Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2025 SCPE. Volume 26, Issues 3, pp. 1231–1241, DOI 10.12694/scpe.v26i3.4122

INTERPRETABLE AI ARCHITECTURE OF MACHINE LEARNING ALGORITHM FOR INTELLIGENT VIDEO SURVEILLANCE BASED ON FOG AND EDGE COMPUTING

JIE ZHANG,* WEIPING SONG,† WENKUI SHE,‡ HUANHUAN LI,§ FEIHU HUANG,¶ AND FAN YANG‖

Abstract. In order to address the application scenarios and technical requirements of video monitoring, the centralized data processing model represented by cloud computing has a large cost in terms of resource requirements, relies excessively on the network bandwidth of the cloud computing center, and is difficult to meet the needs of video processing in real-time and other aspects, the author proposes an interpretable AI architecture of intelligent video monitoring machine learning algorithm based on fog and edge computing. The author proposes a edge computing model suitable for video monitoring scenarios. From the three main resources of computing, network bandwidth and storage as the entry point, the system architecture is designed. After using the computing power of edge nodes to complete video preprocessing, the Docker container platform is built, and hierarchical scheduling strategy is adopted to reduce network congestion. The results showed that compared to video surveillance with added motion detection function, analyzing data from week 1 to week 3 compared to video surveillance without added motion detection function, the number of video files stored was about 1176 records per week, saving about 41.56% of storage space. This model can effectively reduce the computational, storage, and network transmission costs in video surveillance scenarios.

Key words: Edge computing model, Intelligent video surveillance, Docker container, scheduling strategy

1. Introduction. Fog computing is a concept proposed by Cisco in the United States in 2011. It is a distributed computing oriented towards the Internet of Things, aimed at supplementing and improving cloud computing [1]. Compared to "cloud computing," fog computing can more vividly describe its style of being close to the ground and vast permeability, closer to people's surroundings, and more down-to-earth. The original intention of fog computing design is to deploy a large number of devices with average computing power and low cost, thereby extending large-scale networked computing power and data analysis capabilities to the edge of the network, allowing users to receive fast network service feedback and data feedback. Edge computing is one of the important technologies to realize fog architecture. The European Telecommunications Standards Institute (ETSI) first gave an explanation of edge computing at mobile terminals and provided a reference architecture for edge computing in 2014 [2]. By 2025, Gartner forecasts that the majority — 75% — of enterprise-generated data will be handled outside of conventional cloud platforms. Beginning March 6, 2020, AT&T entered into a partnership with Alphabet's Google Cloud to leverage 5G edge computing capabilities. This collaboration aims to enhance customer experiences by deploying applications nearer to end-users, thereby bolstering speed and security [3].

As one of the terminal devices with the greatest demand for network bandwidth, storage and computing capacity at this stage, video monitoring is a typical scenario of the above edge computing applications. Traditional video monitoring systems generate video stream data, which is mostly transmitted to the business center in real time through IP networks. Edge computing has obvious advantages not only in solving the bottleneck of processing capacity in traditional video monitoring systems, improving system capacity and quality of service, but also in emerging intelligent video monitoring systems. On the one hand, using the idea of edge computing can process data near video monitoring equipment, reduce data sending to the cloud, reduce bandwidth, cloud

*Aostar Information Technologies Co., Ltd., Chengdu, 610041, China.(Corresponding author's email:JieZhang19@126.com)
†Aostar Information Technologies Co., Ltd., Chengdu, 610041, China.(WeipingSong7@163.com)
‡Aostar Information Technologies Co., Ltd., Chengdu, 610041, China.(WenkuiShe@126.com)
§Aostar Information Technologies Co., Ltd., Chengdu, 610041, China.(HuanhuanLi9@163.com)
¶Aostar Information Technologies Co., Ltd., Chengdu, 610041, China. College of Computer Science, Sichuan University, Chengdu, 610065, China.(FeihuHuang6@126.com)
‖Aostar Information Technologies Co., Ltd., Chengdu, 610041, China.(FanYang966@163.com)

computing and storage pressure, and reduce latency and improve QoS for online processing scenarios [4,5]. On the other hand, AI technologies such as data desensitization and federated learning can be combined to improve the system's capability level. These are the reasons for applying edge computing in the field of intelligent video surveillance.

