, IEEE*, and Dongxiao Yu, *Senior Member, IEEE*

***Abstract*—With the rise of applications such as facial recognition and intelligent monitoring, demand for video object detection has surged. To optimize detection performance, existing works primarily focus on either video transmission or processing independently, often overlooking small object detection, which is particularly difficult for resource-limited terminals. To address these limitations, we propose a solution that leverages edge servers to assist video object detection, enabling real-time and high-precision performance under resource constraints. We introduce *Adaptive-EVOD*, the first edge-assisted video detection system that optimizes communication and computation simultaneously through adaptive resolution and frame rate adjustment, resource allocation, and model selection strategies, with a particular emphasis on small object detection. Specifically, *Adaptive-EVOD* introduces three core innovations: (1) proposing a detection strategy for small objects to enhance precision; (2) employing deep reinforcement learning to dynamically determine detection location, resolution, and frame rate based on heterogeneous environments; and (3) applying Bayesian optimization to dynamically allocate GPU resources and select Low-Rank Adaptation (LoRA) ranks for detection models adaptively. Experiments on a real-world testbed show that *Adaptive-EVOD* reduces end-to-end latency by 5.7%–18.3% and improves detection accuracy by 7.8%–23.4% compared to state-of-the-art methods, which highlights the effectiveness of *Adaptive-EVOD*, particularly in complex and resource-constrained edge scenarios.**

***Index Terms*—Video object detection, edge computing, joint optimization, deep reinforcement learning.**

I. Introduction

As the integration of smart internet of things, multimedia systems and edge computing technique, edge-assisted video object detection has garnered increasing attention from researchers recently [1]. By processing videos on both terminal and edge servers, this approach achieves superior performance compared to local detection methods only on terminal device, because the rich computational and storage resources at the edge enable the deployment of high-precision deep neural network (DNN) models with vast parameters, e.g., YOLOv5x with 86M parameters. In contrast to methods reliant on remote cloud centers, edge servers, being in closer proximity to the data source, can significantly diminish time costs and privacy leakage risks associated with video transmission [2], [3]. Consequently, edge-assisted video object detection is pivotal across various domains, particularly for time/privacy-sensitive applications such as intelligent surveillance [4], autonomous driving [5], and object tracking [6].

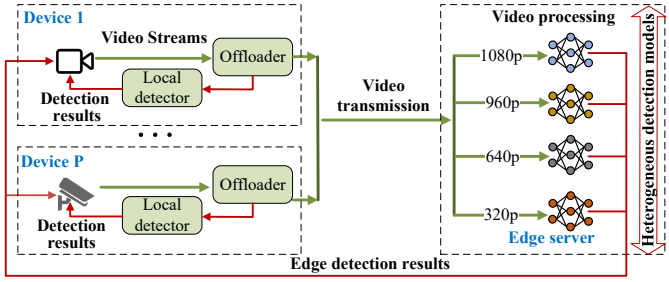


Fig. 1: Edge-assisted video object detection architecture.

Due to its significance, multiple studies have been conducted on this research topic. Yet, there are still with two main concerns. The first concern is that existing studies primarily focus on either video transmission or video processing as isolated stages, lacking a comprehensive consideration and co-design on both. For example, the strategies in [7]–[10] reduce transmission latency through bitrate adaptation, while others employ techniques such as super-resolution enhancement and resource allocation during the video processing stage to improve video detection quality [11]–[13]. However, video object detection on the edge inherently involves both transmission and processing stages, where key decision factors, such as resolution and frame rate, simultaneously affect both stages. Therefore, improving only one stage is insufficient to achieve optimal overall system performance, highlighting the necessity of jointly optimizing both stages. The second concern lies in the limited capability of existing methods to enhance the detection accuracy of small objects in video streams. As video frames with a high proportion of small objects are common in practical applications [13], [14], this limitation results in an approximate 44% degradation in overall detection accuracy reported by [15].

To address the above issues, this paper proposes *Adaptive-EVOD*, an **E**dge-assisted **V**ideo **O**bject **D**etection system with **Adaptive** resolution and frame rate adjustment, resource allo-

Xuejian Chi, Yifei Zou, and Xingze Wu are with the School of Computer Science and Technology, Shandong University, and also with Shandong Provincial Key Laboratory of Computing-Network Integration (Shandong University), Qingdao, Shandong 266237, China,. E-mail: {chixuejian, wuxingze}@mail.sdu.edu.cn, yfzou@sdu.edu.cn

Dongxiao Yu is with the School of Cryptography Science and Engineering, Shandong University, Jinan, Shandong 250101, China. E-mail: dxyu@sdu.edu.cn

Xiaoxi Zhang is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, 510006, China. E-mail: zhangxx89@mail.sysu.edu.cn

Zhipeng Cai is with Department of Computer Science, Georgia State University, Atlanta, 30302, USA. E-mail: zcai@gsu.edu

Falko Dressler is with the School of Electrical Engineering and Computer Science, TU Berlin, Berlin, 10587, Germany. E-mail: dressler@ccs-labs.org

(Corresponding author: Xiaoxi Zhang.)

cation, and model selection, which performs joint optimization across the transmission and processing stages, with special consideration for small object detection. As illustrated in Fig. 1, we consider an edge network consisting of multiple heterogeneous terminal devices and an edge-based server. Terminal devices capture various types of videos. Heterogeneous machine learning models are deployed on the terminal devices and edge server for video object detection. The edge server performs centralized decision making to determine whether each video frame should be processed locally or uploaded to the edge server for detection. If video frames are uploaded to the edge server, the system can strategically select their resolutions, detection models, and resource amounts allocated for the detection tasks performed on the edge, achieving the joint optimization over communication and computation. The edge server will then return the results to the terminal devices once their detection tasks are completed. Intuitively, this joint optimization balances the trade-offs between latency and accuracy across video transmission and processing stages: a higher resolution in the processing stage, which typically involves using DNN model inferences, can improve detection accuracy but incurs greater latency in the transmission stage; while detection models with smaller parameters reduce computational latency at the cost of decreased accuracy. Moreover, due to the dynamic variations in bandwidth and other resource conditions in mobile edge computing scenarios, resource configurations have uncertain impacts on latency and accuracy. The edge server should learn the relationship between resource configuration and system performance, e.g., by leveraging historical configuration and performance data. Based on this relationship, the system can periodically adjust resource and model configurations to achieve optimal detection performance under time-varying conditions such as bandwidth and model workload. This intuition drives the design of *Adaptive-EVOD* for real-time, high-precision video object detection, thereby enhancing overall Quality of Service (QoS)*.

To realize Adaptive-EVOD, the first challenge is to achieve adaptive joint optimization across video transmission and processing stages. As demonstrated in Section II, factors such as resolution and frame rate in the video transmission stage, along with model selection and resource allocation in the video processing stage, simultaneously impact the overall accuracy-delay trade-off. Besides, the balance point varies under different bandwidths and video types. Traditional methods, such as combinatorial optimization or linear programming, struggle to grasp the intricate and interdependent relationships among these variables, to find the optimal solution, particularly in heterogeneous and dynamic environments. In *Adaptive-EVOD*, we employ the Advantage Actor-Critic (A2C) algorithm to train a Deep Reinforcement Learning agent, named DRL-A, to adaptively optimize detection decisions for each video frame captured by the terminal devices, including selecting the processing location (terminal or edge), resolution, and detection model. Moreover, at each time epoch, an online Bayesian optimization algorithm is employed to select the optimal GPU type and the Low-Rank Adaptation (LoRA) matrix rank for each deployed detection model, according to the current and historical information of the edge server. For example, if a detection model receives a large number of video frames in a short period, we should adaptively assign stronger GPUs to the high-load model and lowering its LoRA rank to reduce the time-cost of video detection on each frame. By integrating these two optimization strategies, *Adaptive-EVOD* learns the optimal solution in a complex decision space.

The second challenge of Adaptive-EVOD arises from bandwidth limitations. Transmitting all frames containing small objects to the edge server for detection introduces significant communication delays [16], [17], thereby adversely affecting real-time performance. This necessitates a trade-off between transmission efficiency and detection accuracy when deciding whether to offload a video frame containing small objects to the edge server for high-precision detection. To address this, we propose a small object detection strategy (see Section III-A) that determines the detection thresholds for different types of small objects and incorporates them as part of the state space in our DRL-A module. This strategy serves as a feature extractor that enables DRL-A to adaptively optimize detection decisions for small objects, thereby improving the performance of *Adaptive-EVOD* on real-time small object detection.

