Novas: Tackling Online Dynamic Video Analytics with Service Adaptation at Mobile Edge Servers

Liang Zhang, *Student Member, IEEE,* Hongzi Zhu, *Senior Member, IEEE,* Wen Fei, *Student Member, IEEE,* Yunzhe Li, *Student Member, IEEE,* Mingjin Zhang, *Student Member, IEEE,* Jiannong Cao, *Fellow, IEEE,* Minyi Guo, *Fellow, IEEE*

***Abstract*—Video analytics at mobile edge servers offers significant benefits like reduced response time and enhanced privacy. However, guaranteeing various quality-of-service (QoS) requirements of dynamic video analysis requests on heterogeneous edge devices remains challenging. In this paper, we propose a scalable online video analytics scheme, called Novas, which automatically makes precise service configuration adjustments upon constant video content changes. Specifically, Novas leverages the filtered confidence sum and a two-window t-test to online detect accuracy fluctuations without ground truth information. In such cases, Novas efficiently estimates the performance of all potential service configurations through a singular value decomposition (SVD)-based collaborative filtering method. Finally, given the NP-hardness of the optimal scheduling problem, a heuristic scheduling strategy that maximizes the minimum remaining resources is devised to schedule the most suitable configurations to servers for execution. We evaluate the effectiveness of Novas through extensive hybrid experiments conducted on a dedicated testbed. Results show that Novas can achieve a substantial over 27× improvement in satisfying the accuracy requirements compared with existing methods adopting fixed configurations, while ensuring latency requirements. Moreover, Novas improves the goodput of the system by an average of 37.86% compared to existing state-of-the-art scheduling solutions.**

***Index Terms*—video analytics, edge computing, adaptive configuration, task scheduling**

I. Introduction

RECENT years have witnessed advances in Artificial Intelligence of Things (AIoT), a new computing paradigm seamlessly integrating AI and IoT technologies. Many advanced AIoT applications, such as autonomous driving, unmanned aerial vehicles (UAVs), virtual reality (VR), and surveillance cameras, pose a huge surge of online dynamic video analysis requests, such as object detection with deep neural network (DNN) models on each frame of a video, with stringent accuracy and latency requirements. Additionally, such requests may involve privacy concerns, rendering it imperative to conduct video analysis locally at mobile edge servers (MESs) [1], referred to as the *online edge video analytics problem*. These MESs comprise a combination of dedicated and non-dedicated heterogeneous devices, such as base stations (BSs) or roadside units (RSUs) [2], and onboard computing units on smart vehicles [3]. Such a trend of online edge video analytics is further reinforced with the soaring development of hardware and wideband wireless communication technology.

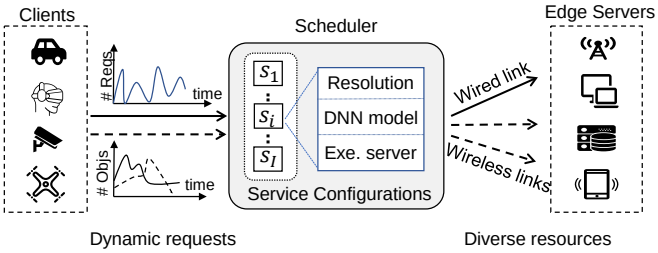


Fig. 1. An illustration of dynamic video analytics executed on heterogeneous edge servers. The service configuration specifying the required video resolution, the selected DNN model and the execution server should adaptively change based on the content of one video to meet QoS requirements.

As illustrated in Fig. 1, a practical online edge video analytics scheme should fulfill three essential requirements as follows. First, the scheme should guarantee various end-to-end latency requirements; otherwise, results may be out of date for time-sensitive applications. Second, the scheme should incorporate adaptive configuration mechanisms to guarantee analysis accuracy of dynamic requests whose video content dramatically varies over time. Last, the scheme should achieve scalable scheduling for video requests, aiming to maximize system throughput and resource efficiency, even in the presence of multi-dimensional heterogeneous resources and diverse requirement distributions.

In the literature, existing online edge video analytics solutions mainly focus on either video preprocessing at clients or elastic resource allocation at edge servers to guarantee quality of service (QoS) at the minimized cost. The mainstream of video preprocessing methods [4]–[10], relying on offline performance profiling for resolution and frame sampling rate adjustment, may lead to unsatisfied analysis accuracy or intolerable delays when dealing with fast-changing video contents. Another direction of preprocessing methods selects crucial frames or key regions within each frame [11]–[14], requiring substantial computing power on client devices to execute filtering models. In contrast, elastic resource allocation

Manuscript received 2 December 2023; revised 2 April 2024; accepted 18 May 2024. This work was supported in part by National Key Research and Development Program of China under Grant No. 2022YFB4501400, Hong Kong RGC General Research Fund under Grant No. 15204921 and Innovation and Technology Fund - Mainland-Hong Kong Joint Funding Scheme under Grant No. MHP/013/21. *(Corresponding authors: Hongzi Zhu.)*

Liang Zhang, Hongzi Zhu, Wen Fei, Yunzhe Li, Minyi Guo are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China (e-mail: zhangliang@sjtu.edu.cn, fw.key@sjtu.edu.cn, yunzhe.li@sjtu.edu.cn, hongzi@cs.sjtu.edu.cn, guo-my@cs.sjtu.edu.cn).

Mingjin Zhang and Jiannong Cao are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: csmzhang@comp.polyu.edu.hk, csjcao@comp.polyu.edu.hk)

methods like [5] and [7] adjust the resource usage based on the difference in demand between video analytics tasks. Moreover, these methods often assume a fixed number of video requests with pre-determined QoS requirements, limiting their capability to handle unknown dynamic video analysis requests.

In this paper, we propose a scalable online edge video analytics scheme, called *Novas*, which is devoted to dynamic video analytics on a group of interconnected heterogeneous edge servers. The core idea of Novas is to online adapt the service configuration (*e.g.*, a triple of the selected video resolution, the DNN inference model to use, and the designated server of execution) to the video content to meet the QoS requirements of each video analysis request, while preserving the diversity of resources as much as possible for future unknown requests. More specifically, to accommodate the dynamic nature of mobile videos, Novas simultaneously detects significant content changes that may be mismatched with the current service configuration. When such changes occur, or new video requests arrive, Novas online predicts the performance of all service configurations based on a small amount of configuration profiling on the following a few frames. The predicted performance information is then used by the scheduler to make the best service configuration decision.

Three main challenges are encountered. First, it is not straightforward to perceive an out-of-date service configuration for a dynamic video, due to the lack of ground truth information. To deal with this challenge, we leverage the sum of filtered inference confidence of detected objects on each frame as an indication of inference accuracy of the corresponding DNN model. We have one key observation that the filtered confidence sum is positively correlated with accuracy. Furthermore, both the accuracy and the filtered confidence sum of frames with similar contents follow a Gaussian distribution. Therefore, we develop a simple yet effective two-window t-test method to identify severe accuracy fluctuations and promptly notify the scheduler to adjust service configuration to prevent any potential accuracy violations.

Second, it is challenging to online obtain the performance of vast service configurations regarding the ever-changing video contents. It is computationally infeasible to measure the performance of each service configuration for each video frame. To tackle this challenge, Novas incorporates both an offline profiling and an online sampling methods to derive a sparse performance matrix. Particularly, only a small number of sampled service configurations are measured on a few new frames. Then, the singular value decomposition (SVD)-based collaborative filtering and the stochastic gradient descent (SGD) techniques are employed to estimate the missing values in the matrix, obtaining the complete performance information about all configurations for the new video contents.

Last, it is hard to achieve the maximized system throughput in scheduling video analysis requests with diverse QoS requirements on heterogeneous edge servers. This scheduling problem can be formulated as an NP-hard optimization problem with discrete constraints and variables. To handle this challenge and optimize the system's scalability for unknown incoming requests, we propose a heuristic scheduling strategy that prioritizes preserving resource diversity. Specifically, this strategy groups all servers according to their heterogeneity and allocate requests to servers that maximize the minimum remaining resources across all groups. The scheduling process does not rely on specific heterogeneous characteristics, which contributes to the stability and robustness of this strategy in various heterogeneous environments.

We implement Novas on a testbed consisting of five edge devices, *i.e.*, one desktop, two Jetson NX boards, and two Jetson Nano boards, and evaluate it on three representative video analytics workloads. The results show that the average service configuration prediction error is less than 10%, and the recall value of online anomaly detection can reach above 90% in experiments with different accuracy fluctuation ranges. Compared with existing solutions, Novas improves system throughput up to 37.86% on average and achieves 27× improvement in satisfying the accuracy requirements while ensuring latency requirements of all requests.

