This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

A Surveillance Video Real-time Analysis System Based on Edge-cloud and FL-YOLO Cooperation in Coal Mine

Zhi XU, Jingzhao LI, Mei ZHANG

College of Electrical and Information Engineering, Anhui University of Science and Technology, Huainan, 232000 China

Corresponding author: Jingzhao LI (e-mail:ljzaust@outlook.com).

This work was supported by the National Natural Science Foundation of China (Grant No.51874010), the Key Technology Research & Innovation Team Project (Grant No. 201950ZX003), the Natural Science Research Projects of Colleges and Universities in Anhui Province （KJ2020A0309）, and Huaibei Mining Group National Technology Center Project: "Coal Mine Control System Based on Video Monitoring".

ABSTRACT Video monitoring is an important means to ensure production safety in coal mine. However, the currently intelligent video surveillance is difficult to respond in real-time due to the latency of cloud computing. In this paper, a cloud-edge cooperation framework is proposed, which integrates cloud computing and edge computing in a coordinated manner. The cloud computing is used to process non-real-time and global tasks, while the edge computing is responsible for handling local monitoring video in real-time. In order to realize cloud-edge data interaction and online optimization for edge models, the heterogeneous converged network is built. In addition, an object detection model FL-YOLO composed of depthwise separable convolution and down-sampling inverted residual block is proposed, which realizes real-time video analysis at the edge. Finally, this paper discusses the complexity of FL-YOLO by its computational cost and model size. The experiment results show that the model size of FL-YOLO is 16.1MB, which is very light, and it achieves 36.7 FPS on NVIDIA Jetson TX1 and an AP of 76.7% on Multi-scene pedestrian dataset. Comparing with mainstream object detection models, FL-YOLO completes faster detection speed and higher accuracy, and it has lower calculation complexity and smaller model scale. Furthermore, the AP on Single-scene pedestrian dataset of FL-YOLO is improved to 80.7% by cloud-edge cooperation. K-Fold method is also used to further compared the performance of FL-YOLO and other models. Moreover, system test is implemented on coal mine, which validates the actual engineering effect of the proposed cloud-edge cooperation framework.

INDEX TERMS Edge computing, YOLO, cloud-edge cooperation, real-time analysis.

I. INTRODUCTION

Coal mine video surveillance plays an important role in ensuring coal mine production safety and the life of workers, but many coal mining manufacturer still use manpower to process surveillance videos. However, manpower handling inevitably produces a series of problems such as inefficient, untimely response, and human physiological fatigue [1]. In recent years, with the development of AI technology, intelligent video surveillance in coal mine is undoubtedly a major trend in the future. Compared with manual video surveillance, intelligent video surveillance can not only process faster and better, but also greatly reduce the costs of coal mine companies. However, intelligent video surveillance requires a large amount of storage and computing resources. As a result, AI models for video processing are usually deployed on cloud servers with rich computing and storage resources. Unfortunately, cloud computing will produce various problems, such as high latency, network congestion, etc. These problems seriously affect the safety of coal mine production. To solve the above problems, traditional intelligent video surveillance framework and AI algorithm must be improved.

Recently, Object detection algorithm based on Convolutional Neural Networks (CNNs) is used in various video surveillance fields [2][3][4]. CNN is used to extract the features of the input image and eventually detect the objects in the image. Its performance has reached or even beyond the human level. However, traditional CNN-based object detection algorithms require large scale of parameters and computations, and it can only be deployed on cloud servers. Therefore, Cloud-based intelligent video surveillance in coal

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

mine will be limited in many aspects. In terms of network, firstly, the coverage of Industrial Ethernet is limited. Secondly, wireless signal is restricted by the narrow tunnels and the interference from high-current equipment. Thirdly, video transmission requires a large amount of network bandwidth, and the process also generates serious latency. Hence, the application scenario of Cloud-based intelligent video surveillance is constrained by network conditions of coal mine. In terms of video processing, the large amount of monitoring video will put tremendous computing pressure on the cloud servers, it will cause computing latency. To sum up the above, traditional CNN-based object detection algorithms deployed on cloud servers have serious latency, and the latency will extremely reduce the performance of cloud-based intelligent video surveillance.

Edge computing is proposed to solve the problems of high latency and network congestion in cloud computing [5]. As the data centrally processed on cloud servers is dispersed to the edge, computing and network pressure on cloud servers are greatly reduced. Meanwhile, Edge computing also improves the real-time performance of whole system. Currently, with the continuous development of electronic technology, the computing and storage capabilities of embedded devices are constantly improving. At the same time, due to the research and development of lightweight neural networks, embedded devices obtain the ability of intelligent computing. However, the real-time and accuracy performances of AI models are limited by the computing and storage resources of embedded platform. Furthermore, coal mine has numerous monitoring scenarios, and the lightweight model with poor generalization ability cannot adapt to multiple scenarios. Hence, the accuracy of the lightweight models deployed on the edge nodes cannot meet the actual needs of coal mine.