2. Literature Review. The idea of edge computing is to propose content distribution network technology in order to solve the problems of small network bandwidth, large and uneven user access and other problems existing in its own business, deploy cache servers close to users according to geographical location, redirect user requests to cache servers, reduce the pressure on central servers, reduce network latency, and improve service quality [6]. Yu, J. et al. developed a BP neural network model that is enhanced by a genetic algorithm, with edge computing serving as the central component. They validated the effectiveness of this model through case studies. Their practical analysis demonstrates that the predictions generated by this model exhibit higher accuracy and alignment with real-world power grid scenarios. As a result, this model holds promise for practical application in various contexts [7]. Song, W. et al. introduced an innovative approach termed Intelligent Standing Human Detection (ISHD). This method relies on an enhanced Single Shot Multi-Box Detector (SSD) to identify standing human poses within video surveillance frames, particularly in exam stage environments[8]. Isaac Martín de Diego. et al. devised a novel method aimed at enhancing existing intelligent video surveillance systems by imbuing them with environmental sensitivity. This method leverages expert guidance to tailor the system's behavior to specific scenarios, incorporating data for generating alerts and contextual understanding. Encouraging experimental outcomes underscore the promise of this approach, positioning it as a foundation for future system improvements [9]. Khosravi, M. R. et al. investigated various facets of multimedia computing within the realm of Video Synthetic Aperture Radar (SAR), an emerging real-time remote sensing and surveillance radar imaging mode [10].

The author proposes to deploy the video capture framework and container application based on edge computing in the smart campus system. In the video surveillance system based on the edge computing model, edge nodes are used to deploy motion detection algorithms to preprocess the video streams collected by edge devices, such as moving object detection, so as to reduce redundant information and reduce the system's demand for storage and transmission. Deploying a container platform brings better scheduling features, solves the resource scheduling problem of edge computing nodes, and improves the overall performance of the cloud computing center.

3. Research Methods.

3.1. Requirements for motion detection function. In monitoring scenarios, there are often a large number of cameras. If abnormal situations need to be detected in a timely manner, a large number of manual personnel are usually arranged to conduct real-time 7∗ 24 hours of video inspection. Meanwhile, due to the persistent nature of video information flow, the transmission of video information will also bring huge network loads; Over time, the video data generated by monitoring systems will also bring enormous storage pressure [11]. If these massive video data are directly uploaded to cloud computing centers, on the one hand, the processing of video information requires a large amount of computing resources, and on the other hand, the transmission and storage of data will also face enormous pressure.

The main purpose of storing video information in monitoring scenarios is to record changes and suspicious information in the scene. If appropriate techniques or methods are not used, the monitoring system will occupy a large network bandwidth, transmit long-term records of unchanged monitoring scenes, have low effective information content, and video information will lose its storage significance [12]. Therefore, in intelligent video surveillance systems, motion detection technology is the most basic and important technology. This technology detects moving targets in video data streams through appropriate algorithms, replacing manual recognition work. By setting certain parameters, the motion characteristics or position information of moving targets in the monitoring scene can be discovered, achieving automatic alarm or determining whether video information meets the requirements of storage or transmission backup, which can effectively save storage space and reduce network transmission pressure. Among various motion detection algorithms, optical flow method, background difference method, and inter frame difference method are the most common.

3.1.1. Optical flow method. The optical flow technique correlates a two-dimensional velocity field with grayscale images, incorporating constraint equations to derive the fundamental algorithm for optical flow computation. While this method is relatively straightforward and simple to deploy, fluctuations in lighting conditions or occluded objects can distort the optical flow field, thereby amplifying the computational demands of the algorithm. Consequently, these limitations pose challenges when applying optical flow in real-time scenarios [13,14].

3.1.2. Background difference method. The background difference method selects a specific image as the background frame, and then performs a difference operation between the current video frame or image to be judged and the background frame, in order to further determine whether there is a moving target, the processing process of the background difference algorithm is as follows:

(1) Select the image without moving objects entering the monitoring screen as the background frame, defined as $background(x, y)$;
(2) Select the current frame that requires comparison and judgment, defined as $frame_k(x, y)$;
(3) Set the threshold to T, perform a difference operation between the current frame $frame_k(x, y)$ and the background frame $background(x, y)$, compare the difference result with the threshold T, and binarize to obtain the moving target. If it is greater than the threshold T, it is judged that there is a moving target. If it is less than or equal to the threshold T, it is judged that there is no moving target. The formal calculation formula is expressed as follows 3.1:

$$detext(x, y) = \begin{cases} 1 & \text{if } |frame_k(x, y) - background(x, y)| > T \\ 0 & \text{others.} \end{cases} \tag{3.1}$$

$detext(x, y)$ is the binary image obtained by differential operation and binarization between the current frame and the background frame. Only when $detext(x, y) = 1$ is used, it indicates the detection of a moving target.

The background difference method only requires differential detection of one frame, which is fast and accurate. However, the background subtraction algorithm largely relies on the reliability of the background frame $background(x, y)$. If there are changes in lighting, shadows, etc., it is necessary to continuously adjust the background frame to adapt to changes in the environment. Therefore, the background subtraction algorithm is more suitable for fixed cameras [15].

