JAVP: Joint-Aware Video Processing with Edge-Cloud Collaboration for DNN Inference

Zheming Yang
Institute of Computing Technology,
Chinese Academy of Science
Beijing, China
yangzheming19b@ict.ac.cn

Wen Ji
Institute of Computing Technology,
Chinese Academy of Science
Beijing, China
jiwen@ict.ac.cn

Qi Guo
Institute of Computing Technology,
Chinese Academy of Science
Beijing, China
guiorguo@163.com

Zhi Wang
Tsinghua University
Shenzhen, China
wangzhi@sz.tsinghua.edu.cn

ABSTRACT

Currently, massive video inference tasks are processed through edge-cloud collaboration. However, the diverse scenarios make it difficult to allocate the inference tasks efficiently, resulting in many wasted resources. In this paper, we propose a joint-aware video processing (JAVP) architecture for edge-cloud collaboration. First, we develop a multiscale complexity-aware model for predicting task complexity and determining its suitability for edge or cloud servers. The task is subsequently efficiently scheduled to the appropriate servers by integrating complexity with an adaptive resource-aware optimization algorithm. For input tasks, JAVP can dynamically and intelligently select the most appropriate server. The evaluation results on public datasets show that JAVP can improve the throughput by more than 70% compared to traditional cloud-only solutions while meeting accuracy requirements. And JAVP can improve the accuracy by 3%-5% and reduce delay and energy consumption by 16%-50% compared to state-of-the-art edge-cloud solutions.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Computer systems organization** → *Real-time system architecture.*

KEYWORDS

Video processing; collaborative architecture; edge computing

ACM Reference Format:
Zheming Yang, Wen Ji, Qi Guo, and Zhi Wang. 2023. JAVP: Joint-Aware Video Processing with Edge-Cloud Collaboration for DNN Inference. In *Proceedings of Proceedings of the 31st ACM International Conference on Multimedia (MM '23).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3613914

MM '23, October 29–November 3, 2023, Ottawa, ON, Canada.

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3613914

1 INTRODUCTION

Over the years, the widespread applications of Internet of Things (IoT) and deep learning has led to the gradual emergence of the Internet of Everything (IoE) [20], capturing the attention of the general public. IoT devices generate a vast amount of data and video data is progressively assuming paramount significance, with the proportion reaching 89% in 2025 [1]. Many cities around the world deploy millions of surveillance cameras and use deep neural network (DNN) models to process the video data [18, 41]. Cloud servers can guarantee high accuracy by deploying complex DNN models. However, for the exponentially growing surveillance video data, it is difficult to satisfy tasks with low delay requirements if all tasks are uploaded to the cloud for processing due to limited bandwidth resources [7]. To save bandwidth and ensure real-time video processing, a large amount of video data flows from the cloud to the edge side [5]. However, edge servers with limited resources can usually only deploy small DNN models [45], which is difficult to meet the accuracy requirements of complex tasks.

Recently, to improve the performance of video processing, edge-cloud collaborative architecture [13] is becoming the mainstream method, as shown in Figure 1. It can combine the powerful computing capability of cloud servers and the proximity communication capability of edge servers [11]. It can provide lower delay and energy consumption for task transmission and inference while ensuring accuracy requirements [30]. For the required video processing, video tasks must be continuously uploaded to the edge or cloud servers [6] and then inferred and analyzed by the DNN. The exponential growth of video data poses challenges in developing an efficient edge-cloud collaborative decision scheme. Recent efforts have developed DNN inference offloading mechanisms [14, 17, 19, 38, 39] where video tasks are transferred from the end to available resource-rich servers for inference. However, the above methods rely heavily on known video information and lack aware analysis of complexity. In real scenarios, the large amount of video surveillance data is complex and variable [9] (e.g., lighting, number of objects, and weather). This poses a great challenge to edge-cloud collaborative video processing under resource-constrained conditions.

In this paper, we propose a joint-aware video processing architecture for edge-cloud collaboration, named JAVP. To the best of our knowledge, this is the first edge-cloud collaborative real-time

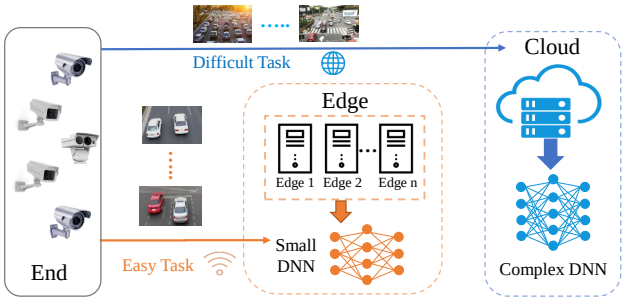


Figure 1: The illustration of edge-cloud collaborative architecture. The input tasks are pre-processed to determine whether to upload them to the edge or cloud server.

video processing architecture that supports both complexity-aware and resource-aware. For real-time video analytics tasks, JAVP can analyze their difficulty and resource situation by pre-processor. The main goal of JAVP is to adaptively select the right server under resource-constrained conditions through joint-aware and optimize the tradeoff between video processing accuracy, delay, and energy consumption. The main contributions of this paper can be summarized as follows.

- To deeply analyze the scene characteristics of video tasks, we propose a multiscale complexity-aware model, which is developed through the renormalization group method. It can predict the complexity based on the global and local features of the task and divide the complexity into three levels C_{low}, C_{medium}, and C_{high}.
- We model the difference between accuracy and energy consumption as a utility function and develop an adaptive resource-aware optimization algorithm. It can optimize the tradeoff between accuracy, delay, and energy consumption under resource-constrained conditions by combining complexity-aware.
- We evaluate the performance of the proposed architecture and compare it with the baseline method. The experimental results show that JAVP can adaptively allot the input task to the optimal server and optimize resource allocation through joint-aware. It can reduce video processing delay and energy consumption by 16%-50% without sacrificing accuracy.

The remainder of this paper is organized as follows. We first summarize the related work in Section 2. We then explain the research motivation in Section 3. In Section 4, the proposed JAVP architecture is described in detail. We evaluate our proposed method through extensive experiments in Section 5. Section 6 concludes the paper and presents future work.

