Received November 4, 2021, accepted November 17, 2021, date of publication November 19, 2021,
date of current version November 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3129465

DDPG-Based Edge Resource Management for Coal Mine Surveillance Video Analysis in Cloud-Edge Cooperation Framework

ZHI XU AND JINGZHAO LI
College of Electrical and Information Engineering, Anhui University of Science and Technology, Huainan 232000, China
Corresponding author: Jingzhao Li (ljzaust@outlook.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 51874010, in part by the Key Technology Research and Innovation Team Project under Grant 201950ZX003, and in part by the Huaibei Mining Group National Technology Center Project "Coal Mine Control System Based on Video Monitoring."

ABSTRACT Intelligent video surveillance is important to ensure production safety in coal mines, while cloud-edge cooperation is an effective means to improve the performance of intelligent video monitoring. However, in edge layers, incorrect resource allocation of computing and network resources will result in the waste of resources and low real-time performance. In this paper, a DDPG-Based (Deep deterministic policy gradient-based) edge resource allocation method for cloud-edge cooperation framework is proposed. Firstly, the cloud-edge cooperation framework is designed for different tasks. Secondly, the joint minimizing problem of latency and bandwidth usage caused by edge computing is modeled. To quickly solve the joint optimization problem, we convert it to MDP (Markov Decision Process). In addition, ESPN (Edge status perception network) is proposed, which enhances the ability of feature perception and action output of DDPG. Finally, DDPG-ESPN is proposed to solve the joint optimization problem. Simulation results show that compared with other methods, DDPG-ESPN improves the real-time performance and bandwidth usage by up to 18.88% and 42.81% respectively.

INDEX TERMS Edge resource allocation, intelligent video surveillance, deep deterministic policy gradient, edge status perception network.

I. INTRODUCTION

Intelligent video surveillance plays an important role in coal mine safety production. For example, when workers are too close to the equipment in operation, the intelligent video surveillance system can promptly issue an alarm or shut down the equipment to ensure the safety of the personnel. Currently, the ANN-Based (Artificial Neural Network) AI model is an effective method to process surveillance video. However, the ANN-Based model for the video process usually needs a large number of parameters and calculations. Therefore, the traditional method of coal mine intelligent surveillance needs to transmit the monitoring video to cloud server for intelligent analysis, and then return the results to the monitoring terminal. This process causes serious latency, and occupies network congestion. It makes the intelligent processing of surveillance video lose the ability of real-time response, and cannot effectively guarantee safety in coal mines.

Edge computing [1] can improve the real-time performance of intelligent processing and reduce network congestion by performing operations close to the data source. Meanwhile, with the continuous development of embedded microprocessors, the computing and storage capabilities of edge computing have been improved to a certain extent, which enables the edge to complete more intelligent processing tasks. Processing surveillance video at the edge can improve the real-time performance of intelligent video surveillance, which is of great significance to the safety of workers in coal mines.

To solve the problems of large-scale parameters and calculations required for intelligent video processing based on ANN, various neural network lightweight technologies have been proposed. Google effectively reduces the amount of calculation and the number of parameters required for convolution operations by adopting depthwise separable

The associate editor coordinating the review of this manuscript and approving it for publication was Daniel Grosu.

convolution [19]. Model pruning deletes the neurons that have little effect in neural networks. Weight quantization reduces the number of bits of the parameter by quantifying the weight in AI models [20]. Wang *et al.* used layer-level and channel-wise pruning methods to prune the YOLOv3 (You only look once verson3) model, which effectively enhanced the calculation speed of the YOLO model and reduced the number of its parameters [21]. The above methods can reduce the size and computational complexity of AI models from different aspects, and enable intelligent video processing to be implemented on embedded platforms. However, the compression of the neural network will result in a loss of accuracy. Moreover, the real-time performance of intelligent computing supported by the currently embedded platform is still insufficient because of computing capacity.

Cloud-Edge cooperation [2]–[7] combines the advantages of cloud computing and edge computing. Cloud computing has rich computing and storage resources, which can support AI models inference and training, but its real-time performance is limited by various environments. Edge computing has fewer computing and storage resources, but deploying lightweight AI models on edge can improve real-time performance and reduce network bandwidth usage. Therefore, in the cloud-edge cooperation framework, cloud computing is usually used to provide non-real-time services such as model training and data storage, while edge computing is used to process tasks with high real-time requirements such as inference.

Coal mine surveillance video intelligently processed by edge computing can improve the real-time performance and reduce bandwidth usage caused by video transmission. The deployment density of video surveillance in coal mines is greater than that on the ground. However, the pedestrians in coal mines are sparse, and fewer areas require high-real-time intelligent surveillance. Moreover, considering the distributed arrangement of edge nodes in coal mine, the network and computing resources need to be allocated and managed for real-time surveillance video processing. Traditional management methods of edge computing and network resources usually include average allocation or non-allocation. Those methods resulting in insufficient real-time performance in ''busy'' areas with pedestrians, and too high real-time performance in ''idle'' areas without pedestrians. Hence, the edge resource is wasted by the traditional methods, and it is necessary to propose an edge resource allocation method to optimize the latency and bandwidth occupation of edge computing. To reasonably allocate edge layer computing and network resources, appropriate edge resource allocation methods must be adopted.

DQN (Deep Q-Learning) is a DRL-Based (Deep Reinforcement Learning Based) method, which is often used in various resource allocation fields [11]–[18]. However, DQN is difficult to handle continuous action space. Hence, the DDPG-Based method which can output continuous action is used in some fields that require higher fine-graininess. The Actor and critic network of the traditional DDPG algorithm is constituted by FCN (Fully Connected Network). However, there are two drawbacks of Fully Connected Network. The first is the requirement of a large of amount parameters, which makes it difficult to store. Secondly, the feature perception ability of Full Connected Network for a larger matrix is weaker than that of CNN (Convolutional Neural Network).

Traditional cloud-edge cooperation frameworks do not consider the full use of computing resources of the edge layer, and the cooperation between edge nodes and edge nodes. These methods may cause a waste of resources and reduce the real-time performance of the system. However, the way of cooperation between edge nodes is also a problem. It is a big challenge to make edge computing resources adjust with the location and quantity of ''busy'' area, while maintaining the real-time performance of ''idle'' area, and minimizing the network bandwidth usage by video transmission as much as possible.

To solve the abovementioned problems, we proposed a DDPG-Based edge resource allocation method in the cloud-edge computing environment. In the Cloud-Edge framework, cloud computing is used to train various edge models and provide data storage services, while edge computing is used to provide real-time intelligent inference for surveillance video. To the problem of joint minimizing the latency and bandwidth usage in edge layer, we first convert it into MDP (Markov Decision Process), and solve it in continuous action space by the use of DDPG. Meanwhile, we built ESPN (Edge Status Perception Network) with fully convolutional network and residual structure, to improve the edge environment feature perception ability of DDPG. The main contributions of this paper are as follows.

(1) The framework of Cloud-Edge cooperation is designed. The non-real-time tasks are handed in by cloud computing. Edge computing is responsible for real-time surveillance video inference.

(2) Based on the edge resource of coal mine environment, the latency and bandwidth usage problem has been modeled. In addition, the problem of minimizing latency and bandwidth usage is converted to MDP.

(3) The improved DDPG algorithm DDPG-ESPN is proposed. The ability of edge feature perception and action output is promoted by ESPN.

The remainder of this paper is organized as follows. Related work about Cloud-Edge cooperation and DRL-Based resource allocation method is introduced in Section II. In Section III, a Cloud-Edge cooperation framework is proposed. In Section IV, We model the problem of joint minimizing latency and network usage. Section V converts the minimizing problem to MDP. Meanwhile, the ESPN and DDPG-ESPN method is proposed to quickly solve the joint minimizing problem. Section VI gives simulation and comparison of the proposed method. Section VII concludes this article.

II. RELATED WORK

A. CLOUD-EDGE COOPERATION

Edge computing solves the problems of high latency and network congestion caused by cloud computing [1], [23]. It greatly improves the real-time performance by performing operations close to the data source. However, due to the constraint resources, single edge computing is not suitable for large-scale services that require an amount of computing and storage resources, such as model training and data storage. Therefore, in the field of intelligence monitoring, Cloud-Edge cooperation is a hot topic to promote real-time performance and accuracy. Wang *et al.* proposed to use the Cloud-Edge computing framework in Cyber-Physical-Social. They push the long-term and large-scale tasks to cloud server, while edge computing is used to process small-scale and short-term tasks [2]. Cloud-Edge cooperation provides a smoothly environment for intelligent video surveillance. Wang *et al.* [3] proposed a CNN-Based intelligent visual sorting system in cloud-edge computing framework. The cloud server is used as a model training and algorithm design platform, and edge computing is used to process monitoring video for quickly sorting productions. Wang *et al.* proposed a smart surface inspection system in cloud-edge computing environment. Based on the advantages of cloud computing and edge computing, this paper builds an SPSS architecture, which can complete video detection tasks with high accuracy and high detection speed [5]. Ahn *et al.* proposed a new cloud-side interaction framework for IoT video analysis, which improved the efficiency of cloud-side computing and the real-time performance of edge tasks [4]. Rajavel *et al.* [6] proposed a Iot-Based smart healthcare video surveillance system, it incorporate the edge computing in the gateway level for minimizing network usage and response time between edge and cloud. However, the method they proposed is not considered the task offloading between edge nodes and edge nodes, it may waste the edge resources and increase the response time. Jayaram and Prabakaran [7] use the edge-cloud system to protect personal privacy of patients in the remote healthcare monitoring system. The edge layer is used for secure data processing and filtering data, to minimizing the response time and bandwidth usage, while the cloud is used for prediction and rehabilitation of remote patients. But the network environment of the system they proposed is smoothly, it is not suitable for coal mine environment. It can be seen that Cloud-Edge cooperation can integrate the advantages of cloud computing and edge computing to improve service quality. The abovementioned cloud-edge cooperation method is proposed to solve practice problem. However, the edge environment in those papers did not consider the cooperation between edge nodes, which may lead to the waste or shortage of edge computing capacity in monitoring area.