We highlight the main contributions made in this paper as follows: 1) A neat two-window t-test mechanism is devised to online detect video content change with no ground truth information. 2) A novel service configuration performance prediction method is proposed, using SVD-based collaborative filtering techniques. 3) A heuristic scheduling algorithm is proposed to maximize system throughput and scalability by preserving the diversity of heterogeneous resources.

II. System model and Problem Formulation

A. System Model

As shown in Fig. 1, we consider three types of entities in an online edge video analytics system as follows:

- **Clients:** are end devices, ranging from battery-powered mobile devices such as drones, smartphones, and VR/AR devices to fix-deployed devices such as surveillance cameras. Clients submit video analysis requests to the system via wired/wireless links of various bandwidth. Such requests arrive at different times and have different end-to-end latency and inference accuracy requirements.
- **Scheduler:** is an algorithm that resides on a dedicated edge server. The scheduler collects the global information of all video analysis requests and the running-time status of all servers, and determines the most proper service configuration for each video.
- **Servers:** are heterogeneous edge devices, such as dedicated servers deployed at BSs or RSUs and voluntary servers like laptop computers and vehicular computational units. Voluntary servers are free to join or leave the system. These servers are connected to the system via high-bandwidth backbone connections or wireless networks, and are equipped with different amount of memory and computational resources. Moreover, servers download a rich set of pre-trained DNN models of different structure and size before service.

B. Problem Formulation

Our problem focuses on scheduling dynamic video analysis requests to heterogeneous edge servers to guarantee QoS

requirements and maximize resource efficiency. Specifically, for each video analysis request $i \in I$, where I is the set of video analysis requests to be served, let $q_i = (a_i, l_i)$ denotes the QoS requirement of request i, where a_i and l_i are the accuracy and latency requirements, respectively. We aim to find the best service configuration $s_i = (r_i, m_i, n_i)$ for each request i to maximize the number of requests in service (*i.e.*, the goodput of the system) while meeting their QoS requirements, where $r_i \in R$, $m_i \in M$ and $n_i \in N$ are the video resolution, the DNN model, and the allocated execution server selected from the resolution set R, the DNN model set M, and the edge server set N, respectively. We define an indicator function $\mathbb{I}(i) \rightarrow \{0, 1\}$ to represent if request i is put in service by the system. The edge video analytics problem can be formulated as the following optimization problem:

$$\max \sum_{i=1}^{|I|} \mathbb{I}(i) \tag{1}$$

$$s.t. \quad \mathcal{L}^{net}(i, s_i) + \mathcal{L}^{com}(i, s_i) \leq l_i, \forall i \in I \tag{2}$$

$$\mathcal{A}(i, s_i) \geq a_i, \forall i \in I \tag{3}$$

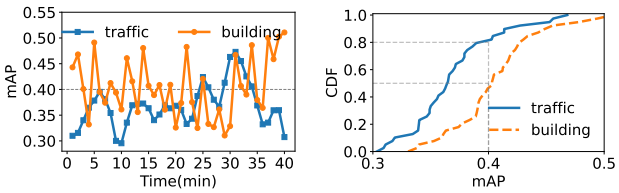
$$\sum_{\forall i, n_i = n} F_i \cdot \mathcal{L}^{net}(i, s_i) \leq 1, \forall n \in N \tag{4}$$

$$\sum_{\forall i, n_i = n} F_i \cdot \mathcal{L}^{com}(i, s_i) \leq 1, \forall n \in N \tag{5}$$

$$\sum_{\forall i, n_i = n} \mathcal{C}(i, s_i) \leq C_n, \forall n \in N \tag{6}$$

where $\mathcal{L}^{net}(i, s_i)$ denotes the latency of transmitting each frame of video i with resolution r_i to server n_i; $\mathcal{L}^{com}(i, s_i)$, $\mathcal{A}(i, s_i)$ and $\mathcal{C}(i, s_i)$ denote the latency, the accuracy and the memory cost of conducting model inference using model m_i on that frame, respectively; F_i is the required frame rate of video i and C_n is the memory capacity of server n. Constraint (2) and (3) represent the latency and accuracy QoS requirements, respectively. Constraint (4), (5) and (6) represent the network, the computing, and the memory resource constraints. Particularly, constraint (4) and (5) practice the concept of *offered load* in queuing theory [15]. The total transmission load and the total computational load for each server n must be less than one to avoid the extra delay caused by queuing.

The above problem is NP-hard in the strong sense, which can be proved by the following deconstruction. First, we need to determine feasible resolution and model configurations for each video request on all servers based on constraints (2) and (3). Then, for a specific feasible resolution and model configuration combination across all videos, the remaining constraints (4)-(6) and the objective construct a scheduling problem. This scheduling problem can be deduced as a multiple multidimensional knapsack problem, where n servers represent n knapsacks; the network, computing, and memory are three dimensions for each knapsack. This problem is non-trivial to solve due to its NP-hardness [16]. Our problem needs to traverse all feasible resolution and model configuration combinations across all videos and solve their corresponding scheduling problems. Therefore, it is even harder. This concludes the proof.



(a) Accuracy fluctuates over time (b) CDFs of inference accuracy

Fig. 2. Examples of accuracy fluctuation observed on different datasets.

In addition to the inherent NP-hardness of the problem, there are two reasons that hinder the solution of the problem. First, it is challenging to obtain real-time parameter information (e.g., $\mathcal{L}^{com}$, $\mathcal{L}^{net}$ and $\mathcal{A}$) for all videos and configurations due to their complex relationships. We discuss video content change and service configurations effects on these performance parameters in Section III and propose a performance predictor in Section IV to estimate these parameters. Second, considering our online scenario where video requests arrive one by one rather than in bulk, we have to make service decisions in a serialized manner. This motivates us to propose a heuristic strategy to obtain an approximate solution in Section IV.

III. Empirical Study

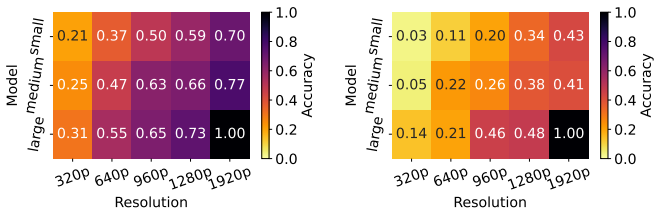
In this section, we study the relationship between different service configurations and the performance of video analytics on three real-world dynamic video datasets as follows:

- **MOT16** [17]: contains twelve 1920×1080@30fps video clips recorded from both static and handheld moving cameras at shopping malls, intersections, and squares.
- **Argoverse** [18]: contains multiple 1920×1200@30fps video clips recorded from in-vehicle cameras in diverse urban outdoor scenes from two US cities.
- **Ekya** [19]: contains traffic video clips recorded from five pole mounted fish-eye cameras at 1280×720@30fps in the city of Bellevue, WA and 24 hours of video clips recorded from a PTZ public camera with a non-stationary view in Las Vegas.

We run three versions of YOLOv8 [20] model of small, medium and large sizes, *i.e.*, YOLOv8n, YOLOv8s and YOLOv8m, respectively, on a Nvidia Jetson NX and a Nvidia Nano devices connected a WiFi AP to detect two types of target objects, *i.e.*, pedestrians and vehicles, on each frame of each video clip.

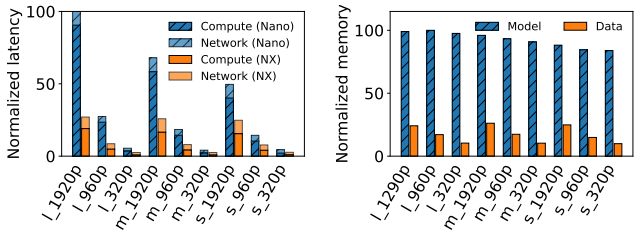
A. Impact of Video Content Change

When the video content dramatically changes over time, such as varying density of target objects or illumination condition, constantly happened in mobile settings, the analysis accuracy of using the same service configuration will inevitably fluctuate. For example, Fig. 2(a) depicts the fluctuation of inference accuracy observed on a 960p Ekya building video and a 640p traffic video over a duration of 40 minutes, respectively. The large YOLOv8m model is used. It can be seen that accuracy severely vibrates on both videos. Moreover, no recognizable pattern regarding the frequency or the amplitude is identified. Furthermore, Fig. 2(b) plots the cumulative



(a) Pedestrian detection (b) Vehicle detection

Fig. 3. Accuracy achieved in different configurations.