2 RELATED WORK

2.1 Edge-cloud Collaboration for Inference Offloading

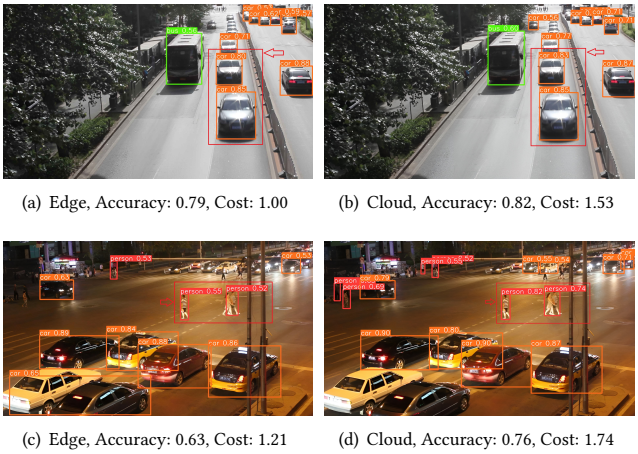
Some researchers have proposed various edge-cloud collaboration solutions based on task offloading. The works in [23, 33] try to improve the scheduling strategy for edge-cloud collaboration from the perspective of task priority analysis but ignore the tradeoff between accuracy and resources. To reduce the delay, the authors in [27] propose an edge-cloud collaborative inference scheduling system based on time-aware. It can predict the inference time of a DNN by characterizing the network and then determining the processing server for the task by a scheduling algorithm. To reduce the bandwidth requirement, an object detection system based on pipelined bidirectional tracking was proposed [25], which can adaptively perform keyframe selection for different input videos.

Video data is typically very large and requires frequent transfers to the edge and cloud servers. The simultaneous upload of large-scale tasks can affect the accuracy and real-time of video processing. Offloading video tasks from the end side to edge or cloud servers is a critical step in computationally intensive video processing. The works in [35] describe an edge-cloud collaboration system for real-time querying of large-scale video streams. They use an intelligent task allocator to balance the load between different compute nodes and improve the performance of real-time queries. In addition, the quality of the tasks is emphasized and divided according to the performance of the task model [44]. They develop a polynomial time algorithm based on task forest to solve this problem. To ensure the accuracy of video processing, the authors in [29] propose a dynamic adaptive task offloading framework that jointly considers the network bandwidth and the parameter configuration of the video content. It can maximize the DNN inference accuracy by dynamically adjusting the video bitrate and resolution. Due to the complexity of tasks, the above methods are difficult to achieve efficient edge-cloud collaborative task offloading with the current task information.

2.2 Edge-cloud Collaboration for Resource Allocation

Resource allocation in edge-cloud collaboration is crucial to achieving the tradeoff between accuracy, delay, and energy consumption for DNN inference. For the resource-constrained problem, the video processing algorithm needs to be optimized to reduce the computational volume. The frame masking mechanism [26] reduces the resource consumption of video processing by retaining only those regions that may contain objects of interest. A pareto-optimal tradeoff between accuracy and bandwidth is then achieved by controlling the masked portion of the frames and the video resolution. AppealNet [22] predicts whether input data can be transmitted to the edge by bidirectional neural network architecture. It explicitly considers the inference difficulty and optimizes the tradeoff between accuracy and energy consumption of task processing. Unbalanced resource allocation makes video processing suffer from significant inefficiencies. The authors in [42] propose a resource scheduling framework that reduces costs by aggregating a number of dispersed video tasks and assigning them to fewer preselected nodes. However, this approach may affect the final inference accuracy when the task is complex.

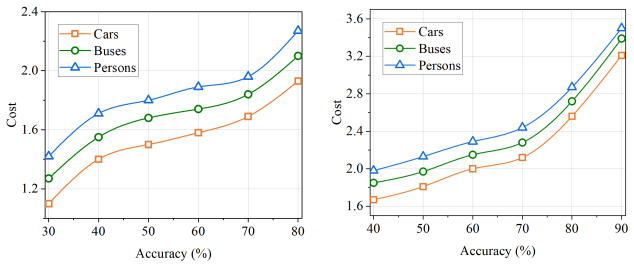
In addition, some works consider how to rationally allocate computing and bandwidth resources between the edge and cloud servers to achieve optimal video processing efficiency. The authors in [36] propose a deep reinforcement learning-based approach that improves the efficiency of resource allocation between edge and



(a) Edge, Accuracy: 0.79, Cost: 1.00 (b) Cloud, Accuracy: 0.82, Cost: 1.53

(c) Edge, Accuracy: 0.63, Cost: 1.21 (d) Cloud, Accuracy: 0.76, Cost: 1.74

Figure 2: The inference results for different tasks on edge and cloud. The (a) and (b) are easy tasks, and the (c) and (d) are difficult tasks. Simple tasks are more suitable at the edge, and the cloud is suitable for processing difficult tasks.



(a) The relationship between accuracy requirements and cost at the edge (b) The relationship between accuracy requirements and cost at the cloud

Figure 3: The results show that a large difference in the cost of different accuracy requirements and the effect of different servers on a video task is different.

cloud servers through dynamic DNN model selection and inference load optimization. The authors in [15] model the dynamic decision-making of collaboration strategies as an online optimization problem. The resource consumption of edge-cloud collaborative video processing is then optimized by a budget-constrained multi-armed bandit algorithm. NestDNN [12] is a framework that takes the dynamics of runtime resources into account. It dynamically selects the best resource accuracy tradeoff for each DNN model to match the model's resource requirements with the system's available resources. However, in some complex scenarios, the above methods do not analyze the difficulty of the tasks and cannot meet the requirements of variable tasks.

3 MOTIVATION

This section first analyzes the differences of existing object detection tasks under different servers, mainly including task processing accuracy and energy consumption cost. Then we show our measurement results on actual servers and describe the necessity and feasibility of JAVP.

We take the object detection task of traffic scene as an example to measure in the UA-DETRAC dataset [37]. The detection objects include cars, buses, and persons. Then we implement YOLOv5 [2] on NVIDIA Jetson Xavier NX and NVIDIA A100 GPU. NVIDIA Jetson Xavier NX is used as an edge server and deployed with YOLOv5s, and NVIDIA A100 GPU is used as a cloud server and deployed with YOLOv5x. The number of DNN model parameters on the cloud server is about 10 times more than that of the edge server, the test results are shown in Figure 2. Where accuracy is calculated by mAP50, which is the mean average precision of multiple objects when the threshold of Intersection over Union (IoU) is greater than 0.5. The cost is represented by the energy consumption during inference and is normalized. It is evident that the inference accuracy is higher on the cloud server, but it is also accompanied by more cost. On the other hand, inference on edge servers is less accurate but also costs less. Further, we find that different scenarios have a huge impact on the inference process of edge-cloud collaboration. For example, the task in Figure 2(a) achieves good results on edge servers due to its low complexity. In contrast, the task in Figure 2(c) is more suitable for uploading to the cloud for processing. The high complexity leads to poor processing results on edge servers.