3.1.3. Two frame difference method. The two frame difference method, also known as the inter frame difference method, has a similar algorithm design approach to background difference. It adopts an improved approach to select adjacent two frames of images, grayscale the current frame $frame_k(x, y)$ and the previous frame $frame_{k-1}(x, y)$, and perform differential operation. The inter frame difference method is not affected by slow light changes, and the algorithm is simple and easy to implement. The formal calculation formula is expressed as follows 3.2:

$$detext(x, y) = \begin{cases} 1, & \text{if } |frame_k(x, y) - frame_{k-1}(x, y)| > T \\ 0, & \text{others.} \end{cases} \tag{3.2}$$

But only the parts that change before and after the two frames can be detected, and the overlapping parts cannot be detected, which can easily lead to problems such as blurred and incomplete edges. When the object moves slowly, misjudgment or hollow phenomena may occur [16].

3.1.4. Three frame difference method. On the basis of the two frame difference method, researchers have proposed the three frame difference method. The basic idea is to extract consecutive three frames of images $frame_{k-1}(x, y)$, $frame_k(x, y)$, and $frame_{k+1}(x, y)$ The algorithm flow is shown in Figure 3.1.

The algorithm calculation process is as follows:

(1) Perform inter frame difference operation between frame k-1 and frame k according to formula 3.2 to obtain $detext_1(x, y)$;
(2) Perform inter frame difference operation between frame k and frame k+1 according to formula 3.2 to obtain $detect_2(x, y)$;

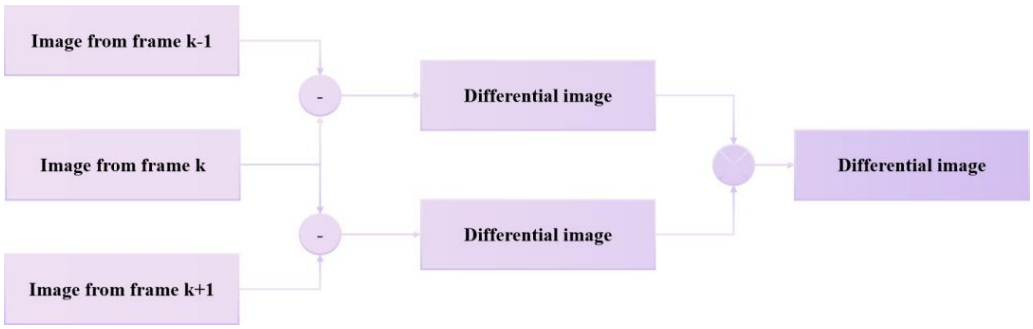

Fig. 3.1: Three frame differential algorithm flowchart

(3) The calculation results of $detect_1(x, y)$ and $detect_2(x, y)$ are combined, and the formal calculation formula is expressed as follows 3.3:

$$DETE(x, y) = detect_1(x, y) \bigotimes detect_1(x, y) = \begin{cases} 1 & detect_1(x, y) detect_1(x, y) \neq 0 \\ 0, & \text{others.} \end{cases} \tag{3.3}$$

Among them, $DETE(x, y)$ is the result of logical AND operation.

Similar to the two frame difference method, the three frame difference method still has the phenomenon of holes in the process of detecting moving targets. However, the three frame difference method can locate the position of the moving target in the monitoring screen, improving the accuracy of motion detection. The detection results are more accurate than the two frame difference method [17].

Based on the analysis of the above motion detection algorithms, in the author's research process, the three frame difference method can be used to implement the motion detection module, build a video monitoring system for the campus network, reduce the difficulty of system construction, achieve selective storage of video information, filter out effective video frames, reduce redundant information in the monitoring video, and achieve the goal of improving the effectiveness of video information. At the same time, it reduces the storage space and network bandwidth expenses of the massive video information obtained by a large number of video capture devices, achieving the goal of saving construction costs.

3.2. Analysis of storage and network bandwidth requirements.

3.2.1. Camera stream analysis. In order to meet the construction requirements of smart campuses, high-definition cameras are mainly used in the monitoring system. The specifications of cameras vary, and the amount of data generated also varies. Taking HD digital cameras as an example, calculated based on a 2048 Kbps stream, each camera generates approximately 900 M of video data per hour and approximately 21 GB of new video data per day.

3.2.2. Analysis of Video Data Transmission and Storage. Taking the teaching building as an example, the floor structure is in a zigzag shape, with corridors above and below, and 5 classrooms distributed on each side. There are 10 classrooms on each floor, with a total of 4 corridors up, down, left, and right. The construction cost of the comprehensive monitoring system includes installing one FHD digital camera at each end of the four corridors, and installing two HD digital cameras in each classroom. A total of 28 cameras of two specifications are installed on each floor. When the system is running normally, each camera in the intelligent video monitoring system will generate two real-time data streams, which are used to monitor the video data stream transmitted in real-time and the video data file transmitted to the storage data stream of the cloud computing storage center. When an emergency occurs, the staff of the monitoring center need to view the video surveillance content in real time, and the storage center of cloud computing will also store video information in real time. The network bandwidth of the monitoring system will reach its maximum demand.