(a) Latencies (b) Memory usage

Fig. 4. Latency and memory usage caused in different configurations, *e.g.*, "l_1920p" represents the large model (YOLOv8m) and a resolution of 1920p.

distribution functions (CDFs) of inference accuracy obtained on both videos. Suppose that the accuracy requirement is 0.4, only 20% of frames in the traffic video and 50% of frames in the building video can meet the requirement, using this service configuration. As for a dynamic video, using a service configuration with high video resolution and large DNN model would improve accuracy when dealing with hard content scenarios but leads to huge resource waste when facing simple content scenarios. We have the following observation:

***Observation** 1: Inference accuracy of a specific DNN model varies with changes in video content scenarios. It is optimal for the system to automatically adapt the service configuration to the video content.*

B. Effect of Service Configurations

We measure the average accuracy and latency of detecting pedestrians and vehicles across different configurations in terms of video resolution, DNN model, and execution device. Fig. 3 shows the normalized detection accuracy when detecting pedestrians and detecting vehicles, respectively, both of which show a clear upward trend as the model size and the video resolution increase. In addition, different analysis tasks and different video content scenarios correspond to different performance for the same service configurations. We have the observation as follows:

***Observation** 2: Given a particular video analysis task, a performance model of service configurations can be built with linear regression methods. However, such a model cannot be directly applied to another distinct video analysis task.*

Fig. 4 plots the normalized latency and the memory usage in different service configurations. It can be seen that the Nano costs 3× more time on average to execute model inference than the NX. Due to the large number of service configurations and the diverse computing power of edge servers, it is prohibitive to measure the performance of all service configurations on dynamic videos with fast-changing content scenarios. Moreover, in the scenario where every video analysis request is processed by a YOLOv8m model with a 320p input size, the Nano device can manage 4 video requests, while the NX device can handle 12 video requests, given sufficient network bandwidth and memory. Nonetheless, if the NX device experiences a subpar network connection or has limited memory, it may result in a system throughput reduction even below that of the Nano device. Therefore, we have the observation as follows:

***Observation** 3: The system throughput of the edge video analysis workload is constrained by multi-dimensional resources, such as device computing power, network and memory, and such heterogeneous environment should be fully considered when designing scheduling algorithms.*

IV. Design of Novas

A. Overview

Based on the empirical study in Section III, we design three components for the core scheduler of Novas: a predictor to perform performance estimations, a detector to identify severe accuracy fluctuations, and a controller to make service configuration decisions. In addition, Novas also includes the client handler and the server handler to assist the scheduler in gathering information and performing service configuration. Fig. 5 shows the system architecture of Novas.

Scheduler. When new video analysis requests arrive, the scheduler of Novas performs the following process to make service configuration decisions to meet their performance targets. ① Select a few configurations and collect their performance data from the client handler and server handler. ② Call the *configuration performance predictor* to predict performance information on other configurations and generate performance matrices as the input of the *service configuration controller*. ③ Call the service configuration controller to make service configuration decisions according to a heuristic algorithm and inform the client/server handler to execute them. ④ Call the *content change detector* periodically to identify severe accuracy fluctuations. Once the accuracy violation occurs, the above process ①-③ will be repeated to generate new service configurations to meet all QoS requirements.

Client / Server Handler. The client handler adjusts the video resolution of a video analysis request, establishes a connection with the allocated server according to the service configuration specified for this request, and monitors whether the results meet the QoS requirements. The server handler loads the designated DNN model and performs model inference on the requested video according to the specified service configuration and monitors the task execution information including the inference confidence of results, the computation latency, and the resource usage at this server.

B. Configuration Performance Predictor

Instead of executing performance profiling over all service configurations, the performance predictor combines an offline profiling and an online prediction methods to quickly and accurately obtain performance estimates for all configurations.

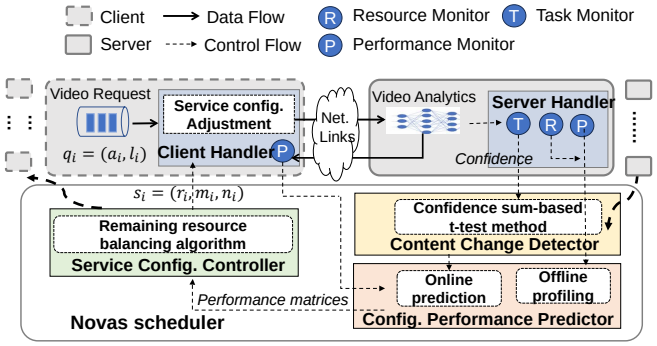


Fig. 5. System architecture of Novas.

1) Offline profiling: Novas offline constructs three performance matrices, denoted as $\mathcal{P}^{net}_{|T|\times|S|}$, $\mathcal{P}^{com}_{|T|\times|S|}$ and $\mathcal{P}^{acc}_{|T|\times|S|}$, to respectively profile the network latency, the computing latency and the inference accuracy over the service configuration set S on a training set of representative video clips T, with each entry $\tau_{i,s}$, for each $i \in T$ and each $s \in S$, represents the measured value $\mathcal{L}^{net}(i,s)$, $\mathcal{L}^{com}(i,s)$ and $\mathcal{A}(i,s)$, respectively.

2) Online prediction: Novas updates the above three performance matrices online as new video requests are received or content changes of in-service videos are detected. Specifically, it appends new rows in each performance matrix for such videos and for each row, it executes a few randomly-selected service configurations to obtain a few performance samples on the row. Then, the SVD-based collaborative filtering technique is used to predict missing performance values in performance matrices. The collaborative filtering technique has been shown to be effective in predicting workload performance [21], [22]. More specifically, SVD is first used to decompose a performance matrix $\mathcal{P} = U \cdot \Sigma \cdot V^T$, where Σ is the matrix of singular values, U and V are the matrices of left and right singular vectors. Let $P^T = \Sigma V^T$ and $Q = U$. Then, SGD iteration is performed over all entries of the reconstructed matrix $\mathcal{P} = QP^T$ until converged. Subsequently, the performance value $\tau_{i,s}$ is modeled as the inner product of latent factor vectors q_s and p_i, i.e., $\hat{\tau}_{i,s} = q_s^T p_i$ ($q_s \in Q$ and $p_i \in P$). The latent factor vectors are learned by minimizing the regularized squared error ϵ on the measured performance matrices:

$$\min_{q^*,p^*} \epsilon = \sum_{\tau_{i,s}\neq 0} \left(\tau_{i,s} - q_s^T p_i\right)^2 + \lambda\left(\|q_s\|^2 + \|p_i\|^2\right) \tag{7}$$

The minimization is performed by a SGD method. For each $\tau_{i,s}$, the SGD update rules are as follows:

$$\begin{aligned} e_{i,s} &\overset{\text{def}}{=} \tau_{i,s} - q_s^T p_i \\ q_s &\leftarrow q_s + \gamma\left(e_{i,s} p_i - \lambda q_s\right) \\ p_i &\leftarrow p_i + \gamma\left(e_{i,s} q_s - \lambda p_i\right) \end{aligned} \tag{8}$$

until ϵ converges. Given p^*, q^* are output vectors of above process, the performance estimation of a new video request i' on a configuration s' will be calculated as $\hat{\tau_{i's'}} = q_{s'}^T p_{i'}$ ($q_{s'} \in q^*$ and $p_{i'} \in p^*$).

C. Content Change Detector

The content change detector proactively detects significant inference accuracy fluctuations caused by video content changes to trigger reconfiguration with QoS guarantee. Due to the lack of ground truth information, it is hard to efficiently identify such content changes. We first introduce an indicator of content change and then present our two-window t-test based detection algorithm.

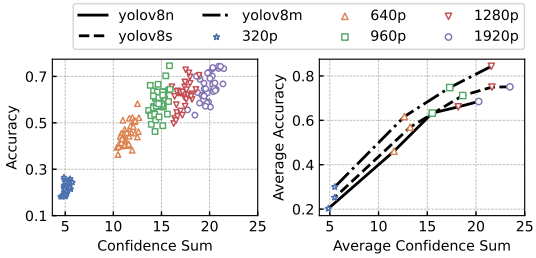


Fig. 6. Left: scatter plot of the inference accuracy and the filtered confidence sum of each frame in an example video clip, analyzed using YOLOv8n in five resolutions. Right: average results of these frames in each resolution using different models.

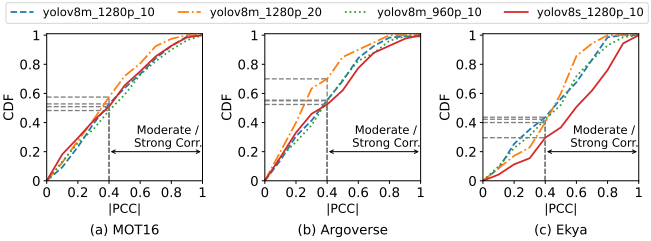


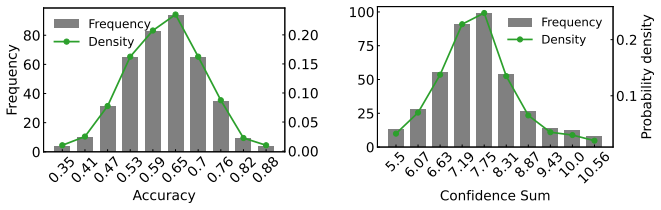
Fig. 7. Pearson correlation coefficients (PCC) between the inference accuracy and the filtered confidence sum across various datasets and configurations. $0.4 < |PCC| \leq 0.6$, $0.6 < |PCC| \leq 0.8$, $0.8 < |PCC| \leq 1$ indicates a moderate, strong, and very strong correlation, respectively.