To overcome the abovementioned problems, we proposed an Edge-Cloud cooperation framework and FL-YOLO (Fast and Lightweight YOLO). In this framework, cloud computing is used to train and optimize edge models, and it also provide other services of video surveillance system. Edge computing is used to analyze surveillance video in real-time with FL-YOLO algorithm. Eventually, the system is able to detect objects with high speed and accuracy, so that coal mine equipment and warning can quickly respond according to the intelligent analysis results. It avoids workers hurt by equipment or other issues, and enhance the safety of coal mine production. The main contributions of this paper are as follows.

1) The framework of Edge-Cloud cooperation is proposed to realize real-time intelligent video surveillance in coal mine. The latency of this framework is much less than that of traditional methods, and it expands the coverage of video surveillance in coal mine.

2) Based on the depthwise separable convolution, a lightweight object detection model FL-YOLO is proposed to implement on embedded platform. The size of the model is only 16.1MB, and it has a great performance of speed and accuracy.

The remainder of this paper is organized as follows. Related work about object detection, Edge computing and Edge-Cloud cooperation framework are introduced in Section II. In Section III, a cloud-edge cooperation framework of coal mine is proposed including edge-cloud service and heterogeneous converged network. In Section IV, a CNN-based object detection model FL-YOLO is proposed. Meanwhile, the complexity of FL-YOLO is discussed. In Section V, experimental results and discussions are given, where the accuracy and speed of FL-YOLO are compared with other models. The performance of Edge-Cloud cooperation framework and traditional framework are also compared in this section. Finally, the conclusion is drawn in Section VI.

II. RELATED WORK

A. OBJECT DETECTION ALGORITHMS

CNN-based object detection algorithms have been receiving a lot of attention from researchers due to the superior performance. The "one-stage" object detection method, as represented by the YOLO series, is widely used in real-time target detection. YOLO [6] was first proposed by Redmon J, Divvala S, Girshick R and Farhadi A. YOLO treats object detection as a regression problem, which is faster but less accurate than "two-stage" methods such as Faster-RCNN. Two years later, the authors of YOLO improved YOLOv1 and proposed YOLOv2 [7], which replaces the fully connected layer of YOLOv1 with a fully convolutional layer, so that it has the ability to handle images of different sizes. YOLOv2 also improved object positioning accuracy though introduces anchor boxes, and improved the capable of small objects detection by multi-scale detection. Redmon J and Farhadi A improved YOLOv2 in 2018, they proposed Darknet53 framework, which is able to extract deeper features compared to Darknet19. Finally, the detection accuracy of YOLOv3 has been greatly improved compared with YOLOv2, while maintaining the detection speed. YOLOv3 framework is widely used in object detection because of its excellent performance [8]. Xie L, Ahmad T, Jin L, Liu Y [9] proposed MD-YOLO that is able to predict the tilt angle of license plates by improving the output dimension of YOLO. VL-YOLO [10] was proposed by improving the framework of YOLOv3, it is more suitable for the detection of small-sized object compared to YOLOv3. IN-YOLO [11] is used to monitor surface condition of outdoor high voltage insulation. The advanced architecture of YOLO is evidenced by wide range of applications [12]-[14], while the excellent real-time performance and low number of network parameters allow YOLO to be applied to edge environments. Meanwhile, except deep learning methods, other methods such as brain programing [15] are also possessing high performance on the field of object detection. However, those methods are difficult to be applied in coal mine environment.

B. EDGE COMPUTING

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

Edge computing is a method to fill the shortcomings of cloud computing. The papers of [5][16][17] explore the concept of edge computing and its future development. Those authors of above paper believe that the massive amounts of data generated by IOT and cloud computing will put a huge strain on cloud servers. They think that due to the dramatic increase in the number of terminal devices in the future, cloud computing will unable to meet the requirements of network and computational cost in the future. Fortunately, the development of embedded devices has enhanced the ability of edge computing, which assist cloud computing in data processing. Edge computing provides the advantages of low latency, low bandwidth requirements, and low cost. Ren J, Guo Y, Zhang D, Liu Q [18] provide real-time object detection services at edge based on edge computing. These papers [19] [20] studied the computing and storage capabilities of edge devices and explored the applications of edge computing. Edge computing makes computing closer to data source, which reduces the latency, power consumption, and cost. Thereby, Edge computing broadens the application fields and practical effects of AI.

Running AI algorithms at edge not only needs to improve the computing and storage capacities of edge devices, but also optimize the traditional neural networks [21] [22]. Lightweight AI models can be obtained by lightweight neural networks