Surveillance video processed intelligently by edge computing can improve real-time performance and accelerate the ability to respond to emergencies. Lightweight processing of neural networks and deploying them in edge environments has become a hot topic of current research work. Ren *et al.* [24] provide real-time object detection services at edge based on edge computing. He *et al.* [25] prune the convolutional filter of the model by the ASFP method, to solve the information loss caused by typical pruning algorithms. Li *et al.* [26] proposed a compression method for CNN to reduce the cost of computation. Jian-Hao *et al.* [27] proposed Thinet framework to realize compress and speedup of CNN models. Although the lightweight model deployed on the edge platform reduces the requirements for computing capacity, it is difficult to maintain the accuracy of the model by pruning the model and quantifying the parameters. In order to improve the efficiency of the edge cloud and minimize the delay of mobile devices, Ren *et al.* [8] studied the collaboration between cloud computing and edge computing. By solving the resource allocation problem, they greatly reduced the processing delay of the task. However, the cloud computing have not been fully applied to improve the quality of edge computing. Yang *et al.* [9] proposed an online video quality and computing resource allocation algorithm to enable low-latency and high-accuracy analysis of urban surveillance video, but the edge intelligent model cannot be evolved and is difficult to adapt to a variety of environments. Hung *et al.* [10] proposed VideoEdge architecture. In this paper, the best balance point between the resources and accuracy of surveillance video processing is found in the cloud-side collaboration environment. Meanwhile, they proposed the "Pareto band" method to reduce the video time search space, and ultimately, greatly improve the processing speed and accuracy of urban surveillance video. However, the method they proposed did not consider the cooperation between nodes with computing power. The above cloud-edge collaborative video processing methods have improved the real-time and accuracy of surveillance video to a certain extent. However, these methods are difficult to be practically applied in coal mines with limited communication and computing environments. Moreover, the above method does not take into account the cooperation between edge devices. When performing video processing, the computing capacity of single edge device cannot support higher real-time performance, or the computing power of the edge device is wasted.

B. DRL-BASED RESOURCE ALLOCATION

Real-time processing of surveillance video through edge computing has been applied in various fields. However, due to the characteristics of video surveillance in coal mines, the deployment of the above-mentioned methods on edge computing devices will result in insufficient real-time performance in areas that need to be monitored, and waste computing power in areas that do not require high real-time performance. Meanwhile, a large number of video transmissions in the edge network will cause network congestion. Therefore, the computing and network resources of the edge layer need to be allocated reasonably.

DRL fits the action function or value function through deep neural networks, so that it can achieve better results in strategy optimization. Hence, the DRL method has a wide range of applications in edge resource allocation. DQN is a Value-Based RL method [35]. Cao *et al.* [11] used the method of combining BPNN and DQN to allocate resources to the MINO-NOMA system, and achieved better results in bandwidth utilization using this method. However, this resource allocation method did not consider the computing latency of edge device. Wu *et al.* [12] proposed the pure-DQN method and the hybrid DQN method to solve the mixed integer problem of MEC resource allocation. Experiment results show, the hybrid method combines the advantages of QL and convex optimization. But from another perspective, this method is not suitable for fine-grained tasks such as edge video process. Sun, *et al.* use the DQN method to balance the energy consumption and user satisfaction issues in the C-RANs system [13]. By improving DQN, [14]–[17] allocate network resources and computing resources for edge computing, so that the delay in the system is lower. DQN-Based resource allocation methods enable edge resources to be more reasonably allocated to different tasks. The monitoring video processed by edge devices requires Fine-grained allocation of edge computing resources. However, it is unable to allocate resources in a more Fine-grain, because DQN is difficult to perform actions in continuous action space.

DDPG [22] is a policy-based RL method. Compared with DQN, DDPG has the ability to output continuous actions. At the same time, the convergence of DDPG is also better than that of DQN. Chen *et al.* enhanced the timing feature extraction capabilities of DDPG through TFEN, and at the same time used rPER to optimize the empirical playback pool to accelerate and stabilize the convergence of the model [28]. But on the other hand, the TFEN composed of Conv-1D and LSTM has a weak situational awareness of large scale edge environment. Wu *et al.* [29] use Lyapunov optimization to convert CMDP to MDP, and then use DDPG-Based method to solve the optimization problem, so that the real-time and long-term accuracy of IoT devices can be guaranteed. Peng *et al.* [30] use the DDPG-Based method to manage the available spectrum, computing and buffer resources of base stations and UAVs, which can achieve lower delay and quality of service compared to random solutions. However, the UAVs have strong mobility, therefor they did not consider the movement of users. Zhang *et al.* [31] use the DDPG method to manage the communication mode and resources of the heterogeneous cellular network, which improves the energy efficiency of the D2D heterogeneous network. However, the computing latency and accuracy of edge devices is not considered in this paper. Based on the DDPG method, Qiu *et al.* [32] manage the energy and data transmission of field wireless sensors, which improves the life and data transmission rate of field wireless sensors. As a policy-based RL method, A3C (Asynchronous-Advantage-Actor-Critic) is also used in various resource allocation environment. J. Zou *et al.* [33] use the A3C algorithm to control YACs for managng the network structure of MEC (Mobile Edge Computing). This method is proposed to minimizing the task execution latency and power consume. However, in the coal mine environment, this method is difficult to optimize the fixed network structure and mobile service object. Tuli *et al.* [34] use the quickly adapt dynamic scenarios advantage of A3C to reduce the power consume of MEC, and the Residual Recurrent Neural Network is also used to quickly update the model parameters. However, the algorithm they proposed require a large amount of calculations, it is not suitable for edge environment with constrained resources. The policy-based RL method has achieved a wide range of applications and excellent results in edge computing resource management. However, for the real-time intelligent processing of surveillance video in coal mines based on edge computing, the above methods are difficult to adapt to the actual environment of coal mines. Moreover, the above articles did not consider the computing latency caused by video processing at edge. In addition, the traditional DDPG method has insufficient ability to perceive environmental characteristics, and cannot be applied to the areas with densely arranged edge nodes and large random distribution of personnel such as coal mines.

III. CLOUD-EDGE COOPERATION FRAMEWORK

A. REAL-TIME EDGE SERVICE

Surveillance video is transmitted and processed by edge nodes. Compared with the cloud server, it has better real-time performance and can reduce the bandwidth usage occupied by video transmission. On the one hand, the real-time edge service performs real-time intelligent analysis of surveillance video, which can respond to emergencies in coal mines more quickly. On the other hand, computing resources are allocated to the "busy" area to improve real-time performance by edge services. When performing real-time intelligent surveillance at the edge, the system dispatches the computing resources in the "idle" area to the "busy" area. Meanwhile, in order to ensure the system's ability to respond to emergencies, the "idle" areas still need to retain a certain amount of computing power. In short, the "busy" areas set the upper limit of the real-time performance, while the "idle" areas set the upper limit of real-time performance.

In this paper, the edge nodes are composed of embedded microprocessors. For the sake of simplicity, we assume that each edge node processes the data of a camera, and the edge node can control the surrounding equipment, alarms, etc. Considering that the construction time of different roadways in coal mines are different, the computing power of edge nodes are also different. We choose the edge node with the strongest computing power as the management node. In addition to completing the work of general nodes, the management node also needs to run a resource allocation algorithm to schedule the computing resources of other nodes within a certain range. The framework of the edge layer is shown in Fig. 1.

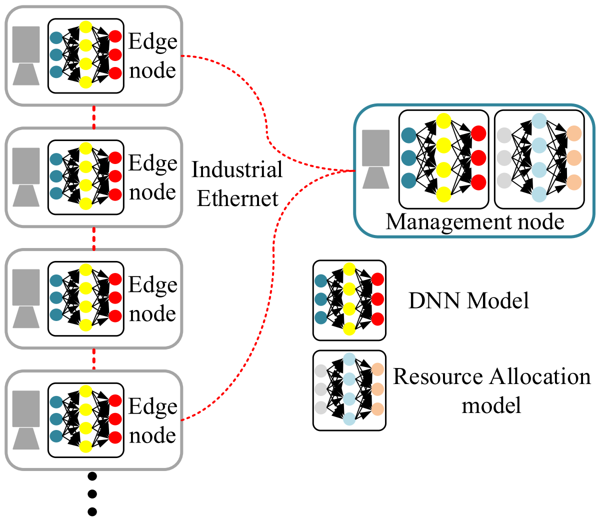


FIGURE 1. The framework of edge layer. All edge nodes are connected to the industrial ethernet. The management node is used to run the resource allocation model for managing the edge nodes.

B. NON-REAL-TIME CLOUD SERVICE

Cloud server has abundant computing and storage resources. However, the cloud-centric computing model needs to transmit data to cloud server for processing, and then return the result. This process cause severe latency and network congestion. Therefore, Cloud computing is difficult to meet the tasks that require high real-time performance. Although it is difficult for a cloud server to handle high-real-time tasks, its abundant resources can provide non-real-time tasks such as data storage and model training. Performing model training in the cloud and deploying the trained model on the edge for real-time inference can combine the advantages of cloud computing and edge computing. In this paper, cloud computing is not only used to train the AI model for video inference, but also responsible for training the DDPG-ESPN model. Meanwhile, a large amount of status-action data is transmitted to the cloud server for composing experience replay buffer. In addition, the DDPG-ESPN approach only needs to deploy the actor network at the edge. The functions of non-real-time cloud service are shown in Fig. 2.