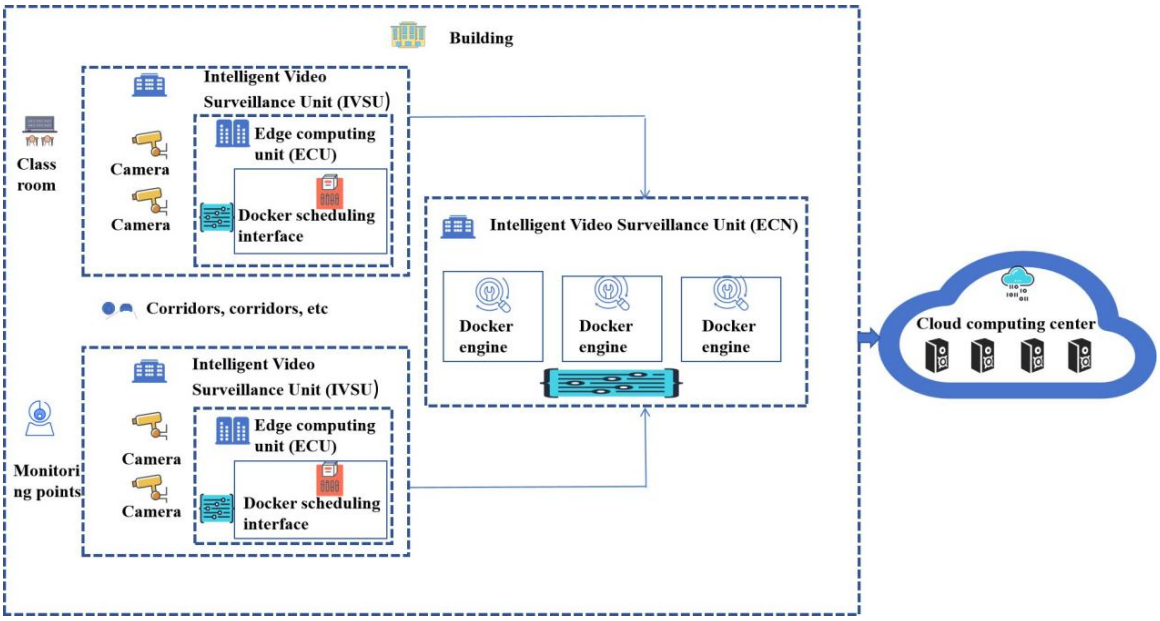

Fig. 3.2: Architecture of intelligent video monitoring system based on edge computing model

When an emergency situation occurs, the maximum real-time video data stream and maximum storage data stream generated on each floor are: 4 Mbps $*$ 8 channels+2 Mbps $*$ 20 channels=72 Mbps [18]. Therefore, calculated by floor, the maximum network bandwidth requirement is a total of 144 Mbps, and the maximum storage write rate requirement for cloud computing centers is 72 Mbps. Deploying according to the same construction specifications, if an emergency occurs in the entire building, calculated based on 6 floors, the maximum transmission bandwidth needs to be 864 Mbps, and the maximum storage write rate needs to be 432 Mbps.

3.3. Management Requirements Analysis. Intelligent video surveillance systems generally include video capture, image preprocessing, motion object detection, motion object tracking, motion object classification, behavior description and understanding, and alarm processing modules. The corresponding management functions are also designed around these modules to improve the accuracy and response speed of the system, enhance the overall reliability of the system, and provide convenient management functions for managers. Compared with the traditional IVSS, the video monitoring system based on the edge computing model, in addition to the functions involved above, uses the edge computing model to pre process the video information at the edge nodes in a decentralized manner and store initial data. Therefore, the design adopts container technology to build a containerized resource scheduling platform, adopts appropriate strategies to achieve scheduling control of network bandwidth resources, uploads video files to cloud computing centers, realizes backup storage of video files, reduces network load and storage space requirements, improves resource utilization, and achieves the goal of reducing system construction costs [19].

3.4. Architecture Design. The architecture diagram is designed based on the structure of the teaching building. The intelligent video monitoring system based on the edge computing model is planned to have the following four roles: Edge computing Unit (ECU), Intelligent Video Surveillance Unit (IVSU), edge computing Node (ECN), Cloud Computing DataCenter (CCDC). The system architecture is shown in Figure 3.2.

(1) ECU. Edge computing unit, with certain computing power, can preprocess the video information collected by the camera, and provide file storage and network transmission functions. In the subsequent model validation, Raspberry Pi Zero W single board computer was used, with Linux installed at the bottom to provide Docker scheduling interface for the future.