1) Filtered Confidence Sum: Given the inference output of DNN models on a frame j, we calculate the *filtered confidence sum* as

$$\mathcal{S}_j = \sum_{o\in O_j} z_o \tag{9}$$

where O_j is the set of detected objects obtained by removing duplicated objects and those objects with a confidence smaller than a threshold from all detected objects using non maximum suppression (NMS) operations on frame j. z_o is the confidence score of a detected object o. As illustrated in Fig. 6, we identify an approximately proportional relationship between the filtered confidence sum of each frame and the inference accuracy over a large number of video clips without scene switching and density changes. We calculate the Pearson correlation coefficient between the accuracy and the confidence sum across numerous video clips sourced from various datasets and under different configurations. The results depicted in Fig. 7 validate the proportional relationship between accuracy and the confidence sum in most cases. For video clips where this relationship is not met, we plan to address them in future work.

Moreover, as shown in Fig. 8, the filtered confidence sum and the accuracy distributions of all frames in a video clip with similar content follow Gaussian distributions. Fig. 9 emphasizes that when the number of frames in a video clip (i.e., the video length) is below 200, over 80% of the video clips exhibit a Gaussian distribution of the filtered confidence sum. These observations imply that the filtered confidence sum is an ideal indicator for identifying varying video content scenarios.



(a) Distribution of inference accuracy (b) Distribution of confidence sum

Fig. 8. Gaussian distribution of accuracy and filtered confidence sum on frames with similar content.

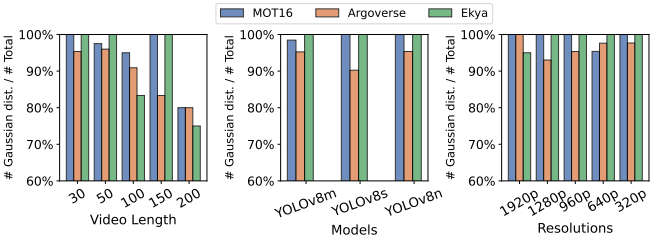


Fig. 9. A considerable percentage of video clips exhibit a Gaussian distribution in the filtered confidence sum. Identification of this Gaussian distribution is based on the Kolmogorov–Smirnov test, where a p-value exceeding 0.05 is considered. The results' generalizability was confirmed across various configurations derived from the base configuration—YOLOv8n model, 960p resolution, and 30 frames in length.

2) Accuracy Fluctuation Detection: Given the Gaussian distributions of the filter confidence sum, we adopt a two-window t-test method to detect accuracy anomaly. Specifically, assume that jth frame is being processed at a server, a sliding detection window of w frames, denoted as $W_d = \{\mathcal{S}_{j-w+1}, ..., \mathcal{S}_j\}$, and a sliding reference window of the same size, denoted as $W_f = \{\mathcal{S}_{j-2w+1}, ..., \mathcal{S}_{j-w}\}$, are used to compare the Gaussian distributions corresponding to these two windows, *i.e.*, $\mathcal{N}_f(\mu_f, \sigma_f^2)$ and $\mathcal{N}_d(\mu_d, \sigma_d^2)$, respectively. Thus, the problem to detect a significant content change is convert to test whether μ_f is not equal to μ_d when variances σ_f and σ_d are unknown, which is a Behrens-Fisher problem [23] in statistics. We adopt the t-test method to solve this problem.

First, we define the null hypothesis H_0 and alternative hypothesis H_1 as follows:

$$\begin{aligned} H_0 &: \mu_f = \mu_d \\ H_1 &: \mu_f \neq \mu_d \end{aligned} \tag{10}$$

We use Welch's approximate solution [23] to construct a t-statistic to test the null hypothesis that the two populations have equal means. The t-statistic is calculated as follows:

$$t = \frac{\bar{s}_f - \bar{s}_d}{\sqrt{\frac{\eta_f^2}{w_f} + \frac{\eta_d^2}{w_d}}} \tag{11}$$

where $\bar{s}_f$ and $\bar{s}_d$ are the means of W_f and W_d; η_f and η_d are their standard deviations; and $w_f = w_d = w$.

The degrees of freedom of the above t statistic is calculated as:

$$\nu \approx \frac{\left(\frac{\eta_f^2}{w_f} + \frac{\eta_d^2}{w_d}\right)^2}{\frac{\eta_f^4}{w_f^2 \nu_1} + \frac{\eta_d^4}{w_d^2 \nu_2}} \tag{12}$$

where $\nu_1 = w_f - 1, \nu_2 = w_d - 1$. Null hypothesis H_0 (*i.e.*, the video content changes) is rejected when $|t| > t_{\alpha/2,\nu}$, where $t_{\alpha/2,\nu}$ is the critical value of t-test with degrees of freedom as ν and level of significance as α and can be obtained from a given t-distribution table. The above t-test process continues as two windows slide along the video stream until the analysis of the entire video request is complete. If the null hypothesis H_0 is rejected and the alternate hypothesis stands, the detector will inform the service configuration controller that the current service configuration is out of date. Fig. 10 shows the core idea of the above two-window t-test mechanism.

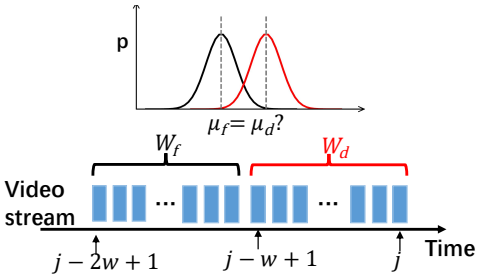


Fig. 10. Two-window t-test method for accuracy anomaly detection.

D. Service Configuration Controller

When a new video request arrives or the video content changes, the controller makes service configuration decisions by solving the optimization problem defined in Section II. Initially, we introduce an exact algorithm tailored to identify the optimal solution when the requirement distribution and performance data for all video requests are pre-known. Subsequently, we present a heuristic algorithm designed to solve the problem promptly, particularly suited for handling unknown and dynamic video analytics requests online.

1) Exact Algorithm: Our exact algorithm assumes that the performance requirements distribution for all forthcoming video requests is pre-known and that the number of service configurations and edge servers is small enough to allow it to solve the problem effectively within an acceptable time. Let I, R, M, N be the set of all video requests, all available resolutions and models, and all edge servers, respectively. $\mathcal{K} = R \times M$ is the set of all service configurations. We calculate the exact solution of the problem through the following steps:

Step 1: Select feasible configurations and construct configuration combinations for all video requests. Assume that the video request $i \in I$ has U_i feasible configurations, where each feasible configuration $\kappa \in \mathcal{K}$ satisfies accuracy and latency requirements of i. As a result, there are $\prod_{i=1}^{|I|} U_i$ potential configuration combinations for all video requests.

Step 2: Utilize existing solvers like Gurobi [24] and SCIP [25] to solve a scheduling problem for each configuration combination and find the optimal one that maximizes system throughput. The scheduling problem is an integer linear programming (ILP) problem with $|I||N|$ zero-one variables; the branch-and-cut algorithm can solve it. The time complexity for solving this ILP problem is $2^{|I||N|}$.

Although the above algorithm can find the optimal solution, its efficiency diminishes when confronted with larger problem sizes, primarily due to its high time complexity

$O(2^{|I||N|}\prod_{i=1}^{|I|}U_i)$. Besides, it is impractical when the requirement distribution and performance information for video requests are unknown. Therefore, a heuristic algorithm should be proposed to make configuration decisions online efficiently.

2) Heuristic Algorithm: In online dynamic video analytics scenarios, due to the unpredictable and sporadic nature of the incoming requests, we cannot gather all requirement distribution and performance data required by the exact algorithm. Therefore, we propose the following heuristic objective:

$$\max \min_{g} \Theta_g$$
$$\Theta_g = \min\{|g| - \sum_{n\in g}\mathcal{U}_n^{net}, |g| - \sum_{n\in g}\mathcal{U}_n^{com}, |g| - \sum_{n\in g}\mathcal{U}_n^{mem}\} \tag{13}$$

where $g \in G$ is a resource group that consists of servers that have same network bandwidth, memory capacity, and computing power; G is the resource group set; $|g|$ represents the number of servers in the resource group g; $\mathcal{U}_n^{net}, \mathcal{U}_n^{com}$, and $\mathcal{U}_n^{mem}$ are the network, computing, and memory utilization of server n, respectively; and Θ_g represents remaining capacity of bottleneck resource in the group g. This heuristic objective is to maximize the minimum remaining resource of all resource groups. The intuition is to preserve the diversity of remaining resource groups to improve the ability to service future video requests with unknown resource demands.