The Edge-Cloud cooperation framework divides the tasks in the system into real-time tasks and non-real-time tasks. The real-time tasks are processed by the edge computing. In the proposed system, surveillance video is processed by edge devices in real-time. Therefore, edge devices only need to send the intelligent processing results and the emergency video to the cloud server for monitoring. Moreover, the intelligent models, such as object detection and edge resource allocated model, are trained by cloud server and then transmitted to edge devices through Industrial Ethernet. In the proposed Edge-Cloud cooperation framework, the data interaction between the cloud and edge is delay tolerable. Hence, the performance of the edge layer for intelligent surveillance video processing is not affected by the data transmission latency between cloud and edge.

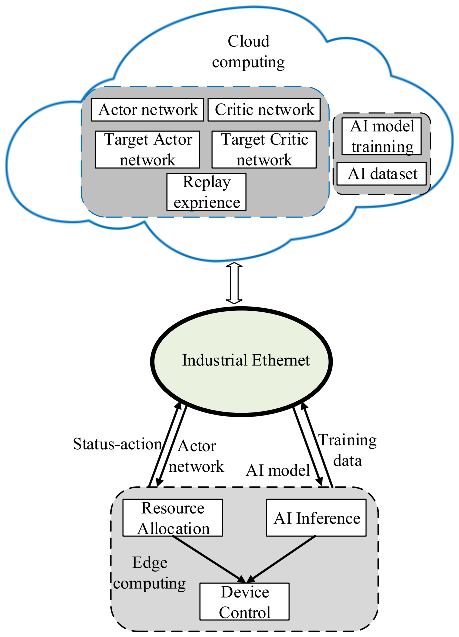


FIGURE 2. The functions of non-real-time cloud service. The trained AI model and actor network are transmitted to the edge layer through industrial ethernet for AI inference and resource allocation at the edge layer, and the edge layer provides the data required to train models.

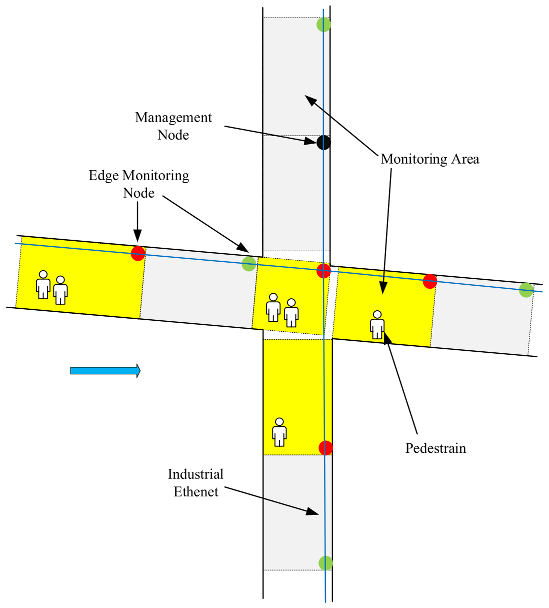


FIGURE 3. The work schematic diagram of roadway edge nodes. Where, the yellow area denotes that the edge node has detected pedestrian by AI model, otherwise the area is represented in gray. Edge nodes with pedestrian areas are represented in red, otherwise they are represented in green, and management nodes are represented in black. The above nodes are all connected to the Industrial Ethernet.

IV. SYSTEM MODEL AND PROBLEM FORMULATION

A. SYSTEM DESCRIPTION

The edge layer of coal mine video surveillance system mainly responsible for the real-time processing of surveillance video. In this paper, the edge resource is allocated by the management node with an allocation algorithm. The work schematic diagram of roadway edge nodes is shown in Fig. 3.

In the roadway environment shown in Fig. 3, edge nodes are all powered by wired. In the traditional method, even if there are no pedestrians in the "idle" area, all the computing power of the edge node is still used to process the surveillance video in the area. However, the real-time performance is usually insufficient because the limited computing capacity of the edge nodes which deployed on the "busy" area with pedestrian. Meanwhile, there is still a serious delay in transmitting the video in this area to the cloud server for processing, and it will take up a lot of bandwidth. Therefore, we use the remaining computing power of idle nodes to assist busy nodes in intelligent analysis. At the same time, "idle" nodes also need to leave a certain amount of computing power to maintain intelligent surveillance of "idle" areas, so as to deal with the pedestrians that may appear in the "idle" area. In addition, it is necessary to minimize video transmission on the network to prevent network congestion.

In this system, we need to set the upper and lower limits of the intelligent surveillance latency. We assume that when there are pedestrians in a surveillance area, the latency of intelligent video surveillance in this area is not higher than D_{busy}. When there are no pedestrians in the surveillance area, the latency is not higher than D_{idle}. The main parameters and symbols used in this paper are summarized in Table 1.

B. NETWORK MODEL

When the edge layer process the surveillance video, it is necessary to allocate the computing resources of each edge node. In case the computing power of "busy" nodes is not enough to support the real-time demand in areas with pedestrians, a part of the video that cannot be processed in time by "busy" nodes needs to be transmitted to "idle" nodes through Industrial Ethernet.

We define matrix $B(t)$ as the video transmission matrix in time slot t:

$$B(t) = \begin{bmatrix} 0 & b_{12}^t & \dots & b_{1k}^t \\ b_{21}^t & 0 & \dots & b_{2k}^t \\ \dots & \dots & \dots & \dots \\ b_{k1}^t & b_{k2}^t & \dots & 0 \end{bmatrix} \tag{1}$$

where, b_{ij}^t $(i, j \in [1, 2, \dots, k]$, k is the total number of edge nodes) represents the percentage of video frames transmitted from the surveillance video of the i-th surveillance area to the j-th edge surveillance node. According to the actual scene of video transmission in the system, the diagonal element of the matrix is 0, and when $b_{ij}^t \neq 0$, $b_{ji}^t = 0$.

For the sake of simplicity, we only consider the video transmission in the Industrial Ethernet of the system, and the video of each monitoring area is transmitted through Industrial Ethernet when transmitting between nodes, so the video transmission delay of the i-th monitoring area is:

$$db_i = \frac{\sum_{j=1}^{k} b_{ij}^t \cdot f_i}{S} I \tag{2}$$

where, f_i is the total number of video frames of the i-th monitoring area, $f_i = {}^1/_{d_i}$. d_i denotes the latency of intelligent video processing of i-th surveillance areas. I is the data volume of each frame of video, and S is the network transmission speed.

TABLE 1. Table of parameters and symbols.

Parameters	Description
D_{busy}	Upper latency limit of busy area.
D_{idle}	Upper latency limit of idle area.
$B(t)$	The video transmission matrix, it consists of b_{ij}^t .
b_{ij}^t	The percentage of video frames transmitted from the surveillance video of the *i-th* surveillance area to the *j-th* node.
db_i	The transmission delay of i-th monitoring area.
$C(t)$	The computing distribution matrix, it consists of c_{ij}^t .
c_{ij}^t	The percentage of computing power used by the *i-th* node to assist the *j-th* node.
dc_i^{local}	The computing delay caused by the local computing of the *i-th* surveillance area
H_m	The amount of calculation required for the model to process the data of each byte image.
dc_{ji}^{off}	The average computing latency for the *i-th* node to process the video frame sent by the *j-th* node.
d_{ji}^{off}	The total latency of each frame.
d_i^t	The latency of intelligent video surveillance if *i-th* area.
$D(t)$	The latency state, it consists of d_i^t .
$\alpha(t)$	The coefficient of busy area.
$\beta(t)$	The coefficient of idle area.
δ	The weight of the video transmission bandwidth in the network. we set $\delta = 0.5$
$A(t)$	The action matrix, it consists of a_{ij} .
a_{ij}	The increase in the percentage of the frame number that the *i-th* node assists in the *j-th* area.
θ^π	The parameters of actor net
ω^π	The parameters of critic network

C. COMPUTING MODEL

1) LOCAL COMPUTING

The intelligent processing of coal mine surveillance video can be divided into two parts. The first part is processed by local edge nodes. While the second part will be processed by other edge nodes if the computing capacity of local edge nodes is insufficient. We define the computing distribution matrix $C(t)$ to represent the percentage of surveillance video computing power undertaken by each edge node:

$$C(t) = \begin{bmatrix} c_{11}^t & c_{12}^t & \dots & c_{1k}^t \\ c_{21}^t & c_{22}^t & \dots & c_{2k}^t \\ \dots & \dots & \dots & \dots \\ c_{k1}^t & c_{k2}^t & \dots & c_{kk}^t \end{bmatrix} \tag{3}$$

where, c_{ii}^t represents the percentage of computing power used by the i-th edge node when processing the monitoring video of itself. c_{ij}^t represents the percentage of computing power used by the i-th edge node to assist the j-th edge monitoring node.

Therefore, the computing delay caused by the local computing of the surveillance video of the i-th surveillance area is:

$$dc_i^{local} = \frac{I \cdot H_m}{c_{ii}^t \cdot H_i} \tag{4}$$

where, H_m denotes the amount of calculation required for the model to process the data of each byte image. H_i represents the calculation speed of the i-th node.