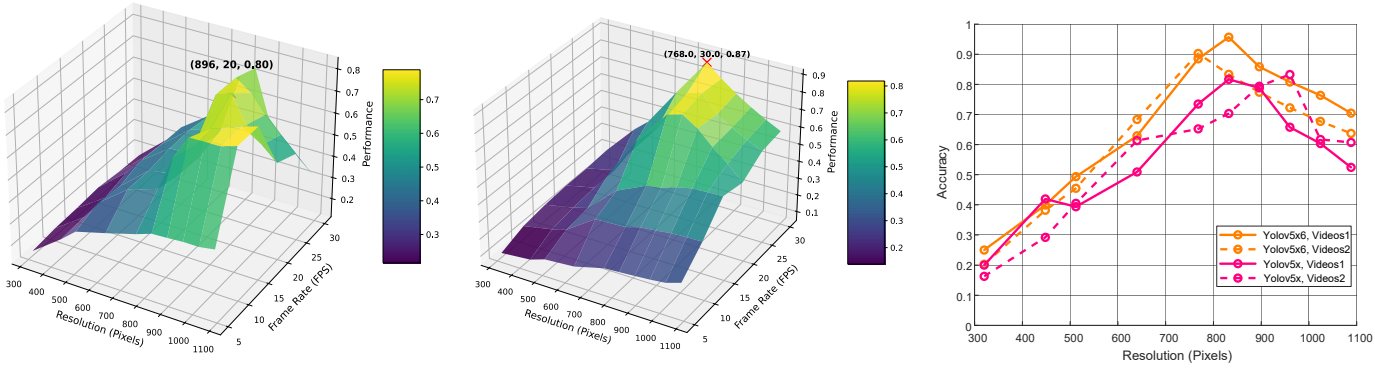
The key contributions of *Adaptive-EVOD* are as follows:

1) We leverage the A2C algorithm to train a DRL agent capable of capturing the complex non-linear dependencies between dynamic environmental factors and the system's objective function. This facilitates the adaptive and joint optimization of detection placement, resolution, frame rate, and model selection for heterogeneous video streams. Such fine-grained, frame-level collaborative decision-making significantly enhances the system's global QoS.
2) An online Bayesian optimization algorithm is designed to determine the optimal GPU type and LoRA rank for each deployed detection model through historical information, dynamically optimizing resource allocation on edge server at each time epoch. This enables the efficient and dynamic adaptation of computational resources to varying video types and workload demands, thereby improving overall system performance.
3) A small object detection strategy is proposed to determine detection thresholds for different types of small objects. Based on these thresholds, the number of small objects in the current frame can be estimated, and frames with a high proportion of small objects are transmitted to the edge for high-precision detection.

Extensive real-world data-driven experiments are conducted with necessary ablation studies and comparisons to state-of-the-art methods, in which *Adaptive-EVOD* reduces end-to-end latency by 5.7%-18.3% and improves detection accuracy by 7.8%-23.4%.

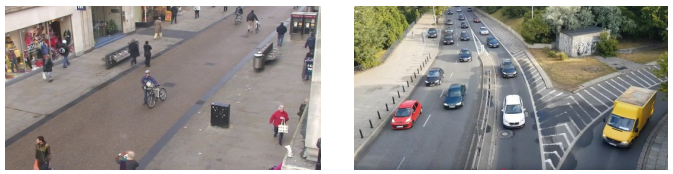
The remainder of this paper is organized as follows. Section II presents the motivating experiments. Section III introduces the system model and main modules. Section IV formulates the optimization problem. Section V presents the proposed joint optimization strategy based on deep reinforce-

*QoS: a measure of performance including latency and detection accuracy in edge-assisted video object detection systems.



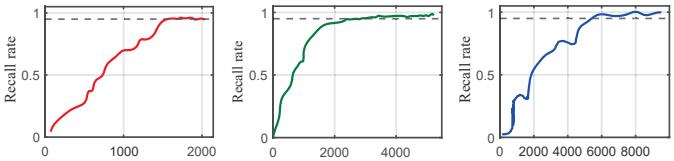
(a) The detection results of video 1 (b) The detection results of video 2 (c) Detection results of different models

Fig. 3: The detection results for different types and configurations of videos using different models.



(a) Slow-moving pedestrians (b) Fast-moving vehicles

Fig. 2: An example of different video types.



(a) Person size (b) Motorcycle size (c) Car size

Fig. 4: Detection thresholds for different objects types.

ment learning and Bayesian optimization. Section VI evaluates the system performance. Section VII reviews related work, and Section IX concludes the paper.

II. Empirical Observations and Motivation

In this section, we present three key experimental observations that directly motivate our work. Observation 1: The relationships between decision variables (such as video type, resolution, frame rate, and detection model) and detection performance are complex and highly nonlinear. Observation 2: The transmission and processing stages are tightly coupled; optimizing them separately can lead to suboptimal results. Observation 3: The object size required to achieve a 95% recall rate varies across different object types.

Setup. We employ YOLOv5 model (YOLOv5x) to perform object detection on two distinct types of videos. As illustrated in Fig. 2, the first type captures pedestrians moving at low speeds [18], while the second captures high-speed moving vehicles [19]. To evaluate system performance, a weighted sum of accuracy and latency (as detailed in Section IV) is used as the evaluation metric, with video frame resolution, frame rate, and other relevant parameters as decision variables.

Observation 1. Fig. 3 presents the detection results using different models on various types of videos. As shown in Fig. 3(a) and (b), using the YOLOv5x model, detecting on videos 1 and 2 achieve their best accuracy-delay trade-off at $(896p, 20fps)$ and $(768p, 30fps)$, respectively. This discrepancy arises from the fact that objects in video 1 are relatively smaller in scale compared to those in video 2, making resolution a more critical factor in detection performance. Additionally, due to the slower movement of objects in video 1, frame rate has a less pronounced impact on detection performance than in video 2. Fig. 3(a) and (b) demonstrates that the impact of decision variables (e.g., video type, resolution, frame rate, and detection model) on detection performance is complex and nonlinear. Therefore, traditional methods struggle to adaptively determine the optimal configuration for video processing, **highlighting the necessity of intelligent and adaptive decision making.**

Observation 2. In Fig. 3(c), the setup of processing stage is fixed to YOLOv5x (with 86M parameters), a high-accuracy, high-latency model, and only the transmission is optimized. The results show that under heavy system load, although strategies like resolution optimization improve system performance, the gain is offset by the long inference time of high-accuracy models. This confirms that improvements in one stage can be negated by limitations in the other. Therefore, optimizing them separately leads to suboptimal outcomes, **making joint optimization essential.**

Observation 3. To enhance the detection accuracy of small objects, it is necessary to establish detection thresholds that determine whether a given frame should be transmitted to the edge server for further processing. Fig. 4 illustrates the detection thresholds for different types of small objects under the same detection model. It can be observed that the pixel threshold required for accurate detection (defined as a recall rate exceeding 95%) varies across object types. For instance, pedestrian objects with a pixel size of 1800 per frame achieve nearly 100% recall, whereas vehicles of the same pixel size exhibit a recall rate of less than 35%. Therefore, setting object-specific detection thresholds **enables adaptive selection of the detection location**, thereby improving the accuracy of small object detection.

III. System Design

We propose an edge-assisted video analysis system consisting of multiple terminal devices $D = \{D_1, D_2, \ldots, D_P\}$ and an edge server E. The system allows that the object detection task for each frame of video can be executed on either the terminal device or the edge server. Intuitively, the limited computational resources of terminal devices restrict them to lightweight object detection models, which often perform poorly on complex videos containing small objects. In contrast, the edge server's greater computational capabilities allows it to employ larger object detection models, thereby improving detection accuracy. However, transmitting video frames to the edge server inevitably introduces additional transmission latency. Therefore, to enhance detection accuracy and reduce end-to-end latency, the system selects the resolution and processing location for each frame, and periodically adjusts the model type and resources for object detection. TABLE I presents the key symbols that will be used throughout this paper.

TABLE I: Table of Main Notations.

Notation	Description
$D = \{D_1, \ldots, D_P\}$	The set of terminal devices
E	The edge server
χ_t	The set of video frames in epoch t
f	The video frame index
θ_f, Δ_f	Number of small objects and frame difference of f
ϖ_t	Available network bandwidth at epoch t
q_t	Queuing status/lengths of detection models at epoch t
$X_{f,t}$	Offloading decision (1 if to edge, 0 if local)
R_f	Resolution level of video frame f
m_f	Detection model selected for frame f
α	Weight factor for latency-accuracy trade-off
$L_{f,t}^{R_f}$	End-to-end latency of frame f at resolution R_f
$IoU_{f,t}$	Detection accuracy (IoU) of frame f at epoch t
A_t	Average object detection accuracy at epoch t
L_t	Total system latency at epoch t
ϕ_t, g_t	LoRA rank and GPU allocation for detection models
z_t	Action-context pair (a_t, c_t) for Bayesian optimization
y_t	Observed reward at epoch t
$k_z(z, z')$	Squared exponential kernel function
σ^2	Observation noise variance of the reward function
β_t	UCB acquisition function exploration parameter

Fig. 5 shows an overview of our edge-assisted video analysis system. ① The terminal devices process the captured video frames through a **feature extractor** to generate feature data of small object judgment for each frame, containing the number of small objects and the frame difference compared to the previous frame. These feature data are then transmitted to a **request scheduler** on the edge server, where a pre-trained DRL-A model dynamically determines the location and resolution decisions (a.k.a. actions) based on the system's state. ② Then, an **offloading controller** on the terminal device extracts the corresponding video frames based on received decisions and forwards them to the **resolution adjuster** for resolution optimization. ③ The video frames are then encoded using the High Efficiency Video Coding (HEVC, a.k.a. H.265) codec and transmitted to the edge server for decoding. ④ The **object detector** on the edge server performs object detection on the decoded video frames and transmits the system state information to the request scheduler to guide

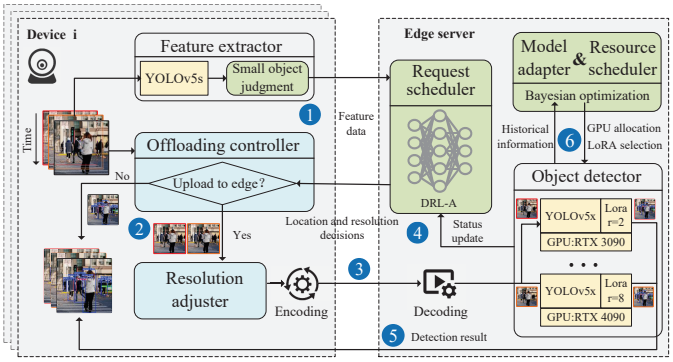


Fig. 5: The *Adaptive-EVOD* system overview.

subsequent decision making processes. ⑤ Simultaneously, the edge server transmits the detection results back to the terminal device, where they are integrated with locally detected video frames to complete the object detection for all video frames. ⑥ Finally, the **model adapter** and **resource scheduler** update the GPU resource allocation and the rank parameters of LoRA for each object detection service in the next time epoch.