Traditional solutions largely ignore the complex diversity of video data, which results in huge cost wastage. For example, many simple tasks are incorrectly offloaded to the cloud server, thereby incurring supplementary costs. However, the video tasks in real-world scenarios are complex and variable, primarily from varying number of objects, distances, light, etc. [8]. This has a great impact on the video processing efficiency of the edge-cloud collaborative architecture. Furthermore, considering the limited resources of edge servers, it becomes crucial to strategically allocate these resources to enhance the processing efficiency of video data, especially for computationally intensive tasks [10]. Figure 3 illustrates the relationship between video processing cost and accuracy requirements. To obtain optimal edge-cloud collaborative inference performance, a method that can efficient analysis the complexity of video tasks is necessary. Then a resource optimization method is used to further improve the video processing efficiency. Therefore, we will explore the edge-cloud collaborative inference method based on joint-aware to solve the above video optimization challenge.

4 THE PROPOSED JAVP ARCHITECTURE

In this section, we present JAVP, an edge-cloud collaborative inference architecture based on joint-aware. Section 4.1 shows the overall design of JAVP. Then, the multiscale complexity-aware analysis module is introduced in Section 4.2 and the adaptive resource-aware optimization module is described in Section 4.3.

4.1 Overall Design

The architecture mainly includes end cameras, edge servers, and cloud servers. Before using this architecture, complex DNN models need to be deployed on the cloud, and small DNN models at the edge for inference of video tasks. The core of the JAVP is the complexity-aware analysis module and the resource-aware optimization module. Figure 4 shows the overall workflow of the JAVP architecture.

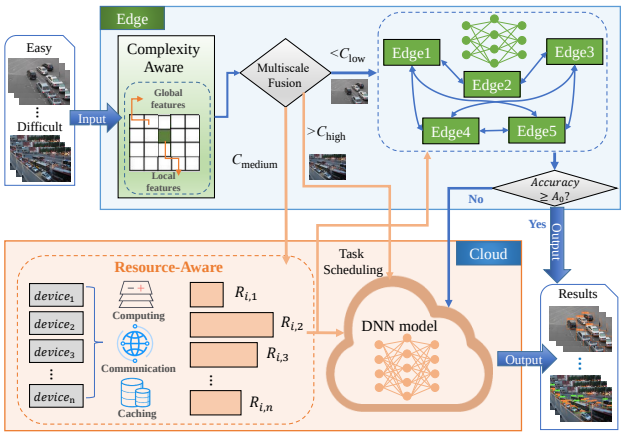


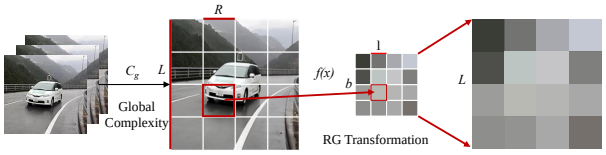
Figure 4: The workflow of the proposed JAVP architecture.

First, the input video is pre-processed in the complexity-aware analysis module. To better describe the scene complexity, we use a multiscale fusion approach to obtain the complexity of the video task. Then the appropriate server is decided based on the complexity of the video task. If the complexity is greater than the set threshold C_{high}, the video task is uploaded to the cloud server for processing. If it is less than the C_{low}, it is processed on the edge server. Otherwise, it is combined with the current resource situation to make a comprehensive decision on whether to upload to the edge or cloud server. In addition, if the inference accuracy on the edge meets the requirements, the result will be output, otherwise it will be uploaded to the cloud for in-depth processing.

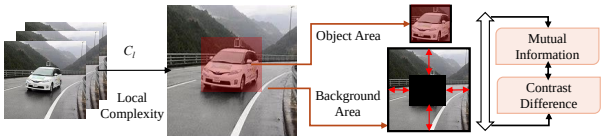
Due to the presence of multiple servers, the resource allocation strategy is critical to the overall processing efficiency when the number of tasks increases [43]. We deploy the resource-aware optimization module and comprehensively analyze computing, communication, and caching resources. Overall, JAVP can reduce energy consumption and ensure accuracy through the complexity-aware analysis module. And the resource-aware optimization module is used to guarantee low delay and reduce energy consumption. Finally, the optimal tradeoff between accuracy, delay, and energy consumption in real-time video analytics is achieved through edge-cloud collaborative joint-aware under resource-constrained.

4.2 Multiscale Complexity-Aware Analysis

The complexity-aware analysis module is mainly used to extract the structural features of input tasks and provide decision information for edge-cloud collaboration. The structural complexity of an image is the key information used to express the characteristics of a scene. By characterizing and quantifying the complexity of an image, the intuitive feeling of "difficult" and "simple" can be effectively reflected. The renormalization group (RG) has been proven to quantitatively describe the complexity of images by relating information between different scales [3]. RG is a method to examine system changes at different scales [21]. Based on this, we propose a complexity calculation method based on multiscale feature fusion, as shown in Figure 5. The global complexity of the image is first analyzed by RG transformation. The RG transformation is a mathematical method employed to address the issue of scale variations in physical systems. It enables the mapping of the original system to a new system while preserving its essential physical characteristics. Furthermore, mutual information and contrast differences are computed to quantify the local complexity. Finally, the results of global complexity and local complexity are combined to get the input task's scene complexity: $C = \alpha \cdot C_g + \beta \cdot C_l$, where α and β are the weights of global complexity and local complexity.



(a) Global complexity-aware analysis



(b) Local complexity-aware analysis

Figure 5: The overview of complexity-aware analysis.

4.2.1 Global Complexity. First, a new feature binary is formed by introducing the location information of image pixels, denoted as (w, h). Where w denotes the row index of the pixel and h denotes the column index of the pixel. A scale can be considered as a function $f(x)$ defined on some image domain ϕ. We then compute the RG transformation by means of the Kadanoff-Baym [31] equation:

$$\begin{aligned} RG_\lambda = & \mid \langle f_\lambda(x) \mid f_{\lambda+d\lambda}(x) \rangle - \\ \frac{1}{2} & \left(\langle f_\lambda(x) \mid f_\lambda(x) \rangle + \langle f_{\lambda+d\lambda}(x) \mid f_{\lambda+d\lambda}(x) \rangle \right) \mid = \\ & \frac{1}{2} \int_\phi \left(f_{\lambda+d\lambda}(x) - f_\lambda(x) \right)^2 dx \end{aligned} \tag{1}$$