(2) IVSU. Intelligent video monitoring unit, based on the ECU module, installs motionEyeOS and CSI cameras to achieve intelligent video monitoring unit, deploys video acquisition points, such as classrooms, corridors, training rooms, libraries, etc.
(3) ECN. The edge computing node has high computing power and can provide large storage space for temporary or long-term storage of video data generated by IVSU. It can select servers or other general-purpose computers as the hardware support environment according to the number of cameras, which is easy to deploy a container platform to support subsequent resource scheduling and model verification.
(4) CCDC. Cloud computing data center, deploying a large number of servers and storage hardware, using platforms or tools such as KVM and VMware, to build a cloud computing resource management center, and store massive video information for intelligent video monitoring systems.

3.5. Video capture framework design. When designing a video capture framework, Raspberry Pi is used as the hardware infrastructure for the framework while ensuring availability, in order to reduce construction costs. Deploy containers based on open-source platforms and provide support for scheduling strategies in a decentralized environment.

3.5.1. Open Hardware Architecture. Raspberry Pi is an open, easily scalable small single board computer with low power consumption, customizable on demand, providing all expected features or capabilities. Widely used in real-time image and video processing, as well as various IoT based applications. The video capture framework adopts edge nodes with Raspberry Pi Zero W as the core component, and loads CSI cameras to achieve the video capture function of the monitoring system.

Raspberry Pi Zero W single board computer, compact design, low power consumption, powered by Micro USB interface, low cost for video capture. This single board computer uses BCM2835 as the SoC, integrating various functions from general-purpose computers. In the calculation module, ARM1176JZF-S is used, providing 700MHz of computing power; In the video module, Broadcom VideoCore IV technology is adopted, which can achieve 1080p H.264 video encoding or decoding at 30 frames per second, while providing miniHDMI output function; In terms of network connection, WiFi and Bluetooth modules are provided, supporting 802.11n connection; In terms of other interfaces, it provides a mini USB On the Go interface, a Micro SD card interface, and a 40pin GPIO interface. For cameras, it provides a CSI interface that can adapt to Raspberry Pi Camera Module V2 cameras and capture high-definition videos up to 8 million pixels.

3.5.2. Open source video capture system. MotionEyeOS is an embedded operating system that uses the BuildRoot tool to complete cross compilation. It is suitable for deployment on single board computers and provides the implementation of a complete video surveillance system. The front-end of the video surveillance system is a motionEye program written in Python, providing web access functionality; The backend adopts a highly configurable motion program, which can view video streams in real time, and also achieve functions such as facial recognition, dynamic monitoring, camera direct recording, recording activity images, and creating dynamic video files. Improve motionEyeOS, simplify the process, use three frame difference algorithm, and achieve motion detection function [20].

3.6. Design of containerized resource scheduling scheme. In the design process of scheduling schemes, load is the main factor affecting the demand for application resources. Combining with the practical application of intelligent video surveillance systems, the system bottleneck mainly focuses on the network and disk I/O. Therefore, when designing a scheduling plan, the characteristics of the Docker container engine should be utilized first to provide information support for containerized scheduling plans by periodically collecting resource load information such as CPU, memory, disk I/O, and network bandwidth. In terms of scheduling mode, it fully reflects the decentralized processing characteristics of the edge computing model. For the architecture shown in Figure 3.1, a two-level scheduling mode is adopted, that is, the cloud computing data center (CCDC) schedules the edge computing node (ECN), and the edge computing node (ECN) schedules the edge computing unit (ECU). When executing scheduling tasks, the initiator is the Active Scheduling Object (ASO) and the other is the Passive Scheduling Object (PSO).

$$Aso_i = [A_{cpu} A_{mem} A_{net} A_r A_w A_{tasked}]^T \tag{3.4}$$

In equation 3.4, Aso_i represents the i-th active scheduling object (ASO) in the corresponding CCDC or ECN. A_{cpu} represents the remaining CPU resources in ASO, A_{mem} represents the remaining memory in ASO, and A_{net} represents the remaining network bandwidth in ASO. The above three resources are calculated as percentages, with a value range of $(0, 100)$; A_r represents the read status of disk I/O operations in ASO; A_w represents the write status of disk I/O operations in ASO;A_{tasked} indicates whether a scheduling task is being executed by a higher level in the current ASO. The result is a logical value set to (Ture | False), where Ture indicates that the scheduling task is currently being executed and False indicates that it has not been executed.

$$P_{so_j} = [P_{cpu}P_{mem}P_{net}P_rP_wP_{tasked}P_{resp}P_{ltime}P_{size}]^T \tag{3.5}$$

In equation 3.5, P_{so_j} represents the jth passive scheduling object (PSO) corresponding to the ECN or ECU. P_{cpu} represents the remaining CPU resources in PSO, P_{mem} represents the remaining memory resources in PSO, and P_{net} represents the remaining network bandwidth in PSO. Similar to formula 3.4, the above three resources are calculated as percentages, with a value range of (0,100); P_r represents the read status of disk I/O operations in PSO, while P_w represents the write status of disk I/O operations in PSO; P_{tasked} indicates whether the PSO is being scheduled by the previous level, and the result is a logical value set to (Ture | False). Ture indicates that the scheduled task is currently being executed, and False indicates that the scheduled task has not been executed; P_{resp} represents the network state between PSO and ASO; P_{ltime} represents the time when PSO was last successfully executed for scheduling tasks, and P_{size} represents the file size that needs to be scheduled for processing.

$$Task_{ij} = [T_{stime}T_{pri}T_{exectime}]^T \tag{3.6}$$

In equation 3.6, $Task_{ij}$ represents the execution parameters of the active scheduling object Aso_i initiating scheduling tasks to the passive scheduling object P_{so_j}.