2) OFFLOADING COMPUTING

When transmitting video frames to other nodes for intelligent inference, not only the network transmission delay and computing latency must be considered, but also the computing tasks of the offloaded node itself. Regardless of whether there are pedestrians in the area monitored by the offloaded node, the node needs to perform calculations on the local surveillance video and the offloaded video of other nodes. Therefore, when an edge node is processing video frames sent by other nodes, it must first process the video frames that it needs to analyze, and cross-process the local surveillance video frames and other video frames in a certain proportion. In this way, the real-time performance of local video and offloaded video can be ensured at the same time. The process is shown in Fig. 4.

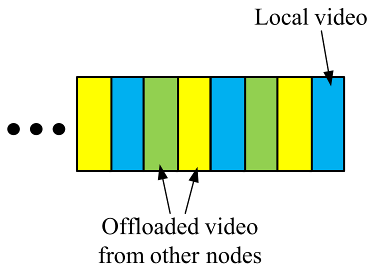


FIGURE 4. The edge node processes the offload video and the local video.

The ratio of the video frame that needs to be inference when the i-th node is offloaded to the j-th node and all the video frames that need to be inference by the i-th node itself is:

$$c_{ij}^t = \frac{b_{ji}^t \cdot f_j}{f_i + \sum_{j=1}^{k} b_{ji}^t \cdot f_j} \tag{5}$$

Therefore, the average computing latency for the i-th node to process the video frame sent by the j-th node is:

$$dc_{ji}^{off} = \frac{I \cdot H_m}{c_{ij}^t \cdot H_i} \tag{6}$$

From the above, the transmission latency of the j-th node transmitting each video frame to the i-th node is:

$$db_{ji} = \frac{\sum_{j=1}^{k} b_{ji}^t . f_j}{S} I \tag{7}$$

Therefore, when the j-th node transmits the video frame to the i-th node for calculation, the total latency of each frame is:

$$d_{ji}^{off} = dc_j^{off} + db_{ji} \tag{8}$$

For node j, when the computing resources cannot meet the real-time requirements, a part of the surveillance area video is analyzed locally, while the other part needs to be transmitted to other nodes for inference. Therefore, the average delay for video frame processing is:

$$d_j = dc_j^{local} + \sum_{i=1}^{k} b_{ji}^t d_{ji}^{off} \tag{9}$$

We define the latency matrix $D(t)$ to represent the latency state of the edge layer:

$$D(t) = \begin{bmatrix} d_1^t & 0 & \dots & 0 \\ 0 & d_2^t & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & d_k^t \end{bmatrix} \tag{10}$$

where, d_i^t denotes the latency of intelligent video surveillance if i-th area in time slot t.

D. PROBLEM FORMULATION

To improve the real-time performance of surveillance video in ''busy'' areas for ensuring the safety of personnel. Coal mine edge nodes process the surveillance video in real-time, and it needs to quickly schedule the edge layer resources according to the edge computing and network environment. Meanwhile, for the ''idle'' area, a certain degree of real-time should also be maintained so that the area has the ability to deal with emergencies. However, due to the limited computing capacity of the edge layer and the pursuit of lower latency, it is necessary to set a higher weight to the ''busy'' area while also taking into account the ''idle'' area. In addition, in order to allow more network resources to transmit important information, the amount of video broadcast on the network should be as small as possible. Therefore, in the time slot t, for the resource allocation problem of intelligent video surveillance nodes, we describe it as a joint minimizing problem of latency and bandwidth usage:

$$\min[\sum_{i=1}^{k}\alpha(t)(D_{busy}-d_i)^2+\sum_{i=1}^{k}\beta(t)(D_{idle}-d_j)^2 + \delta\sum_{i=1,j=1}^{k}\frac{b_{ij}^t}{d_i}] \tag{11}$$

$$s.t.\ C1: b_{ij}^t \geq 0 \tag{11.a}$$

$$C2: \sum_{j=1}^{k} b_{ij}^t < 1 \tag{11.b}$$

$$C3: \sum_{j=1}^{k} c_{ij}^t \leq 1 \tag{11.c}$$

$$C4: d_i < D_{busy} \tag{11.d}$$

$$C5: d_j < D_{idle} \tag{11.e}$$

$$C6: \sum_{i=1}^{k}\frac{I\cdot H_m}{d_i} \leq \sum_{i=1}^{k} H_i \tag{11.f}$$

In (10), When there are pedestrians in area i at time slot t, $\alpha(t) = e^{-(\mathrm{d}_i - D_{busy})}$, otherwise $\alpha(t) = 0$. When there are no pedestrians in area i at time slot t, $\beta(t) = 0.5e^{-(d_i - D_{idle})}$, otherwise $\beta(t) = 0$. δ is the weight of the video transmission bandwidth in the network, we set $\delta = 0.5$. C1 and C2 denote that the surveillance video cannot be completely transmitted to other nodes for processing. C3 denotes that the task assigned to the edge node will not exceed its computing capacity. C4 and C5 represent that the real-time processing of surveillance video needs to reach the upper and lower limits. C6 denotes that the total computing power used for video operations cannot exceed the actual total computing capacity of edge layer.

V. DDPG-BASED EDGE RESOURCE ALLOCATION

A. MDP FORMULATION

In order to quickly solve the joint minimization problem, we first convert it to MDP, and then use the DDPG-Based method to solve it. The agent deployed with DDPG-Bsed method generates an action $a(t)$ based on the edge layer status $s(t)$ via a policy function $\pi(t)$. Each action will generate a reward to evaluate the action.

1) STATUS SPACE

Each edge node sends its computing state, network state, and pedestrian state to the management node at every time slot. Where, the computing state of each node is converged into computing power distribution matrix $C(t)$ and latency matrix $D(t)$. The network state of each edge node is represented by $B(t)$. Pedestrian status is used to represent whether there are pedestrians in each monitoring area in time slot t, which denoted by matrix $P(t)$. Matrix $P(t)$ is a diagonal matrix, if a pedestrian is detected in i-th area, the diagonal element $p(t)_{ii} = 1$, otherwise $p(t)_{ii} = 0$. The status of edge layer is acquired by management node in the time slot t. We define the status matrix $s(t) \in S$:

$$s(t) = \{C(t), D(t), T(t), P(t)\} \tag{12}$$

where, the dimension of status matrix is $4 \times k \times k$.

2) ACTION SPACE

According to the actual state of the edge layer, the agent selects and executes actions based on policy π. When the pedestrian state changes, the computing and network resources of the edge layer need to be reallocated. The agent allocation computing and network resources of edge layer by execute actions. Action matrix $A(t)$ is defined to represents the actions taken by the agent in time slot t:

$$A(t) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{bmatrix} \tag{13}$$

where, a_{ij} represents the increase in the percentage of the frame number that the i-th node assists in the j-th area for video inference. When $a_{ij} > 0$, a_{ij} percentage video frames of j-th node will be added to i-th node for inference. When $a_{ij} < 0$, $|a_{ij}|$ percentage video frames of j-th node which analyzed by i-th node will be reduced.

3) REWARDS

When the management node takes action $A(t)$ according to $s(t)$, the management node will immediately get the reward value $r(t)$ according to the edge layer state. In the training phase, policy π continues to iterate according to the reward value until it converges. In order to minimize the latency of intelligent video surveillance and bandwidth usage, the reward is defined as:

$$r(t) = \frac{1}{[\sum_{i=1}^{k}\alpha(t)(D_{busy}-d_i)^2+\sum_{j=1}^{k}\beta(t)(D_{free}-d_j)^2+\delta\sum_{i=1,j=1}^{k}\frac{t_{ij}}{d_i}]} \tag{14}$$

The reward function indicates that the lower the latency and bandwidth usage, and the restriction conditions (11.a-11.f) are met, the higher the feedback reward value. However, if the system cannot meet the restriction conditions (11.a-11.f) after allocation, $r(t) = 0$.

The Markov Decision Process is shown in Fig. 5.

B. EDGE STATUS PERCEPTION NETWORK

In part A, the problem of joint minimizing latency and bandwidth usage is converted to MDP. Then, the DDPG-Based method is used to quickly solve the MDP. The DDPG-Based method needs to perceive the status of the edge layer and select the optimal action for this status. In the status space of MDP, a status matrix with the dimension of $4 \times k \times k$

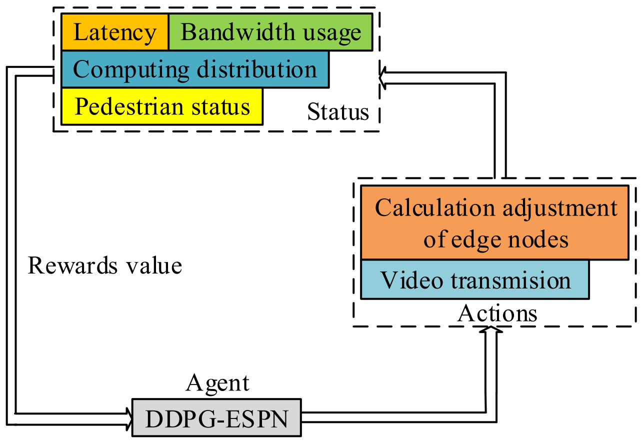


FIGURE 5. **Markov decision process of the proposed system. The agent obtains the status and rewards value from the edge environment, and executes actions based on the status. Then, the status transitions to the next state according to the actions performed, and return the rewards value of the action. The rewards value is calculated by (14).**

is used to describe the edge layer environment. Meanwhile, we define action matrix as the outputs of actor network. Therefore, the $4 \times k \times k$ dimensional state matrix can be regarded as the input of the actor network. In addition, the $5 \times k \times k$ dimensional state-action matrix can be seen as the input of the critic network.