where λ is the scale of the image, and $\langle f(x) \mid g(x) \rangle = \int_\phi dx f(x) g(x)$ represents the non-normalized overlap of two images of different scales. During each iteration of the renormalization procedure, the image is partitioned into blocks of dimensions $R \times R$. Each of these blocks is then replaced by a single pixel whose state is determined by $s_{wh}(n) = 1/R^2 \sum_b \sum_m s_{Rw+m,Rh+b}(n-1)$, where b and m indices enumerate the pixels within the same block and n denotes the number of iterations. This process is repeated multiple times, generating a series of renormalized images at varying resolutions. With this series, we are able to calculate the degree of overlap between images at different scales:

$$\begin{aligned} O_{n,n-1} &= \frac{1}{L_{n-1}^2} \sum_{w=1}^{L_n} \sum_{h=1}^{L_n} \mathbf{s}_{wh}(n) \cdot \sum_{m=1}^{R} \sum_{b=1}^{R} \mathbf{s}_{Rw+m,Rh+b}(n-1) \\ &= \frac{R^2}{L_{n-1}^2} \sum_{i=1}^{L_n} \sum_{j=1}^{L_n} \mathbf{s}_{wh}^2(n) = \frac{R^2}{L_{n-1}^2} \cdot L_n^2 \cdot O_{n,n} \end{aligned} \tag{2}$$

where $n = 0$ denotes the initial image, while $O_(n, n)$ represents the overlap between the image at scale n and its own self. Finally,

the global complexity C_g serves as an integral measure that takes into account the features that arise at each subsequent scale. The computation formula for C_g is presented as follows:

$$C_g = \sum_{n=0}^{N-1} C_n = \sum_{n=0}^{N-1} \left| O_{n+1,n} - \frac{1}{2}\left(O_{n,n} + O_{n+1,n+1}\right) \right| \tag{3}$$

4.2.2 Local Complexity. The local complexity of the input task is represented by the difference between the object area and the background area in the image.First, the object area is defined as the region of interest (ROI) of the image and the rest as the background area. To characterize the spatial features, the pixel point is introduced with information on the neighborhood of that point. We use a to denote the grayscale value of a pixel and e to denote the neighborhood mean grayscale value. The information entropy is calculated as follows: $H = -\sum_a \sum_e P_{a,e} \lg\left(P_{a,e}\right)$. $P_{a,e}$ is the probability of occurrence of a grayscale value in an image. The mutual information is as follows: $\Delta H = H_o + H_b - H_{o,b}$, where H_o and H_b are the information entropy of the object area and the background area. $H_{o,b}$ is the joint entropy of the object area and background area. The difficulty in discerning the object from the background, or the increase in local complexity, is directly proportional to the reduction in mutual information.

Finally, the contrast difference between the object area and background area is calculated by summing the roots of squares as follows:

$$\Delta\theta = \left[(\mu_o - \mu_b)^2 + \sigma_o^2\right]^{\frac{1}{2}} \tag{4}$$

where μ_o is the mean grayscale of the object, μ_b is the mean grayscale of the background, and σ_t represents the grayscale standard deviation of the object. The smaller the contrast difference is, the greater the complexity of the image localization. Overall, the local complexity of the input task is $C_l = \beta_1 \cdot \Delta H + \beta_2 \cdot \Delta\theta$. Where β_1 and β_2 are the weights of mutual information and contrast difference.

4.3 Adaptive Resource-Aware Optimization

4.3.1 Problem Formulation. We assume that $V = \{v_1, v_2, \cdots v_i, \cdots v_K\}$ denotes the video tasks that complexity between C_{low} and C_{high}. And $Y = \left\{y_1, y_2, \cdots y_j, \cdots y_S\right\}$ is the set of servers, where y_S denotes the cloud server. For different input tasks, the appropriate server can be selected for inference according to the current resource situation. Real-time processing of video tasks usually requires low delay and costs a lot of energy consumption [12]. Therefore, our goal is to satisfy the delay requirements while minimizing the processing energy consumption and guaranteeing accuracy. In addition, the accuracy requirements for most of the tasks are already satisfied in the complexity-aware analysis module. We define the utility function as the weighted difference between accuracy and energy consumption, incorporating the delay as a constraint in the optimization problem formulation, which can be expressed as follows:

$$\begin{aligned} & \max \quad \tfrac{1}{T}\textstyle\sum_{i=t}^{T}\left(A_t - \omega E_t\right) \\ \text{s.t.} \quad & C_1 : D_{i,t} \le D_{i,t}^{q}, i \in \{1, 2, \cdots K\}, t \in T \\ & C_2 : \textstyle\sum_{j=1}^{S} z_{i,t}^{j} = 1, z_{i,t}^{j} = \{0, 1\}, y_j \in Y, t \in T \\ & C_3 : \textstyle\sum_{i=1}^{K} U_{i,t} \le \boldsymbol{\mathcal{U}}_t, t \in T \end{aligned} \tag{5}$$

where A_t and E_t are the accuracy and energy consumption at time slot t. $D_{i,t}$ is the delay of task i and $D_{i,t}^{q}$ is the delay requirement. $z_{i,t}^{j}$ indicates whether the i-th task is assigned to the j-th server, and $U_{i,t}$ is the resource consumption of task i and $\boldsymbol{\mathcal{U}}_t$ is the available resource of servers. To address the problem that multiple types of resources are difficult to coordinate uniformly [32], we build a heterogeneous resource metric model: $U^{\text{tri}} = U^{\text{comp}}(v) \cdot U^{\text{comm}}(v) \cdot U^{\text{cach}}(v)$. The metric U^{tri} is the resource pool of integration to computing, communication, and caching. The weighting parameter ω is used to control the tradeoff between accuracy and energy consumption. Constraint C_1 ensures that the delay requirements of each task are met. Otherwise, it will be assigned to a more resourceful server. Constraint C_2 ensures that only one server is selected for tasks at each time slot t. Constraint C_3 means the resource by all tasks is less than or equal to the available resource.

4.3.2 Solution Algorithm. Since the constraints include integer vectors and continuous vectors, the objective function in Eq. (5) is a combinatorial optimization problem. The assignment of the optimal resource solution for each task has been proven to be an NP-hard problem [4], necessitating the utilization of a heuristic algorithm for its resolution. Inspired by the grey wolf optimizer in [28], we conceptualize the aforementioned combinatorial optimization problem as a collaborative foraging process within a gray wolf pack. Accordingly, we propose an adaptive resource-aware optimization algorithm to obtain approximate solutions. The optimization procedure is outlined in Algorithm 1.