The execution steps are as follows:

(1) The preset time length of the system scheduling cycle is $Sche_{time}$, and then the current system time is obtained through system calls as the start time of the scheduling task: T_{stime}.
(2) Calculate the P_{ltime} parameters of the startup time T_{stime} and P_{so_j} to obtain the scheduling priority. The formula is as follows: the larger the values of $T_{pri} = (T_{stime} - P_{ltime}) \div Sche_{time}$ and T_{pri}, the higher the priority and the higher the urgency of scheduling.
(3) Based on the network conditions and P_{size} parameters of P_{so_j}, assuming that the network state between Aso_i and P_{so_j} is in an ideal state, the minimum execution time of the scheduling task can be estimated, with $T_{exectime} = P_{size} \div (A_{net}|P_{net})$, A_{net}, and P_{net} taking the minimum value.
(4) The scheduling strategy uses a relatively simple weighted round robin scheduling algorithm, with T_{pri} as the weight, in order to schedule among the nodes in the edge computing model through polling. Based on the completion status of the scheduled task, set the value of $Task_{ij}$ as the logical value (Ture | False), where Ture indicates that the current task has been completed and False indicates that the scheduled task was not successful.

4. Result analysis.

4.1. Testing Environment. In order to reduce the impact on normal teaching order, a test is conducted before the summer vacation in a studio that is open to students all day. Based on the architecture diagram shown in Figure 4.1, the video capture framework is deployed using the design described in Section 3.5. ECU nodes are deployed, cameras are installed, the network is configured, and an IVSU is constructed. The relevant equipment and main parameters are shown in Table 4.1.

After completing the deployment of IVSU in the studio, the videos collected by IVSU devices are transmitted through wireless networks for data transmission; Edge computing node (ECN) is deployed in the equipment room on the floor; Then, it is transmitted to the Cloud Computing Data Center (CCDC) through the campus network, and the ECN device uses a DELL server. The main parameters are shown in Table 4.2.

During this testing process, the design and deployment of Cloud Computing Data Center (CCDC) was not carried out. A virtual machine (VM) was applied for in the existing cloud computing data center of the school, equipped with corresponding software environment, to simulate CCDC for data storage and scheduling functions. The parameters are shown in Table 4.3.

Table 4.1: Intelligent Video Monitoring Unit Equipment and Its Main Parameters

Serial Number	Equipment (software and hardware)	Parameters
1	Raspberry Pi Single Board Computer(Zere W)	CPU: 1 GHz, single core; Memory: 512 MB Network connection: 802.11 b/g/n; Storage interface: microSD camera interface: CSI; Power interface: micro USB
2	CSI camera (SONY IMX219)	Sensor: 8-megapixel CMOS Static image specifications: 3280 ∗ 2464 (maximum) Video recording specifications: ① 720p 60 fps; ② 1 080 p30 fps
3	storage device	MicroSD: 256G, Kingston KF-C38256-4K I/O performance: ① Read rate: 100 MB/s; ② Write rate: 80 MB/s
4	motionEyeOS	Release: 20190427; Motion: 4 2
5	Wireless router (TL WAR450L)	Transmission frequency band: 2.4 GHz; Transmission rate: 450 M LAN: Gigabit Ethernet port; Wireless gain: 5 dBi
6	exchange board	Huawei, S1700-24GR, 24 port

Table 4.2: Intelligent Video Monitoring Unit Equipment and Its Main Parameters

Option	Parameters/Model	Memo
Server	DELL PowerEdge R510	
CPU Xeon	X5650 * 2	2.60 GHz, single CPU with 6 physical cores, supporting hyper threading
Memory	32G	
Hard disk	1T * 4	Build a RAID-0 array to improve I/O performance
network	1 000 Mbps	
Disk alignment card	Dell PERC H700	512M cache
Operating system	CentOS 7 1810	Kernel version: 3 10 0 957 12 1; Docker version: 1.13.1; Python version: 2.7.5; PHP version: 5.4.16; Database version: MariaDB 5.5.60; Web server: Apache -2.4.6

4.2. Model testing.

4.2.1. Verification of motion detection function. Use IVSU to record a video. Extract the 1057th, 1058th, and 1059th frames for testing. After binarizing the above three frames of images. In the model designed by the author, a three frame difference algorithm is used for motion object detection. The unchanged ones are binarized and transformed into black backgrounds. The motion targets extracted by the algorithm have clear and complete contours, and the results meet the expected motion detection requirements.