Fully convolutional network is used to constitute actor network and critic network in the traditional DDPG method. However, fully convolutional network needs to store a large number of parameters when processing large-dimensional input and output tasks, which is not conducive to deployment at the edge. At the same time, fully convolutional network is less able to perceive the features of matrix data compared to convolutional neural networks.

In order to overcome the above drawbacks, ESPN (Edge Status Perception Network) is proposed to extract the status feature of the edge layer. ESPN is divided into ESPN-actor and ESPN-critic. ESPN-actor is composed of fully convolutional network and deconvolutional neural network. Fully convolutional network is used to extract the status features of edge layer and input the feature information to the deconvolutional neural network. While the deconvolutional neural network is used to output actions. ESPN-critic is used to evaluate the actions generated by ESPN-actor. The first 10 layers of ESPN-critic have the same structure as ESPN-actor, but the second half of ESPN-critic also uses fully convolutional network for feature analysis to obtain a value estimate. In order to improve ESPN's ability to perceive high-dimensional features, we use a residual structure in the convolution layer of ESPN. The structure of ESPN is shown in Fig. 6.

The output layer of ESPN-actor uses the tanh activation function, while the output layer of ESPN-critic uses the sigmoid activation function to fit the reward value. In order to enable ESPN to extract the input features exhaustively, we use the LeakyRelu ($\alpha = 0.1$) function as the activation function of the hidden layers.

C. DDPG-ESPN ALLOCATION METHOD

To the problem (11), we convert it to MDP, and use ESPN for edge layer status perception and action output. Compared with the DQN-Based methods, DDPG-Based methods can output continuous actions. Therefore, the DDPG-Based method can reduce the output dimension of the action output network and improve the efficiency of algorithm operation.

The training process of the DDPG-ESPN model is divided into ESPN-actor and ESPN-critic. The actor network is used to output actions, while the critic network is responsible for evaluating actions based on the feedback environment. We define $\pi(s|\theta^{\pi})$ and $Q(s, a|\omega^{Q})$ represent ESPN-actor and ESPN-critic network respectively. θ^{π} is the parameters of actor net, while ω^{Q} denotes the parameters of critic network. $\pi'(s|\theta^{\pi'})$ and $Q'(s, a|\omega^{Q'})$ denote the ESPN-actor target network and ESPN-critic target network respectively. The structures of these two target networks are the same as those of the ESPN-actor and ESPN-critic networks. The framework of DDPG-ESPN is shown in Fig. 7.

When the DDPG-ESPN model starts to train, it is necessary to build ESPN-actor network and ESPN-critic network separately, and initialize its parameters randomly. Then clone these two networks and parameters as the ESPN-actor target network and ESPN-actor target network. Finally, the agent uses the ESPN-actor to generate actions based on the edge environment and generate rewards $r(t)$. Meanwhile, the edge environment enters the next states $s(t+1)$, $(s(t), a(t), r(t), s(t+1))$ is stored in the experience replay buffer. After a certain number of samples are stored in the experience replay buffer, the agent samples mini-batch samples from the experience replay buffer.

When the samples are sampled, the ESPN-actor target network first predicts the action $a'(t+1)$ based on $s(t+1)$. Then $Q'(s(t+1), a'(t+1)|\omega^{Q'})$ is fitted by the ESPN-critic target network, and the target Q value is calculated:

$$y(t) = r(t) + \gamma Q'(s(t+1), a'(t+1)|\omega^{Q'}) \tag{15}$$

where, γ is the discount rate.

In each mini-batch, the critic network is updated according to the target Q value. The loss function is:

$$loss(\omega^{Q}) = \frac{1}{N_{mini}} \sum_{t=1}^{N_{mini}} (y(t) - Q(s(t), a(t)|\omega^{Q})) \tag{16}$$

where, N_{mini} denotes the number of mini-catch samples.

Update the parameters of the ESPN-actor network through the policy gradient:

$$\nabla_{\theta^{\pi}} \approx \frac{1}{N_{mini}} \sum_{t=1}^{N_{mini}} \nabla_{a} Q(s(t), a(t)|\omega^{Q})|_{s=s(t), a=\pi(s(t))} \times \nabla_{\theta^{\pi}} \pi(s(t)|\theta^{\pi}) \tag{17}$$

In order to ensure the stability of training, the parameters of target network need to be soft updated:

$$\theta^{\pi'} = \delta\theta^{\pi} + (1-\delta)\theta^{\pi'} \tag{18}$$

$$\omega^{Q'} = \delta\omega^{Q} + (1-\delta)\omega^{Q'} \tag{19}$$

where, δ is the update rate.

The training algorithm is shown in Algorithm 1.

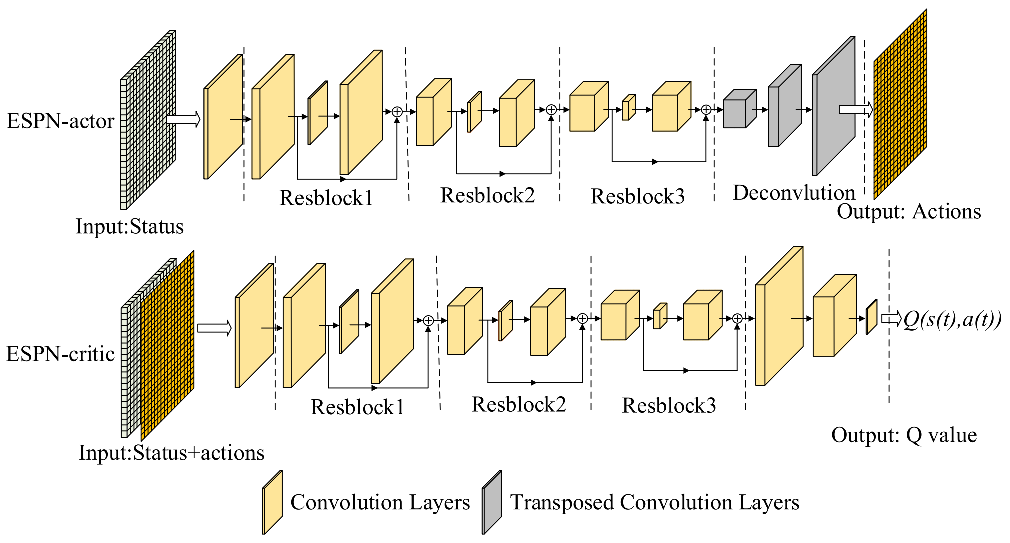


FIGURE 6. The structure of ESPN-actor and ESPN-critic.

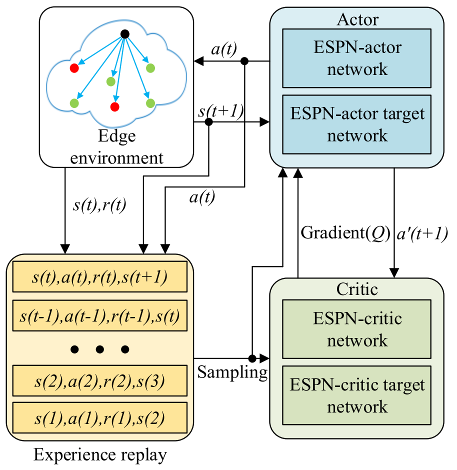


FIGURE 7. The framework of DDPG-ESPN. The entire DDPG-ESPN needs training four networks, including ESPN-actor network, ESPN-actor target network, ESPN-critic network, and ESPN-critic target network.

Algorithm 1 Training Process of DDPG-ESPN

```
1.  Initialize the ESPN-actor and ESPN-critic network $\pi(s|\theta^{\pi})$ and $Q(s, a|\omega^{Q})$ respectively. Randomly initialize the weights $\theta^{\pi}$ and $\omega^{Q}$ respectively.
2.  Initialize target net $\pi'(s|\theta^{\pi'})Q'(s, a|\omega^{Q'})$ and by clone ESPN-actor and ESPN-critic respectively.
3.  Initialize replay buffer R
4.  for $episode = 1$ to max\_$episode$ do
5.      Reset environment randomly;
6.      for $t = 1$ to max_time_slot do
7.          Obtain $s(t)$ from edge environment;
8.          Get action $a(t)$ from actor network and carry out it in manage node;
9.          Obtain reward $r(t)$ and state $s(t+1)$ from environment;
10.         Store $(s(t), a(t), r(t), s(t+1))$ to R
11.         if R is full then do
12.             Get action $a(t+1)$ by target action network;
13.             Get Q' value from $s(t+1)$ and $a(t+1)$ by target critic network
14.             Get y by (15)
15.         Update $\omega^{Q}$ by (16) with Adam [37] optimizer;
16.         Update $\theta^{\pi}$ by (17) with Adam optimizer;
17.         Update the weights of target networks in the critic and actor networks by (18)(19).
```

D. THE TIME COMPLEXITY OF ESPN

DDPG-ESPN is proposed to allocation the edge computing resource. The whole method is deployed on the Cloud servers for training. When it deployed on the management nodes at edge environment, only the ESPN-actor is used to perceive the edge status and make decisions. Therefore, the time complexity of DDPG-ESPN for edge resources allocation is mainly influenced by the time complexity of ESPN-actor. Hence, we only analyze the time complexity of ESPN-actor for simplicity. The time complexity of ESPN-actor is analyzed by calculating FLOPs (Floating Point Operations) during inference.

The time complexity of convolutional is shown in (20):

$$FLOP_{conv} = \frac{M^2K^2NC_{in}}{S} \tag{20}$$

where, M is the size of input feature map; K denotes the size of filter; C_{in} is the number of input channels; N is the number of filters.