A. Feature Extractor

The system performs object detection over a series of epochs $\mathcal{T} = \{1, \ldots, T\}$, with each epoch having a fixed duration, e.g., 1 second. We define χ_t as the set of frames within each epoch. The frames in χ_t are then input into the feature extractor, where frame differences are computed using ffmpeg, and the number of small objects in each frame is obtained using a lightweight object detection model.

The detection of small objects is defined as follows: An object is considered a small object if (1) the feature extractor on the terminal device fails to classify the object into any category (or reports low confidence scores), and (2) the object detector on the edge server classifies the object into a specific category (or reports high confidence scores). In practice, we employ a category-specific size threshold δ_k to determine whether an object of type k is considered small. Formally, δ_k is defined as the minimum bounding box pixel area at which the local lightweight model (e.g., YOLOv5s) can reliably achieve a recall rate above a predefined threshold, e.g., 95% in our experiments. This threshold is empirically tuned using a validation dataset with high-quality ground-truth labels established by a large-scale model and manual verification. Since different object types exhibit distinct visual complexities and feature densities, these adaptive thresholds avoid a one-size-fits-all heuristic, with distinct optimal thresholds for different object types as derived from the method in Observation 3. If the object size is smaller than δ_k, the lightweight detector may fail to detect it reliably, and the corresponding video frame is more likely to be transmitted to the edge server at a high resolution for detection. Otherwise, the object is detected by the lightweight model on the terminal device. Let $X_{f,t} = 1$ denote that frame f is transmitted at high resolution to the edge server in epoch t; otherwise, $X_{f,t} = 0$.

The feature extractor retains the detection results of frames that do not require upload to the edge server and uses them as the final object detection results for those frames.

B. Offloading Controller & Resolution Adjuster

Upon receiving the decision actions, the offloading controller extracts the video frames that need to be uploaded. These selected frames are then forwarded to the resolution adjuster for resolution optimization. Preliminary experiments indicate that, under limited bandwidth resources, increasing the resolution of video frames can enhance the accuracy of small object detection but also incurs higher transmission latency. Therefore, selecting an appropriate resolution level enables a balanced trade-off between latency and accuracy. We define R_f as the resolution level of video frame f.

C. Request Scheduler

The request scheduler is one of the core components of the entire system. In the request scheduler, a DRL-A is trained to receive feature data from the feature extractor, including the number of objects per frame, video type, and frame differences. By incorporating edge server information (such as available bandwidth, detection model performance, and current workload), the DRL-A jointly optimizes the video transmission and object detection to determine the optimal offloading decision for each video frame. For example, when bandwidth is sufficient, frames containing a large number of small objects and large inter-frame differences should be offloaded to the edge server and processed using the detection model with the highest LoRA rank, in order to maximize detection accuracy.

D. Object Detector

As shown in Fig. 5, the object detector is equipped with N detection models $\mathcal{M} = \{m_1, m_2, ..., m_N\}$. Due to differences in GPU allocation and LoRA ranks across these models, their processing capabilities vary. Models with higher GPU allocations achieve faster processing speeds, while higher LoRA ranks improve detection accuracy at the cost of reduced processing speed. The request scheduler dynamically adapts the selection of detection models for subsequent frames based on the waiting queues of each model. For example, if the queue for model m_1 becomes overloaded, the request scheduler can guide subsequent frames to be offloaded to other idle models.

E. Model Adapter & Resource Scheduler

The model adapter and resource scheduler constitute another core component of the system. The model adapter determines the LoRA rank adopted by each detection model. LoRA fine-tuning introduces low-rank adapter layers into the object detection models, enabling efficient fine-tuning with minimal parameter adjustments and resource consumption, while preserving detection performance. It is worth noting that, unlike the standard LoRA implementation where adapter weights are merged into the backbone ($W' = W_0 + BA$) to eliminate inference latency, *Adaptive-EVOD* adopts an unmerged deployment strategy. This design is essential for our edge-assisted system to support multiple devices, allowing real-time switching of fine-tuned models across different video streams without reloading the massive backbone weights. In this setting, the inference process is formulated as $y = W_0x + B(Ax)$, where the frozen backbone W_0 and the adapter branch BA are computed separately. Since the computational complexity of the adapter branch is linearly proportional to the rank r, the LoRA rank effectively serves as a tunable knob for resource management. By dynamically adjusting r, the system can explicitly trade off minor computational overhead for higher accuracy (high r) or minimize latency under heavy loads (low r), thereby addressing the limitation of fixed inference costs in merged models.

The resource scheduler dynamically allocates a certain number of GPU devices to each model, enhancing the overall detection throughput. Unlike the request scheduler, which makes per-frame decisions, the model adapter and resource scheduler operate on a per-epoch basis. To achieve optimal configurations of LoRA ranks and GPU allocations for each epoch, we design a Bayesian optimization algorithm to obtain the best solution with a limited number of sampling points.

IV. Problem Formulation

To evaluate our system, we consider two metrics: latency and accuracy. The formal definitions of these metrics are provided below.

A. System Delay

The latency of each frame consists of two components: data transmission latency and object detection latency. If video frame f is processed locally on the terminal device ($X_{f,t} = 0$), no data transmission latency is incurred, and the processing latency on the lightweight object detection model is denoted by $d_{f,t}$. If video frame f is uploaded to the edge server in epoch t ($X_{f,t} = 1$) with resolution R_f, the transmission latency can then be computed as:

$$D_{f,t}^{R_f} = S_{f,t}^{R_f} / B_t, \tag{1}$$

where $S_{f,t}^{r}$ denotes the data size of video frame f at resolution R_f, and B_t represents the network bandwidth of the system in epoch t. Given the presence of multiple object detection models on the edge server, we introduce $I_{f,t}^{R_f,m_f}$ to represent the object detection latency of video frame f with resolution R_f on detection model m_f. Therefore, the latency for each frame can be expressed as:

$$L_{f,t}^{R_f} = (1 - X_{f,t}) \times d_{f,t} + X_{f,t} \times (D_{f,t}^{R_f} + I_{f,t}^{R_f,m_f}). \tag{2}$$

The total latency for all frames in epoch t can be expressed as the sum of the latency for each individual frame, i.e.

$$L_t = \sum\nolimits_{f \in \chi_t} L_{f,t}^{R_f}. \tag{3}$$

B. Detection Accuracy

We use the Intersection over Union (IoU) metric to measure the object detection accuracy of video frame f [13]. This metric is determined by the overlap area between the predicted bounding box and the ground-truth bounding box, defined as:

$$IoU_o = \frac{|BBox(o) \cap BBox(g)|}{|BBox(o) \cup BBox(g)|}, \tag{4}$$

where $BBox(o)$ represents the bounding box of object o obtained by the system, and $BBox(g)$ denotes the corresponding ground-truth bounding box. It is worth noting that video frame f contains multiple objects, therefore, the detection accuracy $IoU_{f,t}$ of frame f is defined as the average IoU of all objects in f. We define the object detection accuracy of the system as the average detection accuracy of all frames in epoch t, expressed as:

$$A_t = \sum\nolimits_{f \in \chi_t} IoU_{f,t}/K, \tag{5}$$

where K represents the number of video frames in epoch t.

C. Optimization Problem

The analysis of real-time video is latency-sensitive and typically requires high-quality results. Therefore, our objective is to achieve an optimal trade-off between latency and accuracy, i.e., the maximization of the weighted sum of the total system accuracy and the negative total latency, formulated as:

$$\max_{X_{f,t}, R_f, m_i} \quad \sum\nolimits_{t=1}^{T} \alpha A_t - (1-\alpha) L_t \tag{6}$$

$$s.t. \quad X_{f,t} \in \{0,1\}, \ \forall f \in \chi_t, t \in \mathcal{T}, \tag{7}$$

$$L_{f,t}^{R_f} \le \mathcal{L}_{f,t}, \ \forall f \in \chi_t, t \in \mathcal{T}, \tag{8}$$

$$IoU_{f,t} \ge O_{f,t}, \ \forall f \in \chi_t, t \in \mathcal{T}, \tag{9}$$

where weight factor α represents the latency accuracy trade-off, e.g., a larger α shows that the system prioritizes higher accuracy. In practical terms, system administrators determine α based on the specific latency-criticality of their task. Constraint (7) indicates each frame f should be processed by terminal device or edge server. Constraint (8) ensures that the latency of video frame f should not exceed the maximum delay limit $\mathcal{L}_{f,t}$. Constraint (9) requires that the accuracy of frame f cannot be lower than the minimum accuracy limit $O_{f,t}$. To solve this problem with combinatorial optimization or linear programming, the functional relationship between decision variables (such as resolution, model resources, etc.) and performance should be known in advance. However, as illustrated in our motivation section, this functional relationship is complex and various for different types of videos. In addition, edge environment such as network bandwidth and processing latency for each frame cannot be obtained in advance, making it hard to find the optimal solution. In the following section, we introduce *Adaptive-EVOD* that combines deep reinforcement learning and Bayesian optimization, to learn the intrinsic relationship between decision variables and performance indicators based on the system's operating results, and obtains the optimal solution.

V. Algorithm Design for *Adaptive-EVOD*

In this section, we first introduce DRL-A, which makes fine-grained decisions,the detection location, resolution, and model selection, for each frame, running within the request scheduler. Next, we detail our Bayesian optimization algorithm that adjusts model size and resources between epochs, running within the model adapter and resource scheduler. Together, these two algorithms optimize system performance in a bi-level manner, effectively improving the system's QoS. The pseudocode is shown in Algorithm 1.