4.2.2. Verification of storage requirements. According to the design in the third part, and also to simplify the management of storage files, the video stream is stored in time segments to record video information.

Table 4.3: Main parameters of cloud computing data centers

Option	Parameters/Model
CPU	Xeon E5 2609 v2(2 50 GHz) 8vCPU core
Memory	64 G
Hard disk	2 17TB
network	1 000 Mbps
operating system	CentOS 7 1810

Table 4.4: Video surveillance records

Week	Weekly			
	Week 1	Week 2	Week 3	Week 4
one	160	163	167	54
two	161	163	168	34
three	161	166	171	24
four	161	162	177	20
five	162	162	178	10
six	166	168	180	0
day	167	166	180	0

Take every 300 seconds as a time period, which is 5 minutes, as a video file. When a moving object is detected in the stored video frame, the status of the video recording file is marked and video information is stored. When there are no moving objects in a continuous 5-minute video frame, delete the video file without any motion state to save storage space. The standard opening hours of this studio are from 07:30 in the morning to 23:00 at night. Before deploying the IVSU, investigate the regularity of students entering the studio. Activities in the studio are mainly from 07:30 a.m. to 12:15 a.m. and from 14:10 p.m. to 23:00 p.m, there is less activity during the rest of the time. There is no activity time every day, with an average of about 10 hours and 25 minutes, accounting for approximately 43.30% of the entire day. After a 4-week test from June 10th to July 7th, the first to second weeks in Table 4.4 are the normal teaching weeks of the school; The third week is the pre exam review week; The fourth week is the exam week. The video surveillance records are shown in Table 4.4, where the data represents the number of files and each video file has a length of 5 minutes.

From the above test data, it can be seen that the number of video recording files is directly proportional to student activities. During the normal teaching week of the school, students enter and exit the studio and engage in regular activities within the studio. The amount of video storage remains relatively stable, with a slight increase towards the end of the semester. The third week corresponds to the pre exam review week, and the number of practical activities for students in the studio has increased. The fourth week is the exam week, and there is a significant decrease in activities within the studio. The holiday starts on Friday afternoon in the fourth week, and after the studio is closed, the number of video recording files is recorded as 0. Statistically analyze the data, compare the video surveillance with added motion detection function, and analyze the data from week 1 to week 3 to compare the video surveillance without added motion detection function. The number of video files stored is about 1176 records per week, saving about 41.56% of storage space. Therefore, if applied across the entire school, it can significantly reduce construction costs in terms of storage.

4.2.3. Verification of scheduling tasks. During the test, the intelligent video monitoring system has not been deployed in the whole school for the time being, and the scheduling verification focuses on the test from edge computing node (ECN) to edge computing unit (ECU) to detect the utilization of network bandwidth by the intelligent video monitoring system. Utilizing the lightweight and other features of the Docker container engine, by periodically collecting resource load information such as CPU, memory, disk I/O, and network bandwidth, information support is provided for containerized scheduling schemes. Assuming there are n ECU nodes in total, the ECU list is: $ecu = ecu_0, ecu_1, \cdots, ecu_{n-1}\{\}$, $weight(ecu_j)$ represents the weight of the jth

ECU node, that is T_{pri} calculated in formula 3.6, j also represents the object Pso_j scheduled last time, and $max(ecu)$ represents the maximum value among all nodes. $gcdnumber(ecu)$ represents the maximum common divisor of the weights of all nodes in the ECU list. The variable j is initialized to -1, represents the current weight, and initialized to 0. Increase the corresponding weights, determine the priority of each node, and avoid the data of a certain ECU node not being backed up for a long time. At the same time, through the corresponding weight, the load of network traffic and other conditions are fully considered to avoid the centralized scheduling of ECU nodes in the edge computing model, resulting in network congestion and reducing the overall network construction cost.

5. Conclusion. The scheme proposed by the author introduces the edge computing model to organically integrate the cloud computing data center and edge computing environment in the campus network. The open source and open hardware and software architecture are adopted to make full use of the computing resources of edge computing nodes to realize the motion detection function and effectively reduce the storage space requirements of the monitoring system; And use the Docker containerized platform to collect resource status information from each node, design resource scheduling strategies, and improve the utilization of network bandwidth. Intelligent video monitoring technology is a trend in the development of monitoring technology in the era of big data. Therefore, in subsequent work, the computing power of ECU will be combined to achieve target recognition and tracking functions in intelligent video monitoring systems; Optimize storage space, adopt distributed elastic storage mechanism, and fully utilize the storage capacity of ECN.