The time complexity of deconvolution is shown in (21):

$$FLOP_{deconv} = M^2K^2NC_{in} \tag{21}$$

By (19), the time complexity of the first layer of the ESPN-actor can be calculated as follows:

$$FLOP_{conv1} = 4M^2K^2N \tag{22}$$

where, C_{out1} represents the output channels of the first convolutional layer.

The time complexity of the Resblock is shown in (23):

$$FLOP_{\mathrm{Res}} = 3.5M^2N_{resconv1} \tag{23}$$

where, $N_{resconv1}$ is the number of the first convolutional layer of Resblock.

From (20) to (21), the time complexity of ESPN-actor can be calculated as follows:

$$FLOP_{ESPN-actor} = 92M^2N^2 + 44.875M^2N \qquad (24)$$

By (24), the time complexity of ESPN-actor (the details is shown in TABLE 3) is 0.392GFLOPs for an input size of 4 × 64 × 64. The time complexity of ESPN-actor is much smaller than that of YOLOv3-tiny [36] object detect model. Hence, the time complexity of ESPN is not the main factor that causing the computing latency of edge nodes.

VI. PERFORMANCE EVALUATION

A. EXPERIMENTAL SETUP

1) PARAMETERS SETTING OF EDGE COMPUTING ENVIRONMENT

In an edge environment, we set up 50 edge computing nodes and divide them into three levels according to their computing capabilities, as shown in Table 2. We choose a node with the highest computing power level as the management node. We set the Industrial Ethernet bandwidth connected to the edge node to 1000Mbps, and the video resolution to 720p. The DNN object detection model YOLOv3-tiny needs 38.97 GFLOPS for each frame of picture processed [36]. We set $D_{busy} = 40ms$, $D_{idle} = 200ms$, and select 1-20 pedestrians at the beginning of each training iteration, and randomly distribute these pedestrians to 50 monitoring areas. Each time slot will randomly increase or decrease a pedestrian, but the number of pedestrians is always maintained at 1-20. We assume that only video is transmitted in the edge network, and only object detection and resource allocation algorithms are run in edge nodes. The latency caused by memory access is ignored, and personnel detection is regarded as the task goal.

TABLE 2. **The computing capacity level of edge nodes.**

Level	Computing capacity	Numbers
High	1.2TFLOPS	10
Normal	1.0TFLOPS	20
Low	0.472TFLOPS	20

2) PARAMETERS SETTING OF ESPN

The input shape of ESPN-actor is set to 4 × 64 × 64, While the input size of ESPN-critic is 5 × 64 × 64. The details of ESPN are shown in Table 3.

When deploying DDPG-ESPN at the edge, and only the ESPN-actor network needs to be deployed at the edge node for resource allocation. The computational complexity of the ESPN-actor network is the main part of the computing cost of edge node resource allocation. By analysis, the computational complexity of ESPN-actor is 0.392 GFLOPS, which is much smaller than that of the target detection model. Meanwhile, considering that the edge management node performs resource allocation tasks when pedestrians cross the monitoring areas. Therefore, the computing latency of ESPN-actor is ignored for simplicity.

3) PARAMETERS SETTING OF DDPG

The parameters setting of DDPG is shown in Table 4.

The hardware and software environments used in this paper is shown in TABLE 5.

B. PARAMETRIC ANALYSIS

The convergence of the DDPG-ESPN is highly dependent on the training parameters. Reasonable parameters can not only make the model converge faster, but also get higher rewards.

During the training process of the DDPG-ESPN, its convergence is easily affected by the learning rate of the ESPN-actor network. In this paper, the learning rate of ESPN-critic network is fixed, and we investigated the convergence performance of DDPG-ESPN by changing the learning rate of the ESPN-actor network. As shown in Fig. 8, if the learning rate is too large, the convergence process of DDPG-ESPN will become unstable and the rewards will be lower. Meanwhile, if the learning rate is too small, the training process of DDPG-ESPN will fall into local optimization, which will slow down its convergence speed and make it difficult to increase rewards. When we set lr_a = 0.0001, the convergence performance of DDPG-ESPN is outstanding, and the model obtains a higher average rewards.

The long-term rewards and short-term rewards of DDPG-ESPN are affected by the discount factor. The larger the discount factor, the more DDPG-ESPN focus on long-term rewards, while the smaller the discount factor, the more DDPG-ESPN focus on short-term rewards.

As shown in Fig. 9, the large discount factor allows the model to focus on long-term rewards, resulting in low average real-time in "busy" areas and low bandwidth usage for video transmissions, while the excessive real-time performance of the "idle" region causes a waste of computing resources.

When the discount factor is set to 0.99, the model is more sensitive to short-term rewards, resulting in high average real-time performance in "busy" areas, and low real-time performance in "idle" areas. Meanwhile, the bandwidth usage by video transmission is higher, which is likely to cause interference to the transmission of other information.

When the discount factor is set to 0.995, the system allocates computing resources to the edge layer more reasonably. This discount factor avoids computing resources wasted and poor real-time performance, which is caused by the excessive or low allocation of computing power in the monitoring area.

C. SCALABILITY TEST

In order to verify the scalability of DDPG-ESPN, we set a different number of edge devices based on the optimal parameters in the previous part. The input size of ESPN-actor is 4 × 64 × 64. Therefore, the maximum number of nodes that can be managed by DDPG-ESPN is 64. The relationship between the average reward obtained after ESPN convergence and the number of edge nodes is shown in Fig. 10.

TABLE 3. Details of ESPN.

		ESPN-actor	ESPN-critic
layers		Structure	Structure
Input		$4\times64\times64$	$5\times64\times64$
Conv1		filters=32,kernel_size=(3,3),strides=(1,1)	
Conv2		filters=64,kernel_size=(3,3),strides=(2,2)	
Conv3	Resblock1	filters=32,kernel_size=(1,1),strides=(1,1)	
Conv4		filters=64,kernel_size=(3,3),strides=(1,1)	
Conv5		filters=128,kernel_size=(3,3),strides=(2,2)	
Conv6	Resblock2	filters=64,kernel_size=(1,1),strides=(1,1)	
Conv7		filters=128,kernel_size=(3,3),strides=(1,1)	
Conv8		filters=256,kernel_size=(3,3),strides=(2,2)	
Conv9	Resblock3	filters=128,kernel_size=(1,1),strides=(1,1)	
Conv10		filters=256,kernel_size=(3,3),strides=(1,1)	
Deconv1\Conv11		filters=512,kernel_size=(3,3),strides=(2,2)	filters=512,kernel_size=(3,3),strides=(2,2)
Deconv2\Conv12		filters=1024,kernel_size=(2,2),strides=(2,2)	filters=256,kernel_size=(2,2),strides=(2,2)
Deconv3\Conv13		filters=1,kernel_size=(2,2)	filters=1,kernel_size=(2,2),strides=(2,2)

TABLE 4. The parameters setting of DDPG-ESPN.

Parameters	value
ESPN-actor learning rate	1e-4
ESPN-critic learning rate	1e-3
Discount factor	0.995
Target update rate	5e-3
Experience replay buffer size	200000
Minibatch size	128
Max_time_slop	400
Training episodes	3000

TABLE 5. The hardware and software environments.

Training platform	Deployment platform
Intel-i7 9700K	NVIDIA Jetson Nano
NVIDIA GTX 1080Ti	NVIDIA Jetson TX1
Windows 10	NVIDIA Jetson TX2
Python 3.6.8	Ubuntu 18.04
Tensorflow 1.14.0	Tensorflow 1.14.0
	Python 3.6.9

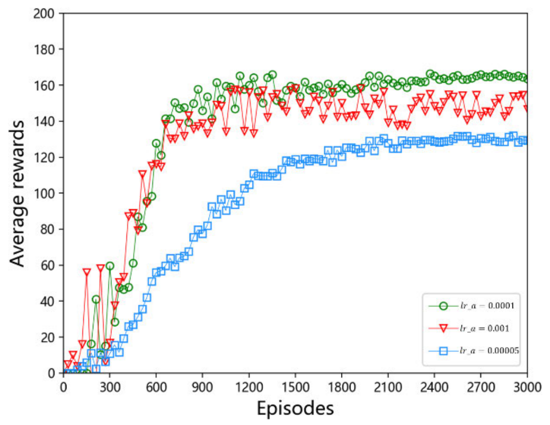


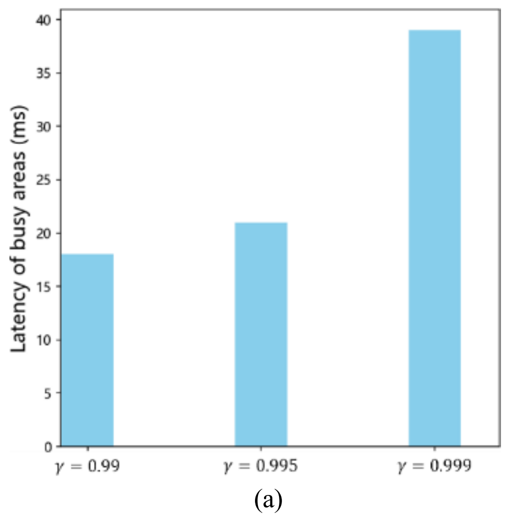
FIGURE 8. The training process of DDPG-ESPN under different learning rates of ESPN-actor. The lr_a denotes the learning rate of ESPN-actor.