Algorithm 1 Edge-assisted Video Analysis Algorithm

Input: Deep reinforcement learning's state space $\mathcal{S}$ and action space $\mathcal{A}$, Bayesian optimization's action space $\mathcal{D}$, historical reward set $M_0 = \emptyset$, weight factor α, GP Prior μ_0, σ_0.

for $t = 1, 2, \ldots, T$ **do**
  $a_t = \underset{a_t \in \mathcal{D}}{argmax}\ \mu_{t-1}(z_t) + \beta_t^{\frac{1}{2}} \sigma_{t-1}(z_t)$;
  Adjust the rank of LoRA and GPU allocation for each detection model based on a_{t+1};
  for $f \in \chi_t$ **do**
    DRL-A determines frame f's action (X_f, R_f, m_f) ;
    if $X_f = 1$ **then**
      Perform object detection on frame f at the edge server;
    else
      Perform object detection on frame f at the terminal;
    end if
  end for
  Compute reward y_t based on (6);
  Update $M_t = \{M_{t-1}, (z_t, y_t)\}$;
  for $a_t \in \mathcal{A}$ **do**
    Observe context c_t;
    Update μ_t, k_t, σ_t based on (14);
  end for
end for

A. Deep Reinforcement Learning Agent

We formulate the single-frame decision problem as a Markov Decision Process (MDP) [20], represented by a four-tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$. $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $\mathcal{P}$ denotes the state transition probability, and $\mathcal{R}$ denotes the reward function. We define the parameters of the MDP as follows:

1) State: The system state consists of terminal-side features and edge-side metrics. The terminal-side features include the number of objects θ and the inter-frame pixel difference Δ, which reflect the visual complexity and temporal dynamics of the current video stream. The edge-side metrics include the available bandwidth ϖ and the queue lengths of the deployed detection models q. Specifically, the terminal device utilizes a lightweight detection model to extract θ and Δ, while the edge server monitors the current bandwidth and model queuing status. These components collectively form the system state $S = (\theta_1, \Delta_1, \theta_2, \Delta_2, \ldots, \theta_P, \Delta_P, \varpi, q_1, q_2, \ldots, q_N)$.

2) Action: The action for each frame consists of three components: detection location ω, resolution R_f, and model selection m_f, denoted as $\mathcal{A} = (\omega, R_f, m_f)$. For example, an action $\mathcal{A} = (0, 0, 0)$ indicates that the current frame is detected at the terminal device, without the need of resolution adjustment or model selection. In contrast, an action $\mathcal{A} = (1, 1080, 3)$ signifies that the current frame should be be transmitted at a resolution of $1080p$ to detection model 3 for processing.

3) Reward: Since the objective of the request scheduler is to optimize the overall coordination of the edge-assisted video detection system to maximize service quality, the reward function should consider two key factors: the detection accuracy of all frames within the current time epoch and the latency required to obtain the detection results. Although latency is formulated as a hard constraint in our optimization problem (i.e., $L_{f,t}^{R_f} \leq \mathcal{L}_{f,t}$), enforcing a strict step-function penalty in the DRL training process creates a discontinuous reward landscape, which can lead to severe policy oscillation and non-convergence. Instead, we formulate the latency penalty as a linear combination (i.e., a soft penalty). This design provides a smooth, dense gradient signal to guide the agent toward the Pareto optimal front of accuracy and latency, while hard constraints (e.g., skipping frames that exceed the deadline) can still be enforced during practical system execution. Our goal is to maximize detection accuracy while minimizing latency as much as possible. Therefore, the reward function for each time epoch is defined as follows:

$$r_t = \alpha A_t - (1 - \alpha) L_t. \tag{10}$$

Training Algorithm and Network Architecture: The objective of the agent is to maximize the expected cumulative return $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, where γ is the discount factor. To achieve this, we employ the **Advantage Actor-Critic (A2C)** algorithm, which improves training stability through a synchronous, deterministic actor-critic framework. The agent comprises a shared feature extractor parameterized by ψ, which feeds into two separate heads: the *Actor* network $\pi_\theta(a_t|s_t)$ and the *Critic* network $V_\phi(s_t)$.

Specifically, the input state s_t is first processed by the shared layers (typically fully connected layers with ReLU activation) to obtain a latent representation $h_t = f_\psi(s_t)$. The two heads then compute their outputs as:

$$\pi_\theta(a_t|s_t) = \text{Softmax}(W_\theta h_t + b_\theta), \quad V_\phi(s_t) = W_\phi h_t + b_\phi, \tag{11}$$

where θ and ϕ denote the specific parameters for the actor and critic branches, respectively.

To reduce the variance of the policy gradient, the Critic estimates the Advantage function A_t using the Temporal Difference (TD) error:

$$A_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t). \tag{12}$$

The overall optimization objective is to minimize the total loss L_{total}, composed of policy loss, value loss, and entropy regularization:

$$L_{total} = \underbrace{-\log \pi_\theta(a_t|s_t) A_t}_{\text{Policy Loss}} + \lambda_v \underbrace{\frac{1}{2} A_t^2}_{\text{Value Loss}} - \lambda_e \underbrace{H(\pi_\theta(\cdot|s_t))}_{\text{Entropy}}, \tag{13}$$

where λ_v and λ_e are weighting coefficients, and $H(\cdot)$ represents the entropy.

In our implementation, the DRL-A agent is trained offline using collected video traces and measured edge-network conditions. During online deployment, the trained actor network is fixed and directly used for real-time per-frame decision making, while no online gradient update is performed to avoid additional training overhead and instability.

B. Bayesian Optimization Algorithm

Since the system operates in a batch processing manner, adjusting model size and resource allocation for individual frames is not appropriate. The selection of LoRA rank and GPU resource allocation for the detection model is modeled as a contextual multi-armed bandit (MAB) problem [21], [22]. At the beginning of epoch t, the policy agent considers the current contextual information $c_t = (A_{t-1}, L_{t-1})$, which guides the generation of a new action $a_t = (\varphi_t^1, \cdots, \varphi_t^N, g_t^1, \cdots, g_t^N)$, where φ_t^i and g_t^i denote the LoRA rank and GPU allocation of the i-th detection model, respectively. During epoch t, the system's detection scheduler uses the configuration from action a_t to execute the detection task.

Classic MAB algorithms, such as LinUCB, typically assume that the reward function has a certain linear relationship in the contextual information. However, in our problem, the reward function exhibits nonlinearity due to complex relationships (such as those between computational resources and total system delay) rendering classical algorithms ineffective. To address this challenge, we introduce Bayesian optimization (BO) [23], [24], which treats the objective function as a black box and infers its value for a given input based on historical input-output observations. Bayesian optimization can efficiently approximate the optimal solution using a small number of sampling points in scenarios where the evaluation cost of the objective function is high. Typically, the reward function in Bayesian optimization is defined as: $y_t = f(z_t) + \epsilon_t$, where $z_t = (a_t, c_t)$ represents the action-context pair in epoch t. $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ represents random noise. We explicitly adopt a non-zero noise variance to account for inherent edge server hardware jitters (e.g., slight GPU inference fluctuations due to thermal throttling), preventing the algorithm from blindly overreacting to occasional system lags.

Based on the above problem framing and corresponding formulations, we treat the objective function as a stochastic process, using a prior distribution to represent the initial assumptions about the unknown function. As new observations are obtained, the prior distribution is updated to a posterior distribution, enabling a more accurate inference of the function's behavior. This update mechanism helps in selecting the next input point that is likely to achieve the highest reward. Here, we define the prior distribution of the objective function as a Gaussian Process (GP), which can be fully characterized by the mean function $\mu(z) = \mathbb{E}[f(z)]$ and the covariance function $k_z(z, z') = \mathbb{E}\left[(f(z) - \mu(z))(f(z') - \mu(z'))\right]$. We use a squared exponential kernel as kernel function, which is defined as:

$$k_z(z, z') = exp(-\frac{\| z - z' \|^2}{2l^2}), \tag{14}$$

where l determines the extent to which z and $z^{'}$ influence each other. The SE kernel is selected because similar resource configurations, such as nearby GPU allocations and LoRA ranks, tend to exhibit correlated system performance. Although these resource configurations are discrete in practice, configurations close to each other in the action-context space usually lead to comparable inference latency and detection accuracy. Therefore, the SE kernel can effectively capture such local correlation between similar action-context pairs.