Algorithm 1: Adaptive Resource-Aware Optimization Algorithm

Input:
The set of tasks V, The set of servers Y;
Output:
The resource allocation scheme Z^*;
Procedure:
Initialize Q, δ, B, and M
  for each search task
    Calculate average fitness function $\bar{F}(t)$
    Calculate $D_{i,t}$ from the first three best $\vec{Q}$
    if $D_{i,t} \le D_{i,t}^{q}$ **then**
      $Z^* \leftarrow z_{i,t}^{j} = 1$
    else
      Updata δ, $\vec{B}$, and $\vec{M}$, $t = t + 1$
    end if
    while $A_{i,t} < A_{i,t}^{q}$ **do**
      $y_j = y_K$
    end while
  end for
return Z^*;

The key idea of the adaptive resource-aware optimization algorithm is to use the optimization history to dynamically adjust the strategy. It adjusts the optimization parameters mainly based on the behavior in previous iterations and the average fitness value. The average fitness is calculated as shown below:

$$\bar{F}(t) = \frac{\sum_{t \in T} F(t)}{T} = \frac{\sum_{t \in T} A_t + \omega E_t}{T} \tag{6}$$

Consider the matching relationship between task requirements and available resources, it employs a resource-aware strategy to assign the task to the most suitable server for processing. First, it needs to quickly identify the server that can meet the delay requirement and then determine whether the accuracy requirement is met. If the edge server cannot meet the accuracy requirements of the task, it is uploaded to the cloud server for processing. Among the acceptable policies, the algorithm selects the solution with the lowest energy consumption and returns the result. Specifically, to better find the optimal solution, we introduce an adaptive factor $\delta = 2 - 2\frac{1}{e-1}\left(e^{\frac{t}{itermax}} - 1\right)$. t is the current number of iterations and $itermax$ is the maximum number of iterations. In the optimization process, the behavior of the search target is defined as follows:

$$\vec{G} = \left| \vec{M} \cdot \vec{Q}_p(t) - \vec{Q}(t) \right| \tag{7}$$

where $\vec{M}$ is a vector of random coefficients between [0,2], $\vec{Q}_p$ is the position vector of the target, and $\vec{Q}$ denotes the current position vector. The updated formula for the position vector is as follows:

$$\vec{Q}(t+1) = \vec{Q}_p(t) - \vec{B} \cdot \vec{G} \tag{8}$$

where $\vec{B} = 2\vec{\delta} \cdot \vec{r}_1 - \vec{\delta}$, r_1 is a random number between [0,1]. The optimal resource allocation solution is eventually found through iterative optimization.

In the above algorithm, a resource allocation scheme needs to be determined for each task. In each round of iteration, the average fitness value is updated based on historical information. Then the solution with the highest fitness value is selected as the current resource allocation result, which requires K individual operations. In the process of finding the optimal resource allocation scheme, the execution of each task also needs to be considered. Therefore, the complexity of the algorithm is $O(KS)$.

5 PERFORMANCE EVALUATION

In this section, we conduct extensive experiments to evaluate the performance of the proposed JAVP architecture. We first describe the specific setup of the experiments and then analyze the advantages of our solution over the traditional cloud-only method. Finally, we compare the performance of our method with several baseline methods with edge-cloud collaboration.

5.1 Evaluation Setup

5.1.1 Datasets and Implementation Details. To evaluate our solution, we conduct object detection experiments on the UA-DETRAC dataset[37] and DAWN dataset[16], with selected detection objects including cars, buses, and persons. The UA-DETRAC dataset is video clips collected from surveillance cameras at traffic intersections, with videos recorded at 25 frames per second. The DAWN dataset consists of real-world images collected under a variety of adverse weather conditions. It emphasizes diverse traffic environments, and these data are divided into four groups of weather conditions: fog, rain, sand, and snow.

We use one NVIDIA A100 GPU with 40 GB memory as the cloud server and four NVIDIA Jetson Xavier NX with 8 GB memory as edge servers. We use YOLOv5 as a DNN model, which is an open-source project widely used for object detection. Specifically, we directly use the pre-trained model on the COCO dataset [24]. Then, two DNN models of different sizes are deployed on the edge server and cloud server, refer to Section 3. According to the experiment setup in [34], the energy consumption per unit of transmission γ_i is 0.5×10^{-5} J. In addition, all the above experiments are performed under the environments of Ubuntu 18.04.3, Python 3.8.13, and PyTorch 1.11.0.

5.1.2 Evaluation Metrics. We use the following metrics to evaluate the performance of different methods.

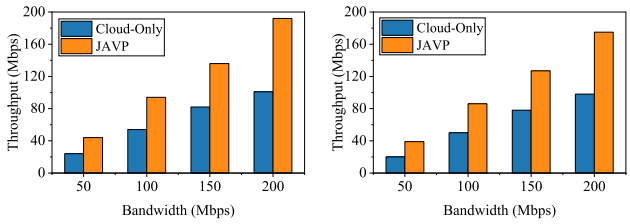
- **Accuracy:** We evaluate the accuracy by the standard metric mAP_{50} in object detection, which is the mean average precision of multiple objects when the threshold of IoU is greater than 0.5.
- **Delay:** This includes complexity-aware analysis delay, transmission delay, and inference delay. We get the total delay by measuring the time between task input and the completion of DNN inference.
- **Energy consumption:** We calculate the inference energy consumption by multiplying the server power by the inference time of the task, and the transmission energy consumption by multiplying γ_i by the size of the task.

5.1.3 Baseline Methods. We compare our solution with the following three baseline methods.

- **RDAP [33]:** This is a resource deployment method based on edge-cloud collaboration, which can optimize server resource allocation and task scheduling by task aggregation and delay thresholds.
- **DAO[29]:** This a dynamic adaptive offloading method for video analysis. Specifically, it improves video processing efficiency by adjusting the video bitrate and resolution.
- **Sniper[27]:** This is an edge-cloud collaborative inference scheduling method based on time-aware. It uses a non-intrusive performance characterization network (PCN) to predict the inference time of a DNN and guides task scheduling.

5.2 Evaluation Results

5.2.1 Joint-Aware Advantage Analysis. To validate the advantages of joint-aware analysis, we first run the proposed multiscale feature fusion model on the training set and get the complexity thresholds C_{low} and C_{high}. The range of values for the task accuracy requirements is [0.5,0.8]. Then the throughput under different bandwidths is compared on the test set, as shown in Figure 6. It can be seen that JAVP can improve the throughput by more than 70% compared to the traditional cloud-only solution. This result is in line with our expectations, as it can reduce the waste of resources caused by uploading simple tasks to the cloud server for processing. The accuracy of the complexity analysis is critical. Therefore, we evaluate the performance of the complexity analysis module indirectly through the task success rate. We show the success rates for meeting the task accuracy requirements in Table 1. We observe that the cloud-only solution can meet the accuracy requirements of 93% and 85% of tasks on different datasets. And JAVP achieves a success rate close to the cloud-only solution, which can reach 91% and 84%.