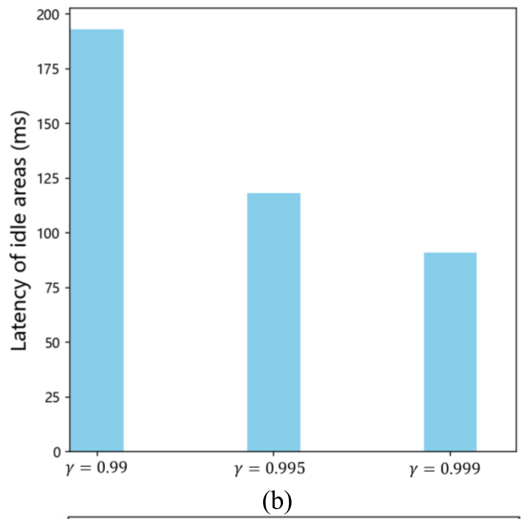
As shown in Fig. 9, the fewer the number of nodes, the fewer resources the system can schedule. Therefore, the rewards for DDPG-ESPN after convergence are less with the decrease of the number of edge nodes. However, the edge environment will be complex when the number of edge nodes is too large. At this time, the ability of edge environment perception and action output are insufficient, which leads to a reduction in rewards. Therefore, the optimal number of edge nodes is between 40-60 based on Fig. 10.

D. PERFORMANCE COMPARISON

In order to verify the performance advantage of the proposed method for edge computing resource allocation, we compare DDPG-ESPN with the A3C-Based method. A3C (Asynchronous advantage actor-critic) is a Policy-based method, which can output continuous actions like DDPG. We combine A3C [33], [34] and ESPN to form the A3C-ESPN algorithm. Meanwhile, an FCN is designed to construct DDPG-FCN and A3C-FCN. We compare the performance of the four methods by observing the latency and bandwidth usage of the edge layer in different numbers of ‘‘busy’’ areas.

The more number of ‘‘busy’’ areas, the more edge computing power the system needs to allocate to the ‘‘busy’’ areas for maintaining high real-time performance. Meanwhile, the real-time performance of ‘‘idle’’ areas is also maintained. As shown in Fig. 11, with the increasing number of ‘‘busy’’ areas, the latency has risen to vary degrees. The environment feature perception ability of the agent is enhanced by ESPN. Therefore, A3C-ESPN and DDPG-ESPN are able to extract deep features of the edge layer for selecting a better resource allocation policy. Compared with ESPN, the feature perception ability of FCN is weaker, and it is difficult for FCN to deal with the high-dimensional output. It results in poor resource allocation effect of A3C-FCN and DDPG-FCN. However, as the number of ‘‘busy’’ areas rises, the computing resources of edge layer are gradually depleted. At this time, the impact of resource allocation methods on computing latency is no longer significant, and the real-time performance can only be improved by accessing more computing resources.





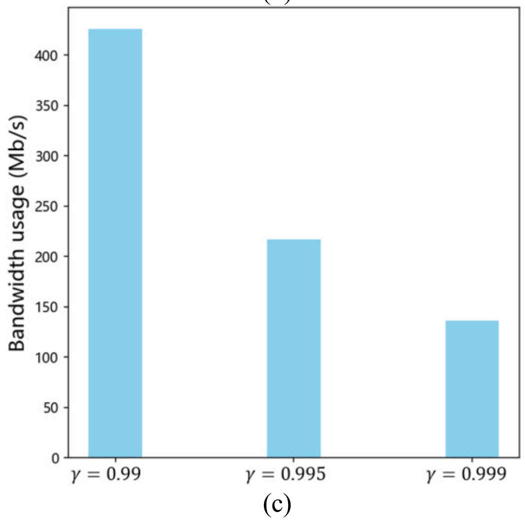


FIGURE 9. The performance of DDPG-ESPN influenced by different discount factor. (a) The latency of "busy" areas (b) The latency of "idle" areas (c) The bandwidth usage.

To intuitively show the advantages of the proposed method, the performance of different methods is represented by numerical values. Considering the performance of each method varies with the number of "busy" areas, the overall performance of the method cannot be expressed as a certain point value in the curve. Therefore, we used average performance values to represent the performances of those methods.

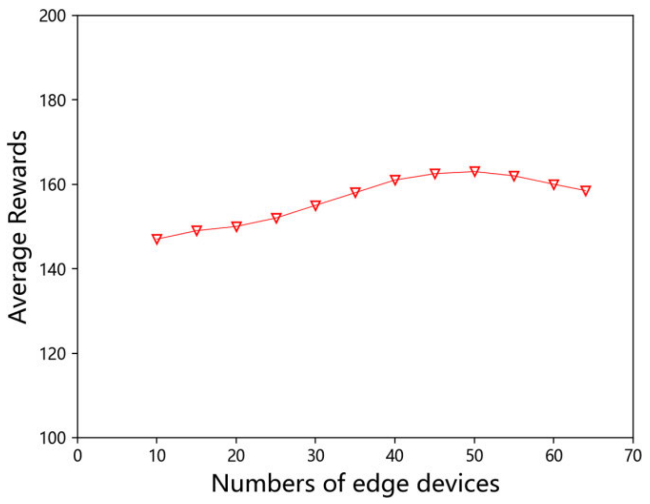


FIGURE 10. Curve of the average convergence rewards and the number of edge nodes.

TABLE 6. The average performance value.

Performance	Method	value
Average latency of busy area	DDPG-ESPN	25.15
	A3C-ESPN	27.67
	DDPG-FCN	29.49
	A3C-FCN	31.01
Average latency of idle area	DDPG-ESPN	117.69
	A3C-ESPN	131.87
	DDPG-FCN	143.03
	A3C-FCN	151.13
Bandwidth usage	DDPG-ESPN	242.60
	A3C-ESPN	304.30
	DDPG-FCN	364.63
	A3C-FCN	424.20

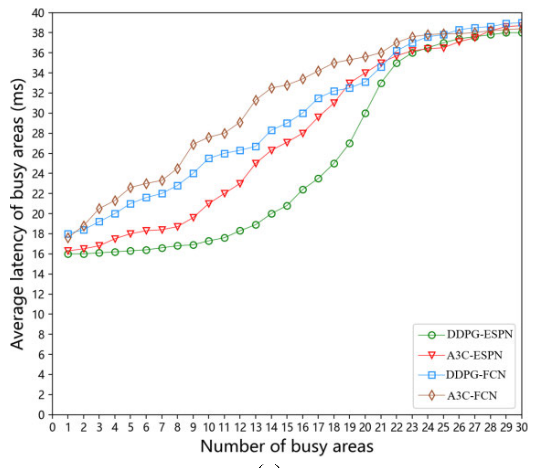
The average performance values can be calculated by (25):

$$P_{\mathrm{a}} = \frac{\sum_{s=1}^{N_n} P_s}{N_n} \tag{25}$$

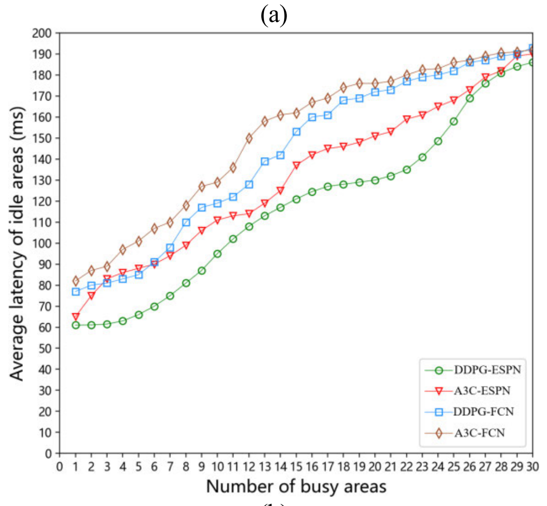
where, the number of nodes is represented by s. N_n denotes the maximum number of nodes. P_s represents the performance index when the number of busy nodes is s. In this paper, N_n is set to 30.

By (25), the average performance values of Average latency of busy areas (Fig. 11 (a)), Average latency of idle areas (Fig. 11(b)) and Bandwidth usage (Fig. 11(c)) are summarized in TABLE 6.

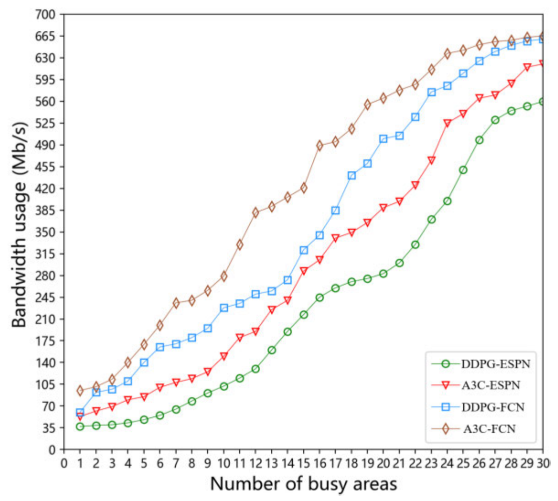
As shown in TABLE 6, the performance of DDPG-ESPN has a great improvement compared with other methods. Compared with A3C-ESPN, the DDPG-ESPN's Average latency of busy area, Average latency of idle area and Bandwidth usage are 9.11%, 10.75%, 20.28% lower respectively.



(a)



(b)



(c)

FIGURE 11. The performance comparison of DDPG-ESPN, DDPG-FCN, A3C-ESPN, and A3C-FCN. (a) The comparison of average latency of "busy" areas (b) The comparison of average latency of "idle" areas (c) The comparison of bandwidth usage.