At epoch t, the observed historical reward set is $Y_t = [y_1, \cdots, y_t]^T$, and corresponding to the action-context pair set is $Z_t = [z_1, \cdots, z_t]^T$. When a new sample point z_{t+1} arrives, the observed distribution is defined as:

$$\begin{bmatrix} Y_t \\ y_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(Z_t) \\ \mu(z_{t+1}) \end{bmatrix}, \begin{bmatrix} K_t + \sigma^2\mathbf{I} & k_t(z_{t+1}) \\ k_t^T(z_{t+1}) & k_z(z_{t+1}, z_{t+1}) \end{bmatrix}\right), \tag{15}$$

where $\mu(Z_t) = [\mu(z_1), \cdots, \mu(z_t)]^T$, $K_t = \left[k_z\left(z, z^{'}\right)\right]_{z,z^{'} \in Z}$, and $k_t(z) = [k_z(z_1, z), \cdots, k_z(z_t, z)]^T$. $M_t = \{(z_1, y_1), \cdots, (z_t, y_t)\}$ represents the observations collected from epoch 1 to epoch t. By applying the Sherman-Morrison-Woodbury formula [25], the posterior predictive distribution of y_{t+1} can be derived as:

$$y_{t+1} \mid M_t, z_{t+1} \sim \mathcal{N}(\mu(z_{t+1} \mid M_t), \sigma^2(z_{t+1} \mid M_t)). \tag{16}$$

Therefore, the posterior distribution over $f(\cdot)$ remains a Gaussian Process. The advantage of this approach is that it enables the prediction of $f(z_t)$ for any action-context pair z_t based solely on previously observed values. The mean, covariance, and variance of the posterior distribution $f(z_t)$ can be expressed as:

$$\begin{aligned} \mu_t(z) &= k_t(z)^T \left(K_t + \sigma^2\mathbf{I}\right)^{-1} Y_t, \ \sigma_t^2 = k_t(z, z), \\ k_t(z, z') &= k_z(z, z') - k_t(z)^T \left(K_t + \sigma^2\mathbf{I}\right)^{-1} k_t(z'). \end{aligned} \tag{17}$$

To determine the action to select in the next epoch, we adopt the Upper Confidence Bound (UCB) acquisition strategy:

$$a_{t+1} = \underset{a_{t+1} \in \mathcal{D}}{argmax}\ \mu_t(z_t) + \beta_t^{\frac{1}{2}} \sigma_t(z_t), \tag{18}$$

where $\mathcal{D}$ is the action space and β_t is a constant that controls the trade-off between exploration and exploitation. In a real-time video system, blindly exploring random configurations can cause severe video freezing. By tuning β_t, the UCB strategy ensures the system only tests new configurations when highly confident, ultimately locking into a stable setup for reliable detection.

We decouple resource allocation from the DRL agent and manage it via BO on a per-epoch basis. Integrating resource allocation directly into the DRL's action space would introduce unacceptable hardware context-switching overheads for per-frame decisions (i.e., timescale mismatch) and exponentially expand the action space, leading to severe DRL training instability. BO efficiently handles this low-frequency tuning, perfectly complementing the DRL's high-frequency decisions.

VI. Performance Evaluation

In this section, we evaluate the performance of *Adaptive-EVOD*, with the implementation and setup described in Section VI-A, ablation studies presented in Section VI-B, and comparative experiments detailed in Section VI-C.

A. Implementation and Setup

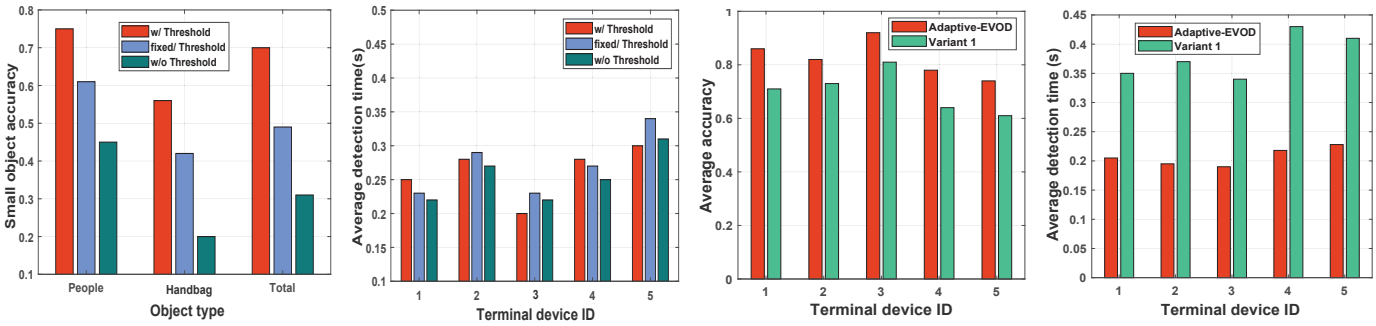
Testbed. Our experimental platform consists of five NVIDIA Jetson TX2 devices as terminals, while the edge server is based on a Supermicro M12SWA-TF server, equipped with an AMD Ryzen Threadripper PRO 5995WX processor and three NVIDIA GPUs (RTX 4090, RTX 3090, and RTX 2080 Ti) [26], [27]. The terminal devices run the lightweight YOLOv5s detection model, while the edge server hosts YOLOv5x, along with its multiple LoRA fine-tuned versions, to perform video-based object detection. The YOLOv5 series, due to its maturity, is widely used in edge-assisted video detection and adopted by State-of-the-Art methods (e.g. [28] in the year 2025). All devices are interconnected through a campus-local area network, where the network bandwidth predominantly fluctuates between 3 MB/s and 15 MB/s. Each time epoch is set to $t = 1s$. At the beginning of each time epoch t, the weight factor α is sampled from a uniform distribution within the range $[0.2, 0.7]$.

Dataset. The dataset utilized for the terminal devices is composed of surveillance videos from five distinct scenarios obtained from YouTube [18], [19], [29]–[31]. These videos encompass diverse visual characteristics: slow-moving street pedestrians with high object density (Video 1), fast-moving highway vehicles with massive inter-frame differences (Video 2), close-range face recognition with non-linear motion (Video 3), stable baseline road traffic (Video 4), and 4K extreme-distance traffic featuring a high proportion of challenging small objects (Video 5). To establish rigorous ground truth annotations, we first utilized the large-scale YOLOv5x6 model to generate preliminary detection results. Subsequently, these pre-labels and original frames were imported into the computer vision annotation tool (CVAT) for a meticulous frame-by-frame review by two independent annotators, who corrected bounding box drifts and manually annotated small objects missed by the pre-labeling model. Finally, a third supervisor conducted cross-validation to resolve any annotator discrepancies, ensuring that only labels reaching a strict consensus were finalized, consistent with previous studies [4], [32].

Metrics. We evaluated the following performance metrics: (a) Accuracy. It is defined as A_t in Section IV-B and evaluated across five types of video data, including overall detection accuracy and small object detection accuracy. (b) Time. This includes algorithm execution time, model inference time, and video transmission time. To comprehensively evaluate the aforementioned metrics, experiments are conducted under varying datasets, different numbers of terminal devices, and diverse bandwidth conditions.

B. In-depth Analysis of Adaptive-EVOD System

To demonstrate the effectiveness of specific components in *Adaptive-EVOD*, a series of ablation experiments are con-



(a) Small object accuracy. (b) Average detection time. (c) Average per-frame accuracy. (d) Average detection time.

Fig. 6: Ablation study on adaptive threshold, model adapter and resource scheduler.

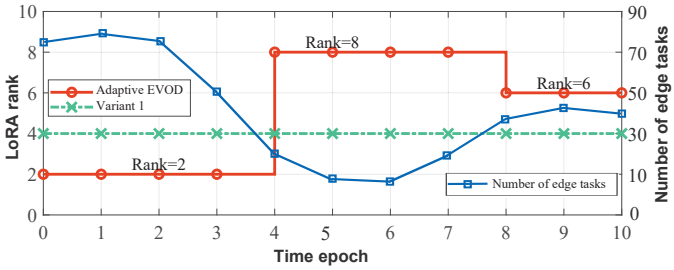
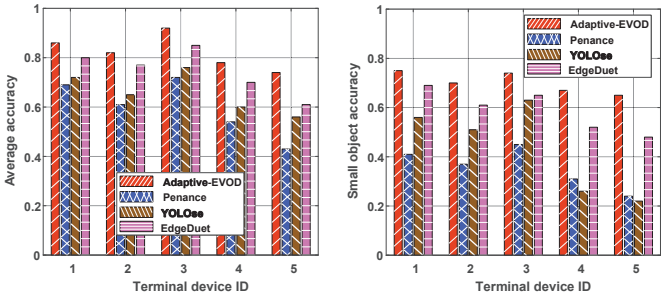
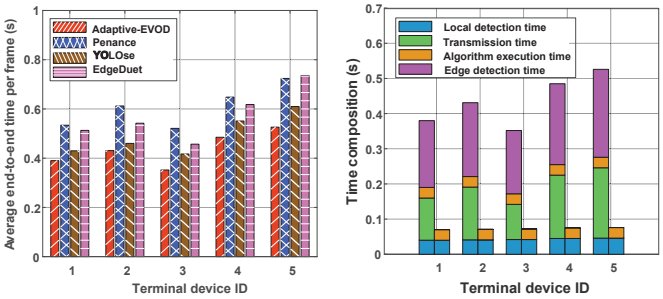


Fig. 7: The decision-making differences between Variant 1 and *Adaptive-EVOD*.



(a) Average per-frame accuracy. (b) Small object accuracy.

Fig. 8: Accuracy comparison of different schemes.



(a) Average end-to-end time. (b) Time composition.

Fig. 9: Time comparison of different schemes.

ducted in our systems.

Small object detection threshold. In *Adaptive-EVOD*, to enhance the detection accuracy of small objects, we assign different detection thresholds based on object types to better localize their positions. To evaluate the importance of this adaptive thresholding mechanism, we compare our work (*w/Threshold*) with two variants: (1) a baseline *Adaptive-EVOD* variant without any detection thresholding, *w/o Threshold*, and (2) a variant with fixed detection thresholds applied uniformly across all objects, *fixed/ Threshold*. As shown in Fig. 6(a) and (b), *w/ Threshold* improves the overall detection accuracy for small objects by 21% and 39% compared to *fixed/ Threshold* and *w/o Threshold*, respectively, with negligible additional detection latency. Furthermore, we report the detection accuracy for two representative small object classes, person and handbag. The results show that *w/ Threshold* achieves accuracy improvements of up to 24.6% and 35.7% for these two categories, respectively.