(a) UA-DETRAC (b) DAWN

Figure 6: Throughput analysis with different datasets.

These results show that JAVP can effectively improve the efficiency of edge-cloud collaborative video processing through joint-aware.

Table 1: Success rates for meeting the accuracy requirements

Dataset	Cloud-Only	JAVP
UA-DETRAC	93%	91%
DAWN	85%	84%

5.2.2 Accuracy-Energy Consumption Tradeoff. We compare our proposed architecture with some baseline methods based on edge-cloud collaboration. We first investigate the accuracy-energy consumption tradeoff optimization results of JAVP with different datasets. We assume that a camera is responsible for a type of task (e.g., monitoring different traffic areas). Then the images in the dataset are assigned to all cameras for simulating the task input. Since different tasks in real scenarios usually have different requirements for accuracy, the accuracy requirements of the input task are chosen randomly from [0.5,0.8].

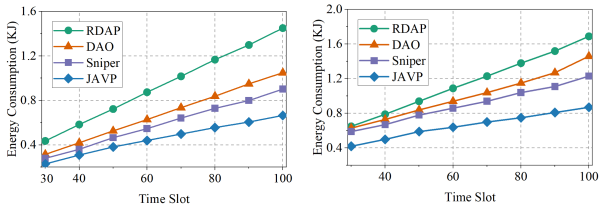
Table 2: Accuracy comparison results of UA-DETRAC

Detection Objects	RDAP	DAO	Sniper	JAVP
Cars	64.58%	67.94%	67.91%	**68.33%**
Buses	60.66%	64.78%	63.44%	**65.70%**
Persons	58.31%	62.58%	61.47%	**63.42%**

The experimental comparison results of accuracy under different datasets are reported in Table 2 and Table 3. It can be found that the accuracy of JAVP is higher than other methods in multiple detection objects and different weather. Among them, DAO also achieves good accuracy due to the optimization of the data resolution. In contrast, RADP and Sniper focus only on bandwidth and energy consumption, so they result in lower accuracy rates. Figure 7 shows the results of energy consumption. Compared with other methods, JAVP has the lowest energy consumption. And its advantage is more obvious as the time slot increases. Sniper also exhibits favorable energy consumption characteristics due to resource optimization focused on edge servers. Overall, the results show that JAVP can achieve the tradeoff optimization of accuracy and energy consumption through joint-aware and edge-cloud collaboration.

Table 3: Accuracy comparison results of DAWN

Detection Objects		RDAP	DAO	Sniper	JAVP
Fog	Cars	62.03%	65.81%	65.44%	**65.78%**
	Buses	59.12%	62.17%	61.06%	**63.26%**
	Persons	57.24%	60.14%	59.17%	**61.03%**
Rain	Cars	63.14%	65.83%	65.72%	**66.71%**
	Buses	59.15%	63.31%	61.89%	**64.23%**
	Persons	57.34%	61.11%	60.03%	**61.95%**
Sand	Cars	62.24%	65.36%	65.23%	**65.82%**
	Buses	58.89%	62.03%	61.58%	**63.07%**
	Persons	56.88%	60.32%	60.04%	**60.91%**
Snow	Cars	62.47%	65.35%	65.12%	**66.29%**
	Buses	58.73%	62.64%	61.38%	**63.67%**
	Persons	57.22%	60.87%	60.59%	**61.43%**

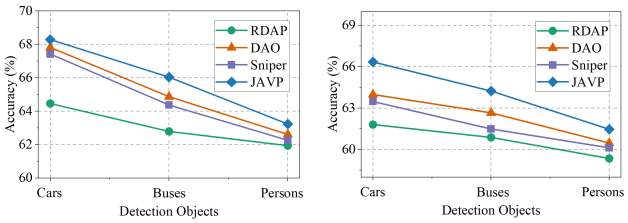


(a) UA-DETRAC (b) DAWN

Figure 7: Energy consumption analysis of different methods.

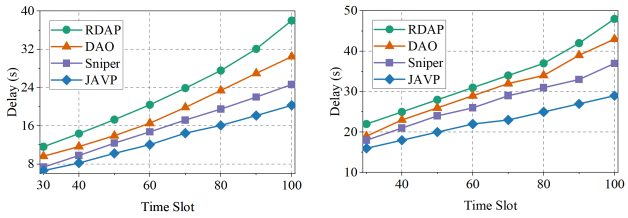
5.2.3 Accuracy-Delay Tradeoff. To test the optimization results of the tradeoff between accuracy and delay, we set the range of delay requirements to [0.2,0.6]. The accuracy and delay are optimized by increasing the value of the weight parameter ω. Since there are multiple types of weather in the DAWN dataset, the results in Figure 8 average the accuracy under different weather. Although we don't limit the accuracy requirements of the task in this test. We still observe that JAVP outperformed RDAP, DAO, and Sniper in multiple detection objects. And it achieves good performance on both datasets. In contrast, other methods perform poorly on the DAWN dataset. This is because they lack the awareness of complex scenarios and cannot adaptively adjust the resource allocation to ensure the performance of dynamic scenarios. Next, we show the comparison results for delay in Figure 9, where JAVP achieves the lowest delay. We also find that the advantage of JAVP is more pronounced as the time slot increases. The reason why JAVP can achieve lower delay and higher accuracy is that the dynamic allocation of resources can have a significant impact on the delay for task processing. The resource-aware module in JAVP can dynamically adjust the resource allocation according to the delay requirement and improve the accuracy at the same time.

5.2.4 Accuracy-Delay-Energy Consumption Tradeoff. In this experiment, we impose restrictions on both the accuracy requirements and the delay requirements. And the accuracy requirement is referenced in Section 5.2.2 and the delay requirement is referenced in



(a) UA-DETRAC (b) DAWN

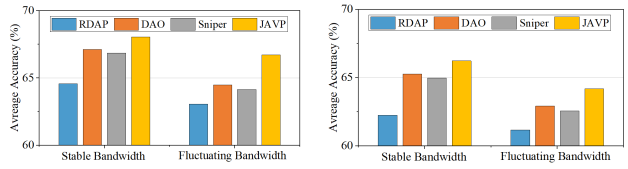
Figure 8: Accuracy analysis of different methods.



(a) UA-DETRAC (b) DAWN

Figure 9: Delay analysis of different methods.