REFERENCES

[1] Rajavel, R., Ravichandran, S. K., Harimoorthy, K., Nagappan, P., & Gobichettipalayam, K. R. (2022). IoT-based smart healthcare video surveillance system using edge computing. Journal of ambient intelligence and humanized computing, 13(6), 3195-3207.

[2] Wan, S., Ding, S., & Chen, C. (2022). Edge computing enabled video segmentation for real-time traffic monitoring in internet of vehicles. Pattern Recognition, 121, 108146.

[3] Xu, X., Wu, Q., Qi, L., Dou, W., Tsai, S. B., & Bhuiyan, M. Z. A. (2020). Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles. IEEE Transactions on Intelligent Transportation Systems, 22(3), 1787-1796.

[4] Wang, F., Zhang, M., Wang, X., Ma, X., & Liu, J. (2020). Deep learning for edge computing applications: A state-of-the-art survey. IEEE Access, 8, 58322-58336.

[5] Yar, H., Imran, A. S., Khan, Z. A., Sajjad, M., & Kastrati, Z. (2021). Towards smart home automation using IoT-enabled edge-computing paradigm. Sensors, 21(14), 4932.

[6] Ke, R., Zhuang, Y., Pu, Z., & Wang, Y. (2020). A smart, efficient, and reliable parking surveillance system with edge artificial intelligence on IoT devices. IEEE Transactions on Intelligent Transportation Systems, 22(8), 4962-4974.

[7] Yu, J., Chen, M., Zhou, H., Wang, L., Luo, H., & Lan, J. B. (2022). Research on application of edge calculation in power grid state prediction. 2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST), 651-655.

[8] Song, W., Tang, Y., Tan, W., & Ren, S. (2023). Ishd: intelligent standing human detection of video surveillance for the smart examination environment. Computer Modeling in Engineering and Science (in English), 7(8), 6722-6747.

[9] Isaac Martín de Diego, Alberto Fernández-Isabel, Ignacio San Román, Conde, C., & Cabello, E. (2022). Novel context-aware methodology for risk assessment in intelligent video-surveillance systems. International Journal of Sensor Networks, 13(5), 118.

[10] Khosravi, M. R., & Samadi, S. (2022). Mobile multimedia computing in cyber-physical surveillance services through uav-borne video-sar: a taxonomy of intelligent data processing for iomt-enabled radar sensor networks. Tsinghua Science and Technology, 27(2), 288-302.

[11] Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., & Zomaya, A. Y. (2020). Edge intelligence: The confluence of edge computing and artificial intelligence. IEEE Internet of Things Journal, 7(8), 7457-7469.

[12] Lv, Z., Chen, D., Lou, R., & Wang, Q. (2021). Intelligent edge computing based on machine learning for smart city. Future Generation Computer Systems, 115, 90-99.

[13] Hua, H., Li, Y., Wang, T., Dong, N., Li, W., & Cao, J. (2023). Edge computing with artificial intelligence: A machine learning perspective. ACM Computing Surveys, 55(9), 1-35.

[14] Zhang, X., Cao, Z., & Dong, W. (2020). Overview of edge computing in the agricultural internet of things: Key technologies, applications, challenges. Ieee Access, 8, 141748-141761.

[15] Jiang, X., Yu, F. R., Song, T., & Leung, V. C. (2021). A survey on multi-access edge computing applied to video streaming: Some research issues and challenges. IEEE Communications Surveys & Tutorials, 23(2), 871-903.

[16] Guo, H., Liu, J., Ren, J., & Zhang, Y. (2020). Intelligent task offloading in vehicular edge computing networks. IEEE Wireless Communications, 27(4), 126-132.

[17] Qiu, T., Chi, J., Zhou, X., Ning, Z., Atiquzzaman, M., & Wu, D. O. (2020). Edge computing in industrial internet of things: Architecture, advances and challenges. IEEE Communications Surveys & Tutorials, 22(4), 2462-2488.
[18] Kong, X., Wang, K., Wang, S., Wang, X., Jiang, X., Guo, Y., ... & Ni, Q. (2021). Real-time mask identification for COVID-19: An edge-computing-based deep learning framework. IEEE Internet of Things Journal, 8(21), 15929-15938.
[19] Zhou, X., Xu, X., Liang, W., Zeng, Z., & Yan, Z. (2021). Deep-learning-enhanced multitarget detection for end–edge–cloud surveillance in smart IoT. IEEE Internet of Things Journal, 8(16), 12588-12596.
[20] Hartmann, M., Hashmi, U. S., & Imran, A. (2022). Edge computing in smart health care systems: Review, challenges, and research directions. Transactions on Emerging Telecommunications Technologies, 33(3), 3710.

Edited by: Bradha Madhavan
Special issue on: High-performance Computing Algorithms for Material Sciences
Received: Jun 10, 2024
Accepted: Jul 19, 2024