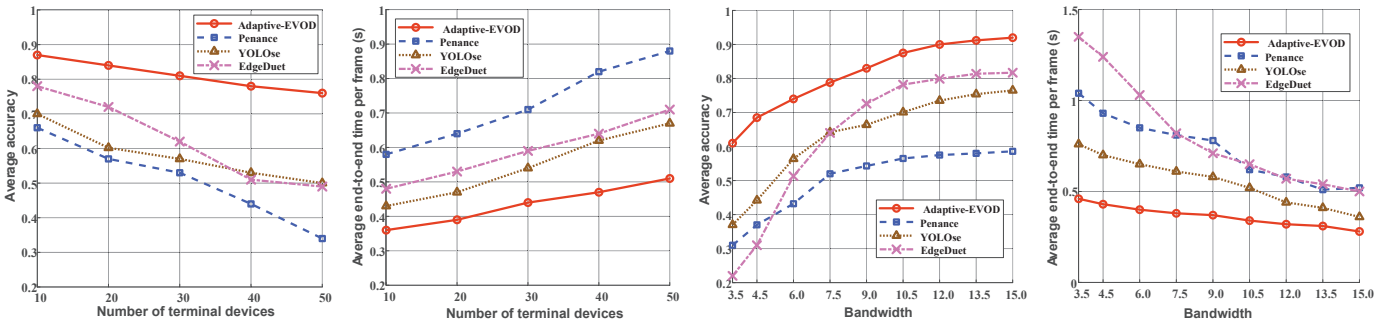
Model adapter and resource scheduler. To evaluate the efficacy of the Bayesian optimization-driven model adapter and resource scheduler, we compare the full *Adaptive-EVOD* system against a baseline variant (Variant 1) that relies on a static configuration with fixed LoRA rank and GPU allocation. For the BO module, the LoRA ranks are selected from $\{2, 4, 6, 8\}$. As shown in Fig. 6(c) and (d), compared to Variant 1, the full *Adaptive-EVOD* improves average per-frame detection accuracy by 12.4% and reduces detection time by 17.3% across the five video sequences.

In detail, the performance degradation of Variant 1 is primarily attributed to its static model selection and resource allocation strategy, which leads to severe GPU overload and prolonged queuing delays. As shown in Fig. 7, Variant 1 adopts a fixed configuration (LoRA rank = 4) across all time periods, regardless of system load. In contrast, at epoch $t = 1$, when the number of tasks on the edge side is more than 70, *Adaptive-EVOD* reduces the LoRA rank to 2, thereby decreasing the number of parameters and accelerating inference. By epoch $t = 4$, as the load decreases, it increases the LoRA rank to 8 to enhance detection accuracy.

C. Comparison with State-of-the-Art

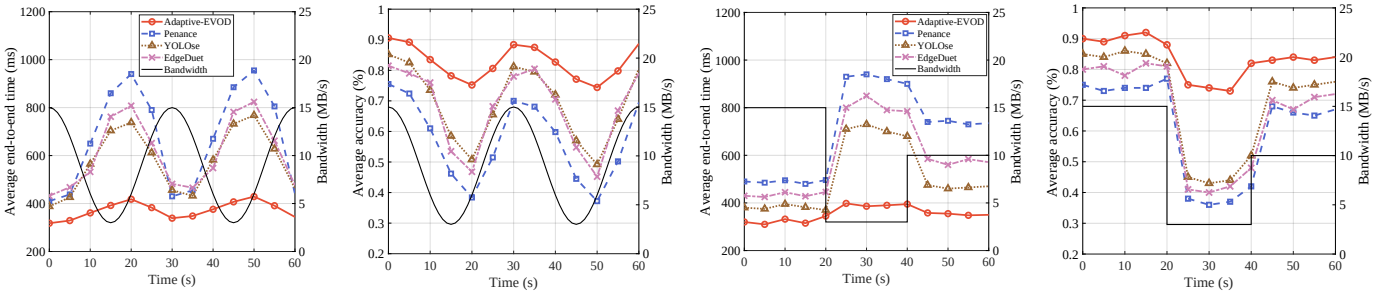
Benchmarks. We compare *Adaptive-EVOD* with the following three baseline schemes. (1) *Penance*, predicts the bitrate to dynamically adjust the compression parameters [33]. (2) *YOLOse*, retrains the model to enhance detection accuracy during the processing stage [28]. (3) *EdgeDuet*, employs tile-level parallel processing to improve the detection accuracy of small objects [15].

Accuracy. As shown in Fig. 8(a), the average detection accuracy of videos on each terminal device shows that *Adaptive-*



(a) Average per-frame accuracy. (b) Average end-to-end time. (c) Average per-frame accuracy. (d) Average end-to-end time.

Fig. 10: Impact of the number of terminal devices and bandwidth.



(a) Average end-to-end time. (b) Average detection accuracy. (c) Average end-to-end time. (d) Average detection accuracy.

Fig. 11: Performance and robustness evaluation under highly dynamic bandwidth conditions.

EVOD outperforms the other three baseline schemes from 7.8% to 23.4%. In addition, due to the varying nature of each video, detection accuracy also differs accordingly. For example, the video on Device 5 involves fast-moving vehicles, with a relatively low initial resolution and large frame differences, resulting in the lowest accuracy. As shown in Fig. 8(b), for small object detection, both *Adaptive-EVOD* and *EdgeDuet* improve accuracy significantly over *Penance* and *YOLOse*. However, as *EdgeDuet* does not consider adaptive model selection and dynamic resource allocation, its accuracy remains lower than that of *Adaptive-EVOD*.

Algorithm overhead. The average end-to-end time per video frame on each terminal device is shown in Fig. 9(a), in which *Adaptive EVOD* is the fastest and reduces latency by 5.7% to 18.3% compared to the baselines. To further analyze the time consumption of *Adaptive-EVOD*, Fig. 9(b) presents the average runtime of each module in *Adaptive-EVOD*. For frames that are not offloaded, only local detection time and algorithm execution time are involved. For frames offloaded to the edge server, the total time cost includes local detection time, transmission time, algorithm execution time, and edge detection time, among which transmission time and edge detection time account for the majority. Specifically, the DRL inference requires < 2 ms per frame via a lightweight MLP, while the Bayesian Optimization update takes only 10–15 ms per epoch. Overall, the combined algorithm execution time constitutes less than 3% of the total end-to-end latency. Thus, the execution time of our algorithm has limited impact on the total time cost.

Impact of scale and bandwidth. We evaluate the performance of *Adaptive EVOD* under varying numbers of terminal devices and network bandwidth conditions. Specifically, Fig. 10 (a) and (b) present results with a fixed bandwidth of 15 MB/s as the number of terminal devices scales from 10 to 50. Fig. 10 (c) and (d) investigate the impact of varying bandwidth from 3.5 MB/s to 15 MB/s while fixing the number of terminal devices at 5. Fig. 10 (a) and (b) show that *Adaptive EVOD* improves the average detection accuracy by 18.8% to 28.2% and reduces the average end-to-end latency by 20.5% to 40.2% compared to state-of-the-art methods. On the other hand, Fig. 10 (c) and (d) show that *Adaptive-EVOD* consistently outperforms all baseline methods in terms of overall system accuracy and latency across varying bandwidth conditions. Additionally, the curves in Fig. 10 demonstrate that our method can adaptively balance detection accuracy and latency under varying numbers of terminal devices (system load) and bandwidth conditions.

Robustness under highly dynamic bandwidth. To further evaluate system robustness, we test *Adaptive-EVOD* under two dynamic bandwidth traces in a 60 s controlled streaming experiment. The sinusoidal trace varies between 3 MB/s and 15 MB/s according to $B(t) = 9{+}6\sin\left(\frac{\pi t}{15} + \frac{\pi}{2}\right)$ MB/s, emulating continuous wireless fluctuations. The abrupt trace changes from 15 MB/s to 3 MB/s and then to 10 MB/s at 20-second intervals, emulating sudden network shocks. As shown in Fig. 11(a) and (b), under sinusoidal bandwidth variation, the baselines show clear latency oscillation and adaptation hysteresis. Their maximum latency reaches 955 ms, 768 ms, and 824 ms, respectively, while *Adaptive-EVOD* keeps its latency below 429 ms. Meanwhile, the minimum accuracy of the baselines drops to 0.372, 0.492, and 0.452, whereas *Adaptive-EVOD* maintains at least 0.744 accuracy. As shown

in Fig. 11(c) and (d), under abrupt bandwidth drops from 15 MB/s to 3 MB/s, the baselines suffer severe latency spikes and over 35% accuracy loss, while *Adaptive-EVOD* limits the accuracy loss to 16% by adaptively adjusting resolution and offloading decisions. These results demonstrate that *Adaptive-EVOD* achieves stable latency and robust accuracy under dynamic bandwidth conditions.

TABLE II: Paired sample t-test results comparing *Adaptive-EVOD* against SOTA baselines on latency and accuracy.

Metric	Compared Baseline	Calculated p-value	Significance
Latency	vs. Penance	1.34×10^{-15}	$p < 0.001$
Latency	vs. YOLOse	4.21×10^{-12}	$p < 0.001$
Latency	vs. EdgeDuet	8.56×10^{-8}	$p < 0.001$
Accuracy	vs. Penance	2.11×10^{-18}	$p < 0.001$
Accuracy	vs. YOLOse	6.73×10^{-14}	$p < 0.001$
Accuracy	vs. EdgeDuet	3.92×10^{-9}	$p < 0.001$

Statistical significance. To ensure the performance improvements are robust and not due to random chance, we conducted a paired-sample t-test comparing the frame-by-frame latency and accuracy of Adaptive-EVOD against the three baselines. The test is performed at the frame level: each paired sample corresponds to the performance of Adaptive-EVOD and a baseline on the same video frame under the same experimental condition. The paired samples are collected across all five video sequences and the evaluated bandwidth/device settings, resulting in 2880 paired frame-level observations for each comparison. As shown in Table II, the calculated p-values for all comparisons are mathematically well below the strict threshold of 0.001. This formally establishes that the performance gains achieved by our joint optimization strategy are highly statistically significant, yielding robust advantages over SOTA methods across heterogeneous video contents and highly dynamic bandwidths.