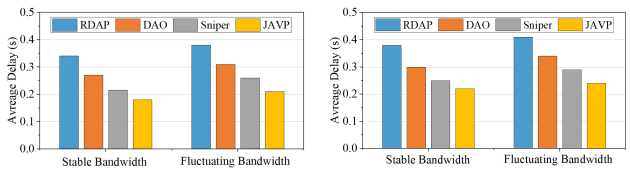
Section 5.2.3. Dynamic networks in real-world scenarios can have a significant impact on the results [40]. To further evaluate the performance of JAVP, we experimented with the above method in stable and fluctuating bandwidths. The stable bandwidth is configured at 100 Mbps, while the fluctuating bandwidth is set within a range of 20%. The accuracy comparison results under different bandwidth environments are shown in Figure 10. We can see the accuracy of JAVP is always higher than other methods. Compared with RDAP, JAVP can improve average accuracy by more than 5%. And JAVP can improve average accuracy by more than 3% compared with DAO and Sniper. This is because JAVP can improve the accuracy by adaptively adjusting the edge or cloud server.



(a) UA-DETRAC (b) DAWN

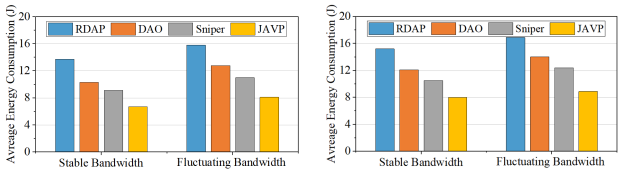
Figure 10: Average accuracy comparison results of different methods.

Figure 11 and Figure 12 show the comparison results of delay and energy consumption in different bandwidth environments, respectively. JAVP is well adapted to both complex scenarios and dynamic networks, consistently showing lower delay and energy consumption than other methods. On average, JAVP has a 16% lower delay in video processing than Sniper, 30% lower than DAO, and



(a) UA-DETRAC (b) DAWN

Figure 11: Average delay comparison results of different methods.



(a) UA-DETRAC (b) DAWN

Figure 12: Average energy consumption comparison results of different methods.

43% lower than RDAP. In addition, the average energy consumption of JAVP is 26%, 35%, and 50% lower than that of Sniper, DAO, and RDAP, respectively. Overall, JAVP can reduce energy consumption and ensure accuracy through complexity-aware analysis. It can also ensure delay through resource-aware optimization. Finally, the tradeoff between accuracy, delay, and energy consumption is realized under resource-constrained conditions.

6 CONCLUSION

In this paper, we introduce JAVP, the first video processing architecture that supports both joint-aware and edge-cloud collaboration. Thanks to efficient multiscale complexity-aware analysis and adaptive resource-aware analysis, JAVP can estimate the difficulty and resources of a video task before it is offloaded, and then select the best server to process. The core design of JAVP is to optimize the tradeoff between accuracy, delay, and energy consumption through joint-aware and edge-cloud collaboration. Experimental results demonstrate that JAVP has significant improvement in video processing efficiency for edge-cloud collaboration compared to state-of-the-art solutions. Future work will focus on the edge-cloud collaboration approach based on the content-aware and adaptive video parameters configuration.

7 ACKNOWLEDGMENTS

This work is supported by the Beijing Natural Science Foundation (L221004), the National Key R&D Program of China (2022YFF0902403, 2022YFE0125400), the National Natural Science Foundation of China (62072440), and the Shenzhen Science and Technology Program (Grant No. RCYX20200714114523079, JCYJ20220818101014030). Corresponding author is Wen Ji.

REFERENCES

[1] 2018. *Unfolding the Industry Blueprint of an Intelligent World.* https://www.huawei.com/minisite/giv/Files/whitepaper_en_2018.pdf

[2] 2022. *YOLOv5: The friendliest AI architecture you'll ever use.* https://ultralytics.com/yolov5

[3] Andrey A Bagrov, Ilia A Iakovlev, Askar A Iliasov, Mikhail I Katsnelson, and Vladimir V Mazurenko. 2020. Multiscale structural complexity of natural patterns. *Proceedings of the National Academy of Sciences (PNAS)* 117, 48 (2020), 30241–30251.

[4] Pierre Bonami, Lorenz T Biegler, Andrew R Conn, Gérard Cornuéjols, Ignacio E Grossmann, Carl D Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, et al. 2008. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete optimization* 5, 2 (2008), 186–204.

[5] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G Andersen, Michael Kaminsky, and Subramanya Dulloor. 2019. Scaling video analytics on constrained edge nodes. *Proceedings of Machine Learning and Systems* 1 (2019), 406–417.

[6] Bo Chen, Zhisheng Yan, and Klara Nahrstedt. 2022. Context-aware image compression optimization for visual analytics offloading. In *Proceedings of the 13th ACM Multimedia Systems Conference*. 27–38.

[7] Jiasi Chen and Xukan Ran. 2019. Deep learning with edge computing: A review. *Proc. IEEE* 107, 8 (2019), 1655–1674.

[8] Zhiming Chen, Kean Chen, Weiyao Lin, John See, Hui Yu, Yan Ke, and Cong Yang. 2020. Piou loss: Towards accurate oriented object detection in complex environments. In *European Conference on Computer Vision (ECCV)*. 195–211.

[9] Nicola Conci, Francesco GB De Natale, Stefano Messelodi, Carla Maria Modena, Marco Verza, and Roberto Fioravanti. 2016. An integrated framework for video surveillance in complex environments. In *2016 IEEE International Smart Cities Conference (ISC2)*. IEEE, 1–6.

[10] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. 2020. Server-driven video streaming for deep learning inference. In *ACM Special Interest Group on Data Communication (SIGCOMM)*. 557–570.

[11] Sijing Duan, Dan Wang, Ju Ren, Feng Lyu, Ye Zhang, Huaqing Wu, and Xuemin Shen. 2022. Distributed Artificial Intelligence Empowered by End-Edge-Cloud Computing: A Survey. *IEEE Communications Surveys & Tutorials* 25, 1 (2022), 591–624.

[12] Biyi Fang, Xiao Zeng, and Mi Zhang. 2018. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Annual International Conference on Mobile Computing and Networking (MobiCom)*. 115–127.

[13] Philipp M Grulich and Faisal Nawab. 2018. Collaborative edge and cloud neural networks for real-time video processing. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2046–2049.

[14] Qing Han, Xuebin Ren, Peng Zhao, Yimeng Wang, Luhui Wang, Cong Zhao, and Xinyu Yang. 2023. ECCVideo: A Scalable Edge Cloud Collaborative Video Analysis System. *IEEE Intelligent Systems* 38, 1 (2023), 34–44.

[15] Qing Han, Shusen Yang, Xuebin Ren, Cong Zhao, Jingqi Zhang, and Xinyu Yang. 2020. OL4EL: online learning for edge-cloud collaborative learning on heterogeneous edges with resource constraints. *IEEE Communications Magazine* 58, 5 (2020), 49–55.