Based on the above heuristic objective, we propose a remaining resource balancing (RRB) based scheduling algorithm. The algorithm first obtains all available resource groups and service configurations according to performance matrices of new video requests. Then it sorts all available resource groups according to their remaining capacity in descending order and prioritizes the resource group with the most remaining resources to analyze the incoming video request. Inside each resource group, the algorithm assigns each request to the server with minimum load on the bottleneck resource to achieve load balancing. The remaining details are shown in Algorithm 1. The time complexity of the algorithm is $O(|N|\,|G|\,|M|\,|R|)$.

3) Scalability Comparison: We compare the scalability of the exact algorithm and the heuristic algorithm in terms of time complexity. The exact algorithm has a time complexity of $O(2^{|I||N|}\prod_{i=1}^{I}U_i) \approx O((2^{|N|}|M||R|)^{|I|})$, while the heuristic algorithm has a time complexity of $O(|I|\,|N|\,|G|\,|M|\,|R|)$ for all incoming video requests. The time cost of the exact algorithm increases exponentially with the number of video requests $|I|$. In contrast, the heuristic algorithm has better scalability due to its linear time complexity with respect to $|I|$. Therefore, the heuristic algorithm is more suitable for online dynamic video analytics scenarios.

Algorithm 1 RRB-based scheduling algorithm

Input:
Request set I, Service configuration set S,
Resource group set G, Edge server set N,
Performance matrices $\mathcal{P}^{net}$, $\mathcal{P}^{com}$ and $\mathcal{P}^{acc}$
Output:
Optimal configurations $\{s_i|\mathbb{I}(i)=1, i\in I\}$

for all request $i \in I$ **do**
  Obtain available resource group set Λ_i and configuration set Γ_i according to $\mathcal{P}^{net}$, $\mathcal{P}^{com}$ and $\mathcal{P}^{acc}$;
  Obtain remaining resource Θ_g for all $g \in \Lambda_i$;
  Sort all groups in Λ_i according to Θ_g in descending order, denoted as $\Lambda_i^{'}$;
  for all $g \in \Lambda_i^{'}$ **do**
    Try to assign the request i to resource group g;
    Obtain available configs of i in g, denoted as Γ_{ig};
    Find the best configuration s^* ($\in \Gamma_{ig}$) that minimizes resource utilization (denoted as $\mathcal{U}_{s^*}$);
    if $\mathcal{U}_{s^*} > 1$ **then**
      Try to assign the request i to next group g
      continue
    else
      $s_i = s^*$; $\mathbb{I}(i) = 1$
    end if
  end for
end for
return $\{s_i|\mathbb{I}(i)=1, i\in I\}$

V. Performance Evaluation

A. Experimental Setup

Implementation. We implement Novas on a testbed consisting of five edge devices (software and hardware specifications are described in Table I). A desktop machine with 16GB memory and Intel i7-8700 CPU (6 × 3.2GHz) runs some client simulators to generate vide analysis requests. The remaining four machines, comprising two Jetson Xavier NX devices and two Jetson Nano devices, serve as servers to execute DNN models using Triton Inference Server v2.8.0 with the TensorRT backend. These servers host preloaded models with different input sizes, including 1920p, 1280p, 960p, 640p, and 320p. All machines are connected to a 450 Mbps TP-LINK Router via network cable or Wifi to form a local area network. Data exchanges between client and server utilize the HTTP protocol, while control messages are transmitted via the MQTT protocol to minimize network costs.

TABLE I
Hardware and software specifications.

	Specification
Hardware	1 × Desktop, 6 Intel(R) Core(TM) i7-8700 CPUs @3.2GHz NVIDIA GeForce GTX1060 3GB; 16GB memory
	2 × NVIDIA Jetson Xavier NX module with Jetpack 5.0.2 6 CPU cores; 6GB memory,
	2 × NVIDIA Jetson Nano module with Jetpack 4.6.1 4CPU cores; 4GB memory
Software	Ubuntu 18.04, CUDA 10.2+, TensorRT 8.2+, Triton Server v2.19.0+, Python 3.6+, EMQX 4.3.10

Workloads and metrics. We use three real-world dynamic video datasets as described in Section III. Specifically, MOT16 [17] and Argoverse-HD [18] datasets are used to evaluate pedestrian detection and vehicle detection workloads. Ekya urban dataset [19] is used to evaluate accuracy anomaly detection and overall performance of Novas.

- *Normalized Root Mean Square Error (NRMSE):* indicates the error in performance prediction, calculated as $\mathrm{NRMSE} = \frac{1}{\bar{y}_t}\sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}$.
- *Precision and Recall:* evaluate the accuracy of content

change detection.

- *Goodput:* refers to the maximum number of video requests served by all servers.
- *Accuracy gain:* reflects normalized accuracy performance of all served video requests, calculated as $\frac{A_i^{served} - A_i^{goal}}{A_i^{goal}}$.
- *Latency gain:* reflects normalized latency performance of all served video requests, calculated as $\frac{L_i^{goal} - L_i^{served}}{L_i^{goal}}$.

Baselines. We compare our SVD-based collaborative filtering method with other estimation methods including KNN [26], Slopeone [27], and Co-clustering [28]. Meanwhile, we compare our RRB-based scheduling algorithm with two baseline scheduling algorithms:

- *Best-Fit* firstly selects the resolution and model with the least resource usage on each server for each video request; Then it computes a score $S(i, s_i)$ for each available configuration s_i of the request i and select the configuration corresponding to the lowest score. $S(i, s_i)$ is computed by $\sum_j w_j \left(a_j(i, s_i) - d_j(i, s_i)\right)$, where $a_j(i, s_i)$ is the proportion of free resources j (i.e., network, computing, and memory) in the server $n_i (\in s_i)$, $d_j(i, s_i)$ is the proportion of required resources if the request i, and w_j is the weight of the resource (it is set as 1/3 in our paper). This heuristic has been used for many similar scheduling problems [29], [30].
- *First-Fit* traverse the configuration list and select first configuration s' that can meet the resource requirements, i.e., $d_j(i, s') \leq a_j(i, s')$ meets for all resource types j. This heuristic is used by the work [7].

Besides, we compare the overall performance of Novas with four video analytics schemes:

- *Fix-FF/Fix-BF:* scheme uses offline measured performance data and the *First-Fit/Best-Fit* algorithm to determine service configurations.
- *Ada-FF/Ada-BF:* scheme updates performance data online based on predictor and detector of Novas, but calls the *First-Fit/Best-Fit* algorithm to adjust configurations.

B. Accuracy of Service Configuration Performance Prediction

We conduct offline performance profiling on 20 video clips from MOT16 and Argoverse datasets over 30 configurations (5 resolutions, 3 models, and 2 devices). The initial performance matrices consist of performance measures for 10 video clips, and the remaining 10 video clips form a test set. For each video clip in the test set, we randomly select performance data of three configurations and use all candidate methods to estimate missing performance data. We perform performance prediction ten times, record their execution times, and calculate NRMSE errors of all 100 prediction samples relative to the actual performance measures.

The experimental results are summarized in Fig. 11. It is clear to see that the error of the SVD-based collaborative filtering method is significantly lower than other methods both for accuracy, network latency, and compute latency estimation. Specifically, its average error for accuracy is 9.36%, for compute latency is 7.69%, and for network latency is 7.74%. Moreover, the average execution time of SVD, KNN, Slopeone, and co-cluster methods are 6.04ms, 0.24ms, 0.37ms, and 3.55ms, respectively. In practice, a delay of 6.04ms does not significantly affect overall performance.

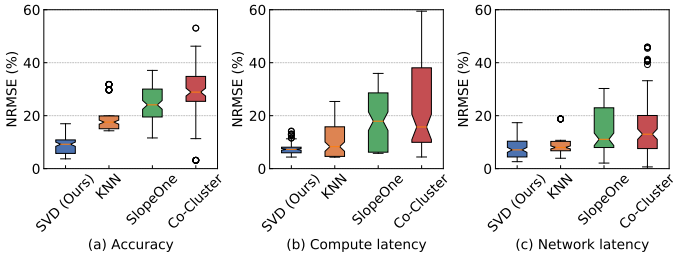


Fig. 11. NRMSE of different performance estimation methods

C. Accuracy of Video Content Change Detection

We conduct multi-object detection using the YOLOv8m model on a 45 min-long building video clip in 960p and a one-hour-long traffic video clip in 640p, and calculate the accuracy and confidence sum of each frame based on the ground truth obtained by the YOLOv8m model on raw video clips with 1920p. Then we use the two-window t-test method to detect accuracy anomalies in the video that the fluctuation range exceeds the threshold (0.05, 0.1, or 0.15). We vary the window length, ranging from 5 to 20, and the confidence level α of the t-test, ranging from 0.001 to 0.5, to find the optimal configuration for the content change detector.