VII. Related Work

In edge-assisted object detection systems, video transmission, video processing, and small object detection are all critical components that collectively influence overall performance [34]–[37]. In video transmission, existing studies have centered on ensuring real-time and stable transport by adjusting quality parameters [9], regions of interest [10], and video block bitrates [38], [39] to enhance the efficiency of video frame transmission from the device to the edge server, thereby laying the foundation for real-time object detection. Regarding video processing, another core component of the system, video object detection optimization has also garnered significant attention [13], [14]. Research efforts have largely focused on super-resolution reconstruction [12], computational resource allocation [40], and detection model selection [41], [42] to improve detection accuracy, reduce inference latency. Lastly, for video frames containing a high proportion of small objects, enhancing small object detection can significantly improve the overall detection accuracy [15].

Aligned with the fundamental trend of integrating sensing, communication, and computation in edge AI [43]–[45], due to the intrinsic coupling between the two core tasks of edge-assisted object detection systems—video transmission and object detection—certain key factors simultaneously impact both stages, making it difficult to achieve system-wide optimal performance through single-stage optimization [46], [47]. To the best of our knowledge, a few studies have attempted to jointly optimize video transmission parameters (e.g., resolution, frame rate) and detection model selection within the field of edge-assisted video analytics [48], [49]. However, these approaches often overlook the importance of small object detection, resulting in suboptimal handling of frames containing a large number of small objects and consequently degrading overall detection performance [33], [50]. Furthermore, existing joint optimization studies often overlook the adaptive scheduling of edge server computational resources, making it challenging to maintain real-time detection in the highly dynamic environments [15], [51]–[53]. Compared with previous works, our approach performs joint optimization of video transmission and processing, and adaptively formulates a globally optimal strategy that incorporates small object detection, thereby maximizing the overall system efficiency.

VIII. Acknowledge

This work was supported in part by the Major Basic Research Program of Shandong Provincial Natural Science Foundation under Grant ZR2025ZD18, the National Natural Science Foundation of China under Grant 62572280, U24A20244.

IX. Conclusion

We propose *Adaptive-EVOD*, an edge-assisted video object detection system that enables intelligent decision making on parameter selection in video transmission stage and resource allocation in processing stage. *Adaptive-EVOD* first employs deep reinforcement learning to adaptively determine strategies on frame resolution and rate, and then utilizes Bayesian optimization to dynamically adjust resource usage and model selection, enabling joint optimization of both transmission and processing stages. In addition, *Adaptive-EVOD* places special emphasis on small object detection by introducing adaptive detection thresholds to enhance accuracy. We evaluate *Adaptive-EVOD* on a real-world platform, and experimental results demonstrate that it achieves higher detection accuracy and lower latency compared to three state-of-the-art baseline schemes, highlighting its overall superiority. In future work, we plan to extend Adaptive-EVOD to multi-edge scenarios, focusing on distributed resource allocation and service migration to enhance system scalability.

References

[1] Z. Li, Y. Zhang, L. Cai, and Y. Zhang, "HearLoc: Locating Unknown Sound Sources in 3D With a Small-Sized Microphone Array," *IEEE Transactions on Mobile Computing*, vol. 24, no. 4, pp. 3163–3177, 2025.

[2] X. Chi, H. Chen, G. Li, Z. Ni, N. Jiang, and F. Xia, "Edsp-edge: Efficient dynamic edge service entity placement for mobile virtual reality systems," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 2771–2783, 2024.

[3] X. Chi, H. Chen, Z. Ni, H. Sun, P. Sun, and D. Yu, "H-step: Heuristic stable edge service entity placement for mobile virtual reality systems," *IEEE Transactions on Mobile Computing*, vol. 24, no. 8, pp. 7377–7388, 2025.

[4] X. Dai, Z. Zhang, P. Yang, Y. Xu, X. Liu, and J. C. Lui, "Axiomvision: Accuracy-guaranteed adaptive visual model selection for perspective-aware video analytics," in *ACM MM 2024*, 2024, pp. 7229–7238.

[5] Z. Ma, Z. Zheng, J. Wei, X. Wei, Y. Yang, and H. T. Shen, "Open-scenario domain adaptive object detection in autonomous driving," in *ACM MM 2023*, 2023, pp. 8453–8462.

[6] S. Hao, W. Chai, Z. Zhao, M. Sun, W. Hu, J. Zhou, Y. Zhao, Q. Li, Y. Wang, X. Li, and G. Wang, "Ego3dt: Tracking every 3d object in ego-centric videos," in *ACM MM 2024*, 2024, pp. 2945–2954.

[7] C.-Y. Chen and H.-Y. Hsieh, "Cross-frame resource allocation with context-aware qoe estimation for 360° video streaming in wireless virtual reality," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 7887–7901, 2023.

[8] W. Ma and L. Mashayekhy, "Video offloading in mobile edge computing: Dealing with uncertainty," *IEEE Transactions on Mobile Computing*, vol. 23, no. 11, pp. 10 251–10 264, 2024.

[9] S. Bi, H. Chen, X. Li, S. Wang, Y. Wu, and L. Qian, "A two-stage deep reinforcement learning framework for mec-enabled adaptive 360-degree video streaming," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 14 313–14 329, 2024.

[10] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "Drl360: 360-degree video streaming with deep reinforcement learning," in *IEEE INFOCOM 2019*, 2019, pp. 1252–1260.

[11] T. Yuan, L. Mi, W. Wang, H. Dai, and X. Fu, "Accdecoder: Accelerated decoding for neural-enhanced video analytics," in *IEEE INFOCOM 2023*, 2023, pp. 1–10.

[12] J. Shen, H. Sheng, S. Wang, R. Cong, D. Yang, and Y. Zhang, "Blockchain-based distributed multiagent reinforcement learning for collaborative multiobject tracking framework," *IEEE Transactions on Computers*, vol. 73, no. 3, pp. 778–788, 2024.

[13] M. Zhang, Y. Zhu, L. Shen, F. Wang, and J. Liu, "Omnisense: Towards edge-assisted online analytics for 360-degree videos," in *IEEE INFOCOM 2023*, 2023, pp. 1–10.

[14] L. Shen, M. Zhang, C. Zhang, and J. Liu, "You only look once in panorama: Object detection for 360° videos with mlaas," in *NOSSDAV 2024*, 2024, pp. 1–7.

[15] Z. Yang, X. Wang, J. Wu, Y. Zhao, Q. Ma, X. Miao, L. Zhang, and Z. Zhou, "Edgeduet: Tiling small object detection for edge assisted autonomous mobile vision," *IEEE/ACM Transactions on Networking*, vol. 31, no. 4, pp. 1765–1778, 2023.

[16] S. Qi, H. Ma, Y. Zou, Y. Yuan, P. Li, and D. Yu, "Br-feel: A backdoor resilient approach for federated edge learning with fragment-sharing," *Journal of Systems Architecture*, vol. 155, p. 103258, 2024.

[17] ——, "Fed-ms: Fault tolerant federated edge learning with multiple byzantine servers," in *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2024, pp. 982–992.

[18] "Avg-towncentre - original video," https://www.youtube.com/watch?v=pC0FLXv5p1U, May 2012, [Online; accessed March 30, 2025].

[19] "4k road traffic video for object detection and tracking - free download now!" https://www.youtube.com/watch?v=MNn9qKG2UFI, October 2022, [Online; accessed March 30, 2025].

[20] J. Wang, M. Xu, L. Jiang, and Y. Song, "Attention-based deep reinforcement learning for virtual cinematography of 360° videos," *IEEE Transactions on Multimedia*, vol. 23, pp. 3227–3238, 2021.

[21] T. Ouyang, X. Chen, Z. Zhou, R. Li, and X. Tang, "Adaptive user-managed service placement for mobile edge computing via contextual multi-armed bandit learning," *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1313–1326, 2023.

[22] L. Sun, J. Hou, and T. Shu, "Spatial and temporal contextual multi-armed bandit handovers in ultra-dense mmwave cellular networks," *IEEE Transactions on Mobile Computing*, vol. 20, no. 12, pp. 3423–3438, 2021.

[23] A. Galanopoulos, J. A. Ayala-Romero, D. J. Leith, and G. Iosifidis, "Automl for video analytics with edge computing," in *IEEE INFOCOM 2021*, 2021, pp. 1–10.

[24] J. Yan, Q. Lu, and G. B. Giannakis, "Bayesian optimization for online management in dynamic mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 23, no. 4, pp. 3425–3436, 2024.

[25] H. Liu and G. Cao, "Deep learning video analytics through online learning based edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 10, pp. 8193–8204, 2022.

[26] S. Choi, S. Lee, Y. Kim, J. Park, Y. Kwon, and J. Huh, "Serving heterogeneous machine learning models on Multi-GPU servers with Spatio-Temporal sharing," in *USENIX ATC 2022)*, 2022, pp. 199–216.

[27] Y. Kim, Y. Choi, and M. Rhu, "Paris and elsa: an elastic scheduling algorithm for reconfigurable multi-gpu inference servers," in *ACM DAC 2022*, 2022, pp. 607–612.

[28] Z.-Y. Li, X. Jin, B.-Y. Sun, C.-L. Guo, and M.-M. Cheng, "Towards raw object detection in diverse conditions," in *IEEE CVPR 2025*, June 2025, pp. 8859–8868.