This result shows that the DDPG-based method is more suitable for resource allocation in the edge environment proposed in this paper. Compared with DDPG-FCN, the DDPG-ESPN's Average latency of busy area, Average latency of idle area and Bandwidth usage are 14.72%, 17.72%, 33.47% lower respectively. The results show that ESPN has stronger edge environment perception ability than FCN. For A3C-FCN, the average performance value is the highest of the four methods. Compared with DDPG-ESPN, the A3C-FCN's Average latency of busy area, Average latency of idle area and Bandwidth usage are 18.88%, 22.13%, 42.81% higher respectively.

VII. CONCLUSION

In this paper, we propose DDPG-ESPN to allocate edge resources of coal mine intelligent video surveillance system in a cloud-edge cooperation environment. First, we designed a cloud-edge cooperation framework in coal mine. Edge computing is used for real-time intelligent surveillance video analysis, and cloud computing provides model training and data storage services for edge computing. Then, we analyzed the latency problem caused by the intelligent inference at the edge, and modeled the problem of joint minimizing latency and bandwidth usage. Finally, the joint minimizing problem is converted to MDP. To quickly solve the problem, the DDPG-ESPN is proposed. In the future, we will research the problems of latency and bandwidth usage caused by the increased number of "busy" areas.

Although the edge resource can be allocated by DDPG-ESPN, the processing latency still increase quickly with the increase number of "busy" area. The reason for this may be the insufficient computing resources of the edge. Meanwhile, the resources of cloud server are not fully utilized. In the future, we will optimize the cloud-edge computing environment to make full use of cloud computing resources. In addition, the resources of edge computing are still constrained, therefore how to offload tasks to cloud servers and reduce the latency as much as possible is the next research goal.

REFERENCES

[1] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016, doi: 10.1109/MC.2016.145.

[2] X. Wang, L. T. Wang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 80–85, Nov. 2017, doi: 10.1109/MCOM.2017.1700360.

[3] Y. Wang, K. Hong, J. Zou, T. Peng, and H. Yang, "A CNN-based visual sorting system with cloud-edge computing for flexible manufacturing systems," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4726–4735, Jul. 2020, doi: 10.1109/TII.2019.2947539.

[4] S. Ahn, J. Lee, T. Y. Kim, and J. K. Choi, "A novel edge-cloud interworking framework in the video analytics of the Internet of Things," *IEEE Commun. Lett.*, vol. 24, no. 1, pp. 178–182, Jan. 2020, doi: 10.1109/LCOMM.2019.2943857.

[5] Y. Wang, M. Liu, P. Zheng, H. Yang, and J. Zou, "A smart surface inspection system using faster R-CNN in cloud-edge computing environment," *Adv. Eng. Informat.*, vol. 43, Jan. 2020, Art. no. 101037, doi: 10.1016/j.aei.2020.101037.

[6] R. Rajavel, S. K. Ravichandran, K. Harimoorthy, P. Nagappan, and K. R. Gobichettipalayam, "IoT-based smart healthcare video surveillance system using edge computing," *J. Ambient Intell. Hum. Comput.*, pp. 1–13, Mar. 2021, doi: 10.1007/s12652-021-03157-1.

[7] R. Jayaram and S. Prabakaran, "Onboard disease prediction and rehabilitation monitoring on secure edge-cloud integrated privacy preserving healthcare system," *Egyptian Informat. J.*, Dec. 2020, doi: 10.1016/j.eij.2020.12.003.

[8] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019, doi: 10.1109/TVT.2019.2904244.

[9] P. Yang, F. Lyu, W. Wu, N. Zhang, L. Yu, and X. S. Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4855–4864, Jul. 2020, doi: 10.1109/TII.2019.2949347.

[10] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, "VideoEdge: Processing camera streams using hierarchical clusters," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Seattle, WA, USA, Oct. 2018, pp. 115–131, doi: 10.1109/SEC.2018.00016.

[11] Y. Cao, G. Zhang, G. Li, and J. Zhang, "A deep *Q*-network based-resource allocation scheme for massive MIMO-NOMA," *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1544–1548, May 2021, doi: 10.1109/LCOMM.2021.3055348.

[12] Y.-C. Wu, T. Q. Dinh, Y. Fu, C. Lin, and T. Q. S. Quek, "A hybrid DQN and optimization approach for strategy and resource allocation in MEC networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4282–4295, Jul. 2021, doi: 10.1109/TWC.2021.3057882.

[13] G. Sun, D. Ayepah-Mensah, A. Budkevich, G. Liu, and W. Jiang, "Autonomous cell activation for energy saving in cloud-RANs based on dueling deep *Q*-network," *Knowl.-Based Syst.*, vol. 192, pp. 1–11, Mar. 2020, doi: 10.1016/j.knosys.2019.105347.

[14] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019, doi: 10.1109/TVT.2019.2935450.

[15] N. Jiang, Y. Deng, A. Nallanathan, and J. A. Chambers, "Reinforcement learning for real-time optimization in NB-IoT networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1424–1440, Jun. 2019, doi: 10.1109/JSAC.2019.2904366.

[16] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, and J. Hu, "IRAF: A deep reinforcement learning approach for collaborative mobile edge computing IoT networks," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7011–7024, Aug. 2019, doi: 10.1109/JIOT.2019.2913162.

[17] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020, doi: 10.1109/JSAC.2020.2986615.

[18] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in IoT networks via machine learning," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3415–3426, Apr. 2020, doi: 10.1109/JIOT.2020.2970110.

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[20] S. Han, H. Mao, and W. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, San Juan, PR, USA, 2017, pp. 1–14.

[21] Z. Wang, J. Zhang, Z. Zhao, and F. Su, "Efficient yolo: A lightweight model for embedded deep learning object detection," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, London, U.K., Jul. 2020, pp. 1–6, doi: 10.1109/ICMEW46912.2020.9105997.

[22] H. Lu, X. He, M. Du, X. Ruan, Y. Sun, and K. Wang, "Edge QoE: Computation offloading with deep reinforcement learning for Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9255–9265, Oct. 2020, doi: 10.1109/JIOT.2020.2981557.

[23] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[24] J. Ren, Y. Guo, D. Zhang, Q. Liu, and Y. Zhang, "Distributed and efficient object detection in edge computing: Challenges and solutions," *IEEE Netw.*, vol. 32, no. 6, pp. 137–143, Nov./Dec. 2018, doi: 10.1109/MNET.2018.1700415.

[25] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, and Y. Yang, "Asymptotic soft filter pruning for deep convolutional neural networks," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3594–3604, Aug. 2020, doi: 10.1109/TCYB.2019.2933477.

[26] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, 2016, pp. 1–13.

[27] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 5068–5076, doi: 10.1109/ICCV.2017.541.

[28] J. Chen, H. Xing, Z. Xiao, L. Xu, and T. Tao, "A DRL agent for jointly optimizing computation offloading and resource allocation in MEC," *IEEE Internet Things J.*, early access, May 19, 2021, doi: 10.1109/JIOT.2021.3081694.

[29] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-guaranteed collaborative DNN inference in industrial IoT via deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4988–4998, Jul. 2021, doi: 10.1109/TII.2020.3017573.

[30] H. Peng and X. S. Shen, "DDPG-based resource management for MEC/UAV-assisted vehicular networks," in *Proc. IEEE 92nd Veh. Technol. Conf. (VTC-Fall)*, Antwerp, Belgium, Nov. 2020, pp. 1–6, doi: 10.1109/VTC2020-Fall49728.2020.9348633.

[31] T. Zhang, K. Zhu, and J. Wang, "Energy-efficient mode selection and resource allocation for D2D-enabled heterogeneous networks: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1175–1187, Feb. 2021, doi: 10.1109/TWC.2020.3031436.

[32] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8577–8588, Oct. 2019, doi: 10.1109/JIOT.2019.2921159.

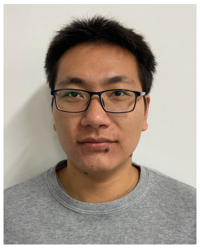
[33] W. Zhang, D. Yang, P. Haixia, W. Wu, W. Quan, H. Zhang, and X. Shen, "Deep reinforcement learning based resource management for DNN inference in industrial IoT," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7605–7618, Aug. 2021, doi: 10.1109/TVT.2021.3068255.

[34] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks," *IEEE Trans. Mobile Comput.*, early access, Aug. 17, 2020, doi: 10.1109/TMC.2020.3017079.

[35] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 6517–6525, doi: 10.1109/CVPR.2017.690.

[36] J. Zou, T. Hao, C. Yu, and H. Jin, "A3C-DO: A regional resource scheduling framework based on deep reinforcement learning in edge scenario," *IEEE Trans. Comput.*, vol. 70, no. 2, pp. 228–239, Feb. 2021, doi: 10.1109/TC.2020.2987567.

[37] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.



ZHI XU received the bachelor's degree in electrical engineering from the College of Electrical and Information Engineering, Anhui University of Science and Technology (AUST), Huainan, China, in 2018, where he is currently pursuing the Ph.D. degree in mechanical engineering. His current research interests include edge computing and artificial intelligence.

JINGZHAO LI received the M.A. degree from the China University of Mining and Technology, in 1992, and the Ph.D. degree from the Key Laboratory of Power Electronics and Power Drives, Hefei University of Science and Technology, in 2003. He is currently a Professor with the School of Electrical Information and Engineering, Anhui University of Science and Technology, China. He has published more than 100 papers in domestic and international academic journals and conference proceedings. These papers are embodied more than 60 times by SCI and EI and cited more than 100 times by others. His research interests include computer control, the Internet of Things technology, and embedded systems.