Fig. 12 plots precision-recall curves of detection results. The solid bold curves represent the optimal window size, as they have the maximum area under the precision-recall curve (AUC). For these two videos, a window size of 10 is optimal for a 0.05 fluctuation threshold, while a size of 5 is best for thresholds of 0.1 and 0.15. Moreover, we should carefully select the value of significance level α due to a trade-off between precision and recall. As α increases, the rejection region expands, the method classifies more instances as positive, and the recall value increases. Yet, this also elevates false positives, reducing precision. The recall is more critical for Novas if we want to report any accuracy anomalies, where the t-test method takes it more than 90% across all fluctuation thresholds.

D. Scheduling Efficiency

We evaluate the scheduling efficiency and scalability of Novas on our testbed and a trace-driven emulated cluster comprising 100 simulated edge servers.

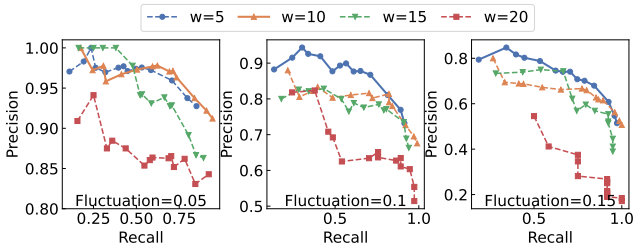
1) Goodput and resource utilization on diverse request distributions: We randomly select 90 video clips from MOT16 and Argoverse datasets as video requests and assign them one of three QoS requirement targets: $R1 = (1000ms, 0.6)$, $R2 = (500ms, 0.5)$, and $R3 = (200ms, 0.4)$. Four video analysis request distributions are considered in our experiment, *i.e.*, $(R1 * 30, R2 * 30, R3 * 30)$, $(R1 * 15, R2 * 30, R3 * 45)$, $(R1 * 15, R2 * 45, R3 * 30)$, and $(R1 * 45, R2 * 15, R3 * 30)$. Similarly, we consider four different resource distributions described in note 1 of Table II. All requests are sent to the system in order after shuffle operation. We call all candidate algorithms to schedule these requests as they arrive until the system can not meet the required performances.

TABLE II
RESOURCE UTILIZATION ON DIFFERENT SERVER AND VIDEO DISTRIBUTIONS (%).

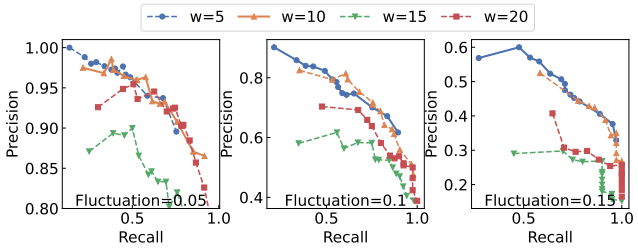
Resource Utilization	Algorithm	Resource Distribution 1 [1]				Resource Distribution 2				Resource Distribution 3				Resource Distribution 4			
		VD-1 [2]	VD-2	VD-3	VD-4	VD-1	VD-2	VD-3	VD-4	VD-1	VD-2	VD-3	VD-4	VD-1	VD-2	VD-3	VD-4
Network	First-Fit (FF)	79.5	88.2	78.6	78.8	79.3	85.9	85.0	77.6	79.8	85.4	78.6	79.2	**86.1**	86.9	85.3	77.3
	Best-Fit (BF)	79.7	79.0	**94.2**	79.0	78.5	**92.0**	93.9	77.8	79.5	**94.2**	**94.5**	78.1	79.2	78.8	**93.1**	79.8
	RRB (ours)	**83.7**	**97.5**	76.2	**83.9**	**85.5**	82.9	**97.5**	**82.0**	**88.8**	86.8	86.0	**81.9**	77.0	**89.1**	85.7	**90.8**
Computing	First-Fit (FF)	59.8	57.5	55.8	64.9	57.5	56.4	**55.5**	63.2	60.0	56.3	57.5	62.0	57.9	55.8	56.3	60.9
	Best-Fit (BF)	62.1	57.4	55.5	64.2	56.6	54.3	55.1	62.5	59.8	55.5	55.6	59.5	59.0	**56.6**	55.3	61.3
	RRB (ours)	**66.5**	**59.9**	**62.7**	**73.0**	**63.8**	**63.2**	54.8	**74.4**	**68.4**	**59.6**	**62.6**	**73.0**	**65.2**	55.1	**62.5**	**70.9**
Memory	First-Fit (FF)	59.7	58.6	42.5	52.0	49.8	48.7	49.8	51.0	42.6	50.3	42.6	43.7	58.6	48.7	57.9	59.7
	Best-Fit (BF)	49.9	40.9	42.3	41.6	40.4	58.6	49.8	42.6	43.7	42.6	41.6	43.0	40.9	41.9	41.6	52.0
	RRB (ours)	**74.3**	**72.2**	**53.6**	**74.3**	**74.3**	**65.4**	**68.7**	**66.3**	**66.0**	**66.6**	**73.3**	**73.3**	**72.2**	**72.2**	**64.9**	**67.3**

[1] There are four types of resources in our experiments (Net. Bandwidth, Device, Memory): $G1 = (25Mbps, Nano, 4GB)$,$G2 = (25Mbps, NX, 6GB)$,$G3 = (50Mbps, Nano, 4GB)$,$G4 = (50Mbps, NX, 6GB)$. Resource Distribution 1 represents the resource types of the four servers are (G1,G1,G0,G0), denoted as $RD1 = (G1, G1, G0, G0)$. Similarly, other resource distributions are $RD2 = (G3, G3, G0, G0)$,$RD3 = (G1, G1, G2, G2)$,$RD4 = (G0, G1, G2, G3)$.

[2] There are three levels of performance requirements in our experiments (Latency, Accuracy): $R1 = (1000ms, 0.6)$,$R2 = (500ms, 0.5)$, $R3 = (200ms, 0.4)$. VD-x represents xth requirement distribution, where VD-1= $(R1*30, R2*30, R3*30)$, VD-2= $(R1*15, R2*30, R3*45)$, VD-3= $(R1*15, R2*45, R3*30)$, and VD-4= $(R1*45, R2*15, R3*30)$.



(a) Building video



(b) Traffic video

Fig. 12. Precision-Recall curve of content change detection.

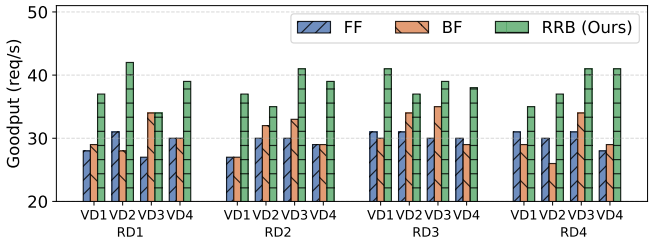
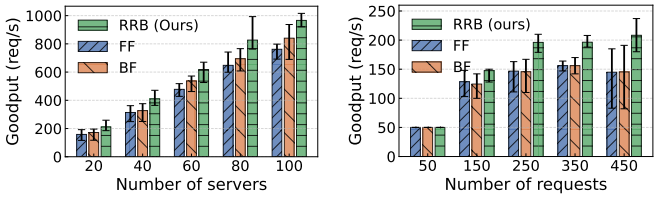


Fig. 13. Goodput comparison of different scheduling methods



(a) Varying number of servers (b) Varying number of requests

Fig. 14. Scalability comparison on emulated servers.

Fig. 13 shows that Novas achieves the highest goodput across all distributions compared with other algorithms. Specifically, our RRB algorithm improves the goodput by 29.3% and 25.6% compared to First-Fit and Best-Fit on average, respectively. In the best-case scenario, it outperforms the other two algorithms by 46.4% and 50%, respectively. Even in the worst-case scenario, it still achieves improvements of 13% and 9% respectively. The resource utilization results are summarized in Table II, which shows that RRB significantly improve resource utilization in most cases. Network and computing resource utilization is improved by 4.8% and 10.4% compared to First-Fit and improved by 1.8% and 11.4% compared to Best-Fit. The memory utilization of RRB is 35.5% and 54.9% higher than that of First-Fit and Best-Fit algorithms due to more models loaded on servers.