[29] "Geovision vd8700 face recognition ip camera demo video," https://www.youtube.com/watch?v=Szt_kHWKwNo, April 2019, [Online; accessed March 30, 2025].

[30] "Road traffic video for object recognition," https://www.youtube.com/watch?v=wqctLW0Hb_0, August 2023, [Online; accessed March 30, 2025].

[31] "4k camera example for traffic monitoring road, interchage," https://www.youtube.com/watch?v=Y7Kv_oJtuJE, February 2023, [Online; accessed March 30, 2025].

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

[33] M. Yan, Y. Wang, X. Xiao, Z. Luo, J. He, and W. Wang, "Think before you leap: Content-aware low-cost edge-assisted video semantic segmentation," in *ACM MM 2023*, 2023, pp. 9224–9233.

[34] X. Yuan, L. Pu, J. Shi, Q. Gong, and J. Xu, "Muster: Multi-source streaming for tile-based 360° videos within cloud native 5g networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 11, pp. 6616–6632, 2023.

[35] L. Zhang, H. Zhou, H. Wang, and L. Cui, "Tbsr: Tile-based 360° video streaming with super-resolution on commodity mobile devices," in *IEEE INFOCOM 2024*, 2024, pp. 501–510.

[36] M. Irfan, K. Muhammad, M. Sajjad, K. M. Malik, F. Alaya Cheikh, J. J. P. C. Rodrigues, and V. H. C. de Albuquerque, "Deepview: Deep-learning-based users field of view selection in 360° videos for industrial environments," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 2903–2912, 2023.

[37] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang, "Pano: optimizing 360° video streaming with a better understanding of quality perception," in *ACM SIGCOMM 2019*, 2019, pp. 394–407.

[38] S. Wang, S. Yang, H. Su, C. Zhao, C. Xu, F. Qian, N. Wang, and Z. Xu, "Robust saliency-driven quality adaptation for mobile 360-degree video streaming," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1312–1329, 2024.

[39] L. Zhang, Y. Suo, X. Wu, F. Wang, Y. Chen, L. Cui, J. Liu, and Z. Ming, "Tbra: Tiling and bitrate adaptation for mobile 360-degree video streaming," in *ACM MM 2021*, 2021, pp. 4007–4015.

[40] M. A. Arefeen and M. Y. S. Uddin, "Poster: Towards resource-efficient detection-driven processing of multi-stream videos," in *ACM MobiCom 2021*, 2021, pp. 843–845.

[41] M. Hanyao, Y. Jin, Z. Qian, S. Zhang, and S. Lu, "Edge-assisted online on-device object detection for real-time video analytics," in *IEEE INFOCOM 2021*, 2021, pp. 1–10.

[42] B. Qian, Z. Wen, J. Tang, Y. Yuan, A. Y. Zomaya, and R. Ranjan, "Osmoticgate: Adaptive edge-based real-time video analytics for the internet of things," *IEEE Transactions on Computers*, vol. 72, no. 4, pp. 1178–1193, 2023.

[43] D. Wen, S. Xie, X. Cao, Y. Cui, J. Xu, Y. Shi, and S. Cui, "Integrated Sensing, Communication, and Computation for Over-the-Air Federated Edge Learning," *IEEE Transactions on Wireless Communications*, vol. 25, pp. 2748–2762, 2026.

[44] L. Song, J. Li, H. Jiang, S. Wei, and Y. Guo, "CHPFL: Clustered Adaptive Hierarchical Federated Learning for Edge-Level Personalization," *High-Confidence Computing*, vol. 6, no. 1, p. 100343, 2026.

[45] H. Jiang, S. Chen, T. Ouyang, M. Lin, X. Zhang, Q. Zhang, and X. Chen, "MFEL-H2B: Multimodal Federated Edge Learning With Heterogeneity-Aware Balancing Across Modalities and Models," *High-Confidence Computing*, p. 100384, 2026.

[46] J. Li, J. Liao, B. Chen, A. Nguyen, A. Tiwari, Q. Zhou, Z. Yan, and K. Nahrstedt, "Latency-aware 360-degree video analytics framework for first responders situational awareness," in *NOSSDAV 2023*, 2023, pp. 8–14.

[47] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *IEEE INFOCOM 2020*, 2020, pp. 257–266.

[48] Y. Yan, S. Zhang, X. Wang, N. Chen, Y. Chen, Y. Liang, M. Xiao, and S. Lu, "Visflow: Adaptive content-aware video analytics on collaborative cameras," in *IEEE INFOCOM 2024*, 2024, pp. 2019–2028.
[49] T. Murad, A. Nguyen, and Z. Yan, "Dao: Dynamic adaptive offloading for video analytics," in *ACM MM 2022*, 2022, pp. 3017–3025.
[50] C. Puligheddu, J. Ashdown, C. F. Chiasserini, and F. Restuccia, "Sem-o-ran: Semantic and flexible o-ran slicing for nextg edge-assisted mobile systems," in *IEEE INFOCOM 2023*, 2023, pp. 1–10.
[51] P. Dai, Y. Chao, X. Wu, K. Liu, and S. Guo, "Context-aware offloading for edge-assisted on-device video analytics through online learning approach," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 12 761–12 777, 2024.
[52] R. Wang, Y. Hao, Y. Miao, L. Hu, and M. Chen, "Rt3c: Real-time crowd counting in multi-scene video streams via cloud-edge-device collaboration," *IEEE Transactions on Services Computing*, vol. 17, no. 4, pp. 1739–1752, 2024.
[53] Y. Zhao, M. Dong, Y. Wang, D. Feng, Q. Lv, R. P. Dick, D. Li, T. Lu, N. Gu, and L. Shang, "A reinforcement-learning-based energy-efficient framework for multi-task video analytics pipeline," *IEEE Transactions on Multimedia*, vol. 24, pp. 2150–2163, 2022.

Xuejian Chi received the B.E. degree in Automation in 2022 and M.E. degree in Control Science and Engineering in 2024 both from China University of Petroleum. He is currently pursuing a Ph.D. degree in the School of Computer Science and Technology in Shandong University, China. His current research interests include edge computing and edge intelligence.

Xiaoxi Zhang (Member, IEEE)is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-sen University. She received the B.E. degree in electronics and information engineering from the Huazhong University of Science and Technology in and the Ph.D. degree in computer science from The University of Hong Kong. She was a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, Carnegie Mellon University. She is broadly interested in algorithm design, optimization theories, and implementation for networked systems.

Yifei Zou (Member, IEEE) received the BEng degree from Wuhan University, China, in 2016, and the Ph.D. degree from The University of Hong Kong, China, in 2020. He is currently an associate professor at the School of Computer Science and Technology, Shandong University, Qingdao, China. His research interests include edge intelligence, Internet of agents, and distributed computing.

Xingze Wu received his Bachelor's degree from North China Electric Power University, China, in 2023. He is currently pursuing his Master's degree at the School of Computer Science and Technology, Shandong University. His research interests include edge computing, wireless sensor networks, and federated learning.

Zhipeng Cai (Fellow, IEEE) is currently an Associate Professor in the Department of Computer Science at Georgia State University, USA. He received his PhD and M.S. degrees in the Department of Computing Science at University of Alberta, and B.S. degree from Beijing Institute of Technology. Dr. Cai's research areas focus on Wireless Networking, Internet of Things, Machine Learning, Cyber-Security, Privacy and Big data. Dr. Cai is the recipient of an NSF CAREER Award. He served as a Steering Committee Co-Chair and a Steering Committee Member for WASA and IPCCC. Dr. Cai also served as a Technical Program Committee Member for more than 20 conferences, including INFOCOM, MOBIHOC, ICDE, and ICDCS. Dr. Cai has been serving as an Associate Editor-in-Chief for Elsevier High-Confidence Computing Journal (HCC), and an Associate Editor for several international journals, such as IEEE Internet of Things Journal (IoT-J), IEEE Transactions on Knowledge and Data Engineering (TKDE), and IEEE Transactions on Vehicular Technology (TVT). He has published more than 70 papers in prestigious journals with more than 40 papers published in IEEE/ACM Transactions.

Falko Dressler (Fellow, IEEE) received his M.Sc. and Ph.D. degrees from the Dept. of Computer Science, University of Erlangen in 1998 and 2003, respectively. He is full professor and Chair for Telecommunication Networks at the School of Electrical Engineering and Computer Science, TU Berlin. Dr. Dressler has been associate editor-in-chief for IEEE Trans. on Network Science and Engineering, IEEE Trans. on Mobile Computing and Elsevier Computer Communications as well as an editor for journals such as IEEE/ACM Trans. on Networking, Elsevier Ad Hoc Networks, and Elsevier Nano Communication Networks. He has been chairing conferences such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, IEEE GLOBECOM. He authored the textbooks Self-Organization in Sensor and Actor Networks published by Wiley & Sons and Vehicular Networking published by Cambridge University Press. He has been an IEEE Distinguished Lecturer as well as an ACM Distinguished Speaker. Dr. Dressler is an IEEE Fellow, an ACM Fellow, and an AAIA Fellow. He is a member of the German National Academy of Science and Engineering (acatech). He has been serving on the IEEE COMSOC Conference Council and the ACM SIGMOBILE Executive Committee. His research objectives include next generation wireless communication systems in combination with distributed machine learning and edge computing for improved resiliency. Application domains include the internet of things, cyber-physical systems, and the internet of bio-nano-things.

Dongxiao Yu (Senior Member, IEEE) received the B.S. degree in 2006 from the School of Mathematics, Shandong University and the Ph.D degree in 2014 from the Department of Computer Science, The University of Hong Kong. He became an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2016. He is currently a professor in the School of Cryptography Science and Engineering, Shandong University. His research interests include wireless networks, distributed computing and graph algorithms.