[16] Mahmoud Hassaballah, Mourad A Kenk, Khan Muhammad, and Shervin Minaee. 2020. Vehicle detection and tracking in adverse weather using a deep learning framework. *IEEE Transactions on Intelligent Transportation Systems* 22, 7 (2020), 4230–4242.

[17] Jin Huang, Colin Samplawski, Deepak Ganesan, Benjamin Marlin, and Heesung Kwon. 2020. Clio: Enabling automatic compilation of deep learning pipelines across iot and cloud. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom)*. 1–12.

[18] Samvit Jain, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, and Joseph Gonzalez. 2019. Scaling video analytics systems to large camera deployments. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. 9–14.

[19] Hyuk-Jin Jeong, Hyeon-Jae Lee, Chang Hyun Shin, and Soo-Mook Moon. 2018. IONN: Incremental offloading of neural network computations from mobile devices to edge servers. In *Proceedings of the ACM Symposium on Cloud Computing*. 401–411.

[20] Wen Ji, Bing Liang, Yuqin Wang, Rui Qiu, and Zheming Yang. 2020. Crowd v-ioe: Visual internet of everything architecture in ai-driven fog computing. *IEEE Wireless Communications* 27, 2 (2020), 51–57.

[21] Maciej Koch-Janusz and Zohar Ringel. 2018. Mutual information, neural networks and the renormalization group. *Nature Physics* 14, 6 (2018), 578–582.

[22] Min Li, Yu Li, Ye Tian, Li Jiang, and Qiang Xu. 2021. AppealNet: An Efficient and Highly-Accurate Edge/Cloud Collaborative Architecture for DNN Inference. In *ACM/IEEE Design Automation Conference (DAC)*. 409–414.

[23] Rui Li, Zhi Zhou, Xu Chen, and Qing Ling. 2019. Resource price-aware offloading for edge-cloud collaboration: A two-timescale online control approach. *IEEE Transactions on Cloud Computing* 10, 1 (2019), 648–661.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*. 740–755.

[25] Ruoyang Liu, Lu Zhang, Jingyu Wang, Huazhong Yang, and Yongpan Liu. 2021. PETRI: Reducing Bandwidth Requirement in Smart Surveillance by Edge-Cloud Collaborative Adaptive Frame Clustering and Pipelined Bidirectional Tracking. In *ACM/IEEE Design Automation Conference (DAC)*. 421–426.

[26] Shengzhong Liu, Tianshi Wang, Jinyang Li, Dachun Sun, Mani Srivastava, and Tarek Abdelzaher. 2022. AdaMask: Enabling Machine-Centric Video Streaming with Adaptive Frame Masking for DNN Inference Offloading. In *Proceedings of the 30th ACM International Conference on Multimedia (MM)*. 3035–3044.

[27] Weihong Liu, Jiawei Geng, Zongwei Zhu, Jing Cao, and Zirui Lian. 2022. Sniper: cloud-edge collaborative inference scheduling with neural network similarity modeling. In *ACM/IEEE Design Automation Conference (DAC)*. 505–510.

[28] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. 2014. Grey wolf optimizer. *Advances in Engineering Software* 69 (2014), 46–61.

[29] Taslim Murad, Anh Nguyen, and Zhisheng Yan. 2022. DAO: Dynamic Adaptive Offloading for Video Analytics. In *Proceedings of the 30th ACM International Conference on Multimedia (MM)*. 3017–3025.

[30] Ju Ren, Deyu Zhang, Shiwen He, Yaoxue Zhang, and Tao Li. 2019. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Computing Surveys (CSUR)* 52, 6 (2019), 1–36.

[31] Dirk Semkat, Dietrich Kremp, and Michael Bonitz. 1999. Kadanoff-Baym equations with initial correlations. *Physical Review E* 59, 2 (1999), 1557.

[32] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. 2018. Legoos: A disseminated, distributed OS for hardware resource disaggregation. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 69–87.

[33] Mingfeng Su, Guojun Wang, and Kim-Kwang Raymond Choo. 2022. Prediction-Based Resource Deployment and Task Scheduling in Edge-Cloud Collaborative Computing. *Wireless Communications and Mobile Computing* 2022 (2022), 1–17.

[34] Can Wang, Sheng Zhang, Yu Chen, Zhuzhong Qian, Jie Wu, and Mingjun Xiao. 2020. Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics. In *IEEE Conference on Computer Communications (INFOCOM)*. 257–266.

[35] Shibo Wang, Shusen Yang, and Cong Zhao. 2020. SurveilEdge: Real-time video query based on collaborative cloud-edge deep learning. In *IEEE Conference on Computer Communications (INFOCOM)*. 2519–2528.

[36] Xuezhi Wang, Guanyu Gao, Xiaohu Wu, Yan Lyu, and Weiwei Wu. 2022. Dynamic DNN model selection and inference off loading for video analytics with edge-cloud collaboration. In *Proceedings of the 32nd Workshop on Network and Operating Systems Support for Digital Audio and Video*. 64–70.

[37] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. 2020. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding* 193 (2020), 1–27.

[38] Zheming Yang, Bing Liang, and Wen Ji. 2021. An intelligent end–edge–cloud architecture for visual IoT-assisted Healthcare systems. *IEEE Internet of Things Journal* 8, 23 (2021), 16779–16786.

[39] Shuochao Yao, Jinyang Li, Dongxin Liu, Tianshi Wang, Shengzhong Liu, Huajie Shao, and Tarek Abdelzaher. 2020. Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 476–488.

[40] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A Lee. 2018. Awstream: Adaptive wide-area streaming analytics. In *ACM Special Interest Group on Data Communication (SIGCOMM)*. 236–252.

[41] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. 2017. Live video analytics at scale with approximation and delay-tolerance. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 377–392.

[42] Rui-Xiao Zhang, Changpeng Yang, Xiaochan Wang, Tianchi Huang, Chenglei Wu, Jiangchuan Liu, and Lifeng Sun. 2022. AggCast: Practical Cost-effective Scheduling for Large-scale Cloud-edge Crowdsourced Live Streaming. In *Proceedings of the 30th ACM International Conference on Multimedia (MM)*. 3026–3034.

[43] Tan Zhang, Aakanksha Chowdhery, Paramvir Bahl, Kyle Jamieson, and Suman Banerjee. 2015. The design and implementation of a wireless video surveillance system. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom)*. 426–438.

[44] Zimu Zheng, Yunzhe Li, Han Song, Lanjun Wang, and Fei Xia. 2022. Towards Edge-Cloud Collaborative Machine Learning: A Quality-aware Task Partition Framework. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM)*. 3705–3714.

[45] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. 2019. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* 107, 8 (2019), 1738–1762.