2) Scalability on varying number of servers and requests: We use simulated clients and edge servers to evaluate the scalability of Novas. Video analysis requests contain pedestrian detection and vehicle detection workloads with different performance demands randomly. Fig. 14 (a) shows the goodput of various algorithms as the number of servers increases from 20 to 100 (and the number of requests correspondingly increases from 300 to 1500 with an interval of 300). The result shows that Novas achieves linear scaling with the number of servers and highest goodput. Fig. 14 (b) shows that the goodput of our algorithm is much greater than other methods as the number of requests increases when 20 servers are used and remains stable until the resource reaches its limit.

E. Overall Performance of Novas

We randomly select 40 video clips of 40 minutes long from Ekya traffic and building datasets to evaluate the overall performance of Novas. Each video requires the latency no higher than 200ms/300ms and accuracy (mAP) no lower than 0.3/0.4. The client simulator sends 10 video requests to the system every 10 minutes. At runtime, Novas automatically identifies accuracy anomalies and select the best configuration to execute. We compare Novas with four baseline schemes (Fix-FF, Fix-BF, Ada-FF, and Ada-BF) on accuracy gain, latency gain, and goodput.

Fig. 15 shows the CDFs of accuracy and latency gain over all video requests of four random request distributions. For Fix-BF and Fix-FF schemes, more than 97% accuracy gains are less than 0, which indicates that using fixed service configurations derived from offline profiling fails to deal with dynamic video analysis requests. In contrast, adaptive adjustment schemes Novas, Ada-BF and Ada-FF have more than 83.33%, 81.25%, and 80.21% accuracy gains greater than

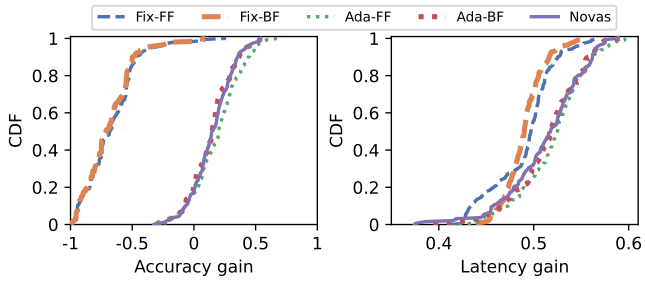


Fig. 15. Overall performance of Novas.

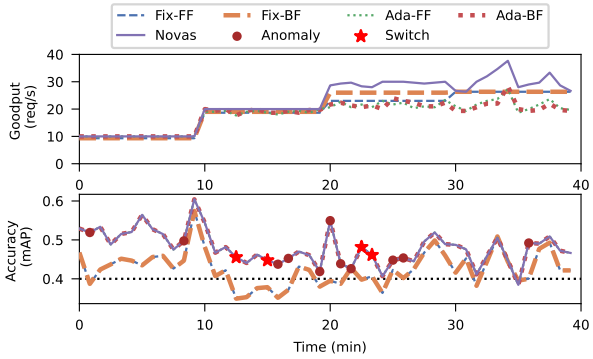


Fig. 16. Goodput and configuration adjustment of Novas.

0, respectively, with the maximum accuracy violation less than 32.5%. Moreover, all schemes can meet latency requirements.

The goodput subgraph of Fig. 16 shows that the peak goodput of Novas is 43% higher than Fix-FF and Fix-BF schemes, 37.72% higher than the Ada-FF scheme, 32.86% higher than the Ada-BF scheme. The bottom subgraph of Fig. 16 plots configuration adjustments during one video clip. Detected accuracy anomalies and configuration switches are marked. Interestingly, the number of configuration switches is far smaller than that of detected accuracy anomalies because the controller finds the current configuration still optimal even if some anomalies occur. This observation indicates that component collaboration enables Novas to tolerate some errors caused by individuals, thereby avoiding their negative impact on overall performance. Besides, we can see that most time Novas can maintain the required analysis accuracy.

F. System Overhead

We evaluate the overhead of different components on a desktop with 6 Intel Core i7-8700 CPUs @3.2 GHz, and the results are shown in Fig. 17. The number of models and resolutions keeps constant in our evaluation. For the predictor, the performance model training time is proportional to the number of video streams, which is only about 110 ms, even if the number of video streams reaches 1000. Its prediction time is less than 1 ms and can be negligible at runtime. The running time of the detector is proportional to the window length, but it is always below 1 ms even when the window length reaches 150. The running time of the controller is proportional to the number of resource groups and servers, where the pre-processing operation takes up to 100 ms, and the scheduling operation always runs under 2 ms. The reading and writing of performance files and related calculations take up most of the

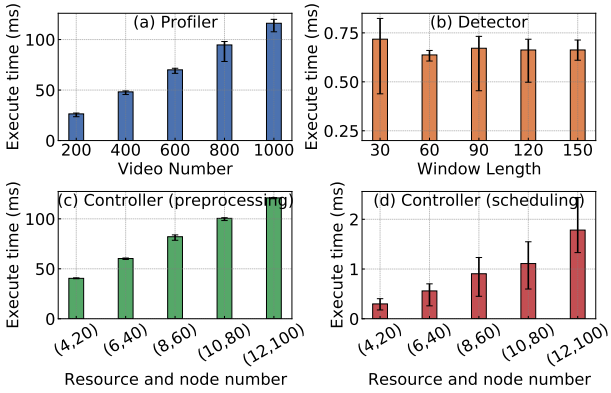


Fig. 17. Overhead of different components in Novas.

time of pre-processing operations. While this can be further optimized, the current time consumption does not affect the performance of online video analysis.

VI. Related Work

Accuracy anomaly detection on DNN models. In the lack of ground truth information, two approaches can be employed to identify accuracy anomalies. One involves utilizing transfer learning methods to generate new labeled data, facilitating the computation of accuracy metrics [31]. The alternative approach is to directly leverage the inputs, internal state, and outputs of DNN models to highlight accuracy anomalies. DISSECTOR [32] ensures accurate analysis results by filtering out unbefitting inputs. SelfChecker [33] monitors internal layer features of DNN models and triggers an accuracy alarm when they are inconsistent with the final prediction. SELFORACLE [34] identifies the confidence boundary between normal and unsupported conditions to predict misbehavior in autonomous driving systems. Furthermore, calibrated confidence can replace the accuracy metric to determine the early-exit point of models [35], [36] or select smaller models [37] for reducing the execution time of models on resource-constrained devices. In this paper, we focus on detecting significant fluctuations in accuracy, and a confidence sum proportional to the change in accuracy is deemed sufficient for this purpose.

Adaptive configuration on video analytics. Adaptive configuration on video analytics workloads aims to achieve analysis accuracy and resource cost tradeoff. Some prior work [4]–[10] adjust video parameters like frame sampling rate and resolution to reduce computing and network cost. They rely on a performance-accuracy profiler to obtain analysis results of different configurations and compute the accuracy by comparing them with ground truth obtained by running a ”golden” configuration. This profiling process need to be executed repeatedly and consume a lot of resources when the video content changes rapidly. Some work encodes important parts [11]–[14] or extracts key features [38] to reduce data transmission and computation complexity. Other work utilizes the low-power GPU/CPU on camera devices to run some cheap DNN models to analyze and filter frames [39] or inference in advance on some frames with salient features [40]. At the mobile edge, resource-constrained devices can only support simple parameter adjustments like resolution. Novas

combines model selection and request scheduling to achieve adaptive configuration and improve resource efficiency.

Scheduling optimization with multiple constraints. Task scheduling problems at the edge focus on how to improve resource utilization [41] or maximize task performance (latency [42], [43], throughput [44], [45], or both [46]) under the condition of diversified resource constraints. Network, computing power and energy constraints are frequently considered in existing work like [43]–[45]. Besides, the memory constraint is also getting more attention with the rise of DNN analysis workloads [47], [48]. Video analysis workloads consume much network, compute and memory resources, and our scheduling algorithm needs to consider multiple resource limitations and robustness in heterogeneous resources.

VII. Conclusion and Future Work

This paper proposes Novas, a scalable online edge video analytics scheme specialized for dynamic videos. Novas achieves maximized system resource utilization while guaranteeing diverse QoS requirements by integrating online inference accuracy fluctuation detection, lightweight configuration performance prediction, and a RRB heuristic scheduling strategy. These innovative techniques enable Novas to automatically fit service configurations to dynamic video analysis requests. We have implemented Novas and conducted both real-world experiments and extensive trace-driven simulations. Novas is easy and scalable for large-scale implementation. Experiment results demonstrate the efficacy of Novas design. In the future, we will further explore the impact of dynamic system resources due to unexpected join and leave of voluntary edge servers.

References

[1] M. Hu, Z. Luo, A. Pasdar, Y. C. Lee, Y. Zhou, and D. Wu, "Edge-based video analytics: A survey," *arXiv preprint arXiv:2303.14329*, 2023.

[2] Intelligent roadside units and v2x solutions. Accessed March 4, 2024. [Online]. Available: https://campaign.advantech.online/en/Road_Side_V2X/

[3] Nvidia agx systems. Accessed March 4, 2024. [Online]. Available: https://www.nvidia.com/en-us/deep-learning-ai/products/agx-systems/

[4] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 236–252.

[5] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and {Delay-Tolerance}," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 377–392.

[6] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 253–266.

[7] S. Zhang, C. Wang, Y. Jin, J. Wu, Z. Qian, M. Xiao, and S. Lu, "Adaptive configuration selection and bandwidth allocation for edge-based video analytics," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 285–298, 2021.

[8] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 756–764.

[9] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, "Videoedge: Processing camera streams using hierarchical clusters," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 115–131.

[10] K. Zhao, Z. Zhou, X. Chen, R. Zhou, X. Zhang, S. Yu, and D. Wu, "Edgeadaptor: Online configuration adaption, model selection and resource provisioning for edge dnn inference serving at scale," *IEEE Transactions on Mobile Computing*, 2022.

[11] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 155–168.

[12] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 359–376.

[13] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *The 25th annual international conference on mobile computing and networking*, 2019, pp. 1–16.

[14] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 557–570.

[15] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of queueing theory*. John Wiley & Sons, 2018, vol. 399.

[16] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. [Online]. Available: https://doi.org/10.1007/978-3-540-24777-7

[17] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[18] M. Li, Y.-X. Wang, and D. Ramanan, "Towards streaming perception," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 473–488.

[19] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 119–135.

[20] Yolov8. Accessed March 4, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[21] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and qos-aware cluster management," *ACM SIGPLAN Notices*, vol. 49, no. 4, pp. 127–144, 2014.

[22] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, "{Heterogeneity-Aware} cluster scheduling policies for deep learning workloads," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 481–498.

[23] C.-H. Chang and N. Pal, "A revisit to the behrens–fisher problem: comparison of five test methods," *Communications in Statistics—Simulation and Computation®*, vol. 37, no. 6, pp. 1064–1085, 2008.

[24] Gurobi optimization. Accessed March 4, 2024. [Online]. Available: https://www.gurobi.com/

[25] Scip optimization suite. Accessed March 4, 2024. [Online]. Available: https://scipopt.org/

[26] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 1, pp. 1–24, 2010.

[27] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 2005, pp. 471–475.

[28] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*. Springer, 2010, pp. 1–35.

[29] O. Hadary, L. Marshall, I. Menache, A. Pan, E. E. Greeff, D. Dion, S. Dorminey, S. Joshi, Y. Chen, M. Russinovich *et al.*, "Protean:{VM} allocation service at scale," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 845–861.

[30] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," *research. microsoft. com*, 2011.

[31] K. Zhang, X. Liu, X. Xie, J. Zhang, B. Niu, and K. Li, "A cross-domain federated learning framework for wireless human sensing," *IEEE Network*, vol. 36, no. 5, pp. 122–128, 2022.

[32] H. Wang, J. Xu, C. Xu, X. Ma, and J. Lu, "Dissector: Input validation for deep learning applications by crossing-layer dissection," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 727–738.

[33] Y. Xiao, I. Beschastnikh, D. S. Rosenblum, C. Sun, S. Elbaum, Y. Lin, and J. S. Dong, "Self-checking deep neural networks in deployment," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 372–384.
[34] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 359–371.
[35] T. Tan and G. Cao, "Deep learning on mobile devices through neural processing units and edge computing," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1209–1218.
[36] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "Spinn: synergistic progressive inference of neural networks over device and cloud," in *Proceedings of the 26th annual international conference on mobile computing and networking*, 2020, pp. 1–15.
[37] D. Kang, S. Lee, H. S. Chwa, S.-H. Bae, C. M. Kang, J. Lee, and H. Baek, "Rt-mot: Confidence-aware real-time scheduling framework for multi-object tracking tasks," in *2022 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2022, pp. 318–330.
[38] S. Wang, C. Ding, N. Zhang, X. Liu, A. Zhou, J. Cao, and X. Shen, "A cloud-guided feature extraction approach for image retrieval in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 292–305, 2019.
[39] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015, pp. 426–438.
[40] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: optimizing neural network queries over video at scale," *arXiv preprint arXiv:1703.02529*, 2017.
[41] Z. Ning, P. Dong, X. Wang, S. Wang, X. Hu, S. Guo, T. Qiu, B. Hu, and R. Y. Kwok, "Distributed and dynamic service placement in pervasive edge computing networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1277–1292, 2020.
[42] Z. Han, H. Tan, X.-Y. Li, S. H.-C. Jiang, Y. Li, and F. C. Lau, "Ondisc: Online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2472–2485, 2019.
[43] J. Meng, H. Tan, X.-Y. Li, Z. Han, and B. Li, "Online deadline-aware task dispatching and scheduling in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1270–1286, 2019.
[44] K. Yang, P. Sun, J. Lin, A. Boukerche, and L. Song, "A novel distributed task scheduling framework for supporting vehicular edge intelligence," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 972–982.
[45] M. Zhang, J. Cao, L. Yang, L. Zhang, Y. Sahni, and S. Jiang, "Ents: An edge-native task scheduling system for collaborative edge computing," in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*. IEEE, 2022, pp. 149–161.
[46] Z. Yang, K. Nahrstedt, H. Guo, and Q. Zhou, "Deeprt: A soft real time scheduler for computer vision applications on the edge," in *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2021, pp. 271–284.
[47] N. Ling, K. Wang, Y. He, G. Xing, and D. Xie, "Rt-mdl: Supporting real-time mixed deep learning tasks on edge platforms," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021, pp. 1–14.
[48] A. Padmanabhan, N. Agarwal, A. Iyer, G. Ananthanarayanan, Y. Shu, N. Karianakis, G. H. Xu, and R. Netravali, "Gemel: Model merging for {Memory-Efficient},{Real-Time} video analytics at the edge," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 973–994.

Liang Zhang received the B.E. degree in Computer Science and Technology from Northeastern University in China, in 2018. She is currently working toward the PhD degree in the Department of Computer Science and Engineering at Shanghai Jiao Tong University. Her research interest includes stream processing and resource scheduling in the cloud or edge computing environment. For more information, please visit https://zl-cs.github.io/.

Hongzi Zhu received the PhD degree in computer science from Shanghai Jiao Tong University, in 2009. He was a post-doctoral fellow with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, and the Department of Electrical and Computer Engineering, University of Waterloo, in 2009 and 2010, respectively. He is a professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include mobile sensing, mobile computing, and Internet of Things. He received the Best Paper Award from IEEE Globecom 2016. He is an associate editor for the IEEE Transactions on Vehicular Technology and the IEEE Internet of Things Journal.

Wen Fei received the B.S. degree from Northeastern University, Shenyang, China, in 2018. He is currently pursuing the Ph.D. degree at the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include deep neural network explanation and compression.

Yunzhe Li received his B.S. degree in computer science from Wuhan University in 2021. He is a PhD student in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include mobile sensing and edge computing.

Mingjin Zhang is currently a Ph.D. candidate with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China. He received the B.Eng. degree in communication engineering from Wuhan University of Technology, China, in 2019. He has been a visiting PhD student in Department of Computer Science and Technology, University of Cambridge, from Feb 2023 to Sep 2023. His research interests include edge computing, edge AI, distributed machine learning, and Internet of Things.

Jiannong Cao (Fellow, IEEE) is the Otto Poon Charitable Foundation Professor in Data Science and the Chair Professor of Distributed and Mobile Computing in the Department of Computing at The Hong Kong Polytechnic University, Hong Kong. He is the Dean of Graduate School, the director of the Research Institute for Artificial Intelligence of Things in PolyU, and the director of the Internet and Mobile Computing Lab. He is now the associate director of PolyU's University Research Facility in Big Data Analytics. Prof. Cao is a member of Academia Europaea, a fellow of IEEE, a fellow of the China Computer Federation, and an ACM distinguished member. His research interests include distributed systems and blockchain, wireless sensing and networking, big data and machine learning, and mobile cloud and edge computing.

Minyi Guo (Fellow, IEEE) is a Zhiyuan Chair Professor in the Department of Computer Science and Engineering at Shanghai Jiao Tong University, China. He received the PhD degree in computer science from the University of Tsukuba, Japan. He is currently Zhiyuan Chair professor and head with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His present research interests include parallel/distributed computing, compiler optimizations, embedded systems, pervasive computing, big data and cloud computing. He is now on the editorial board for IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Cloud Computing and Journal of Parallel and Distributed Computing. He is a fellow of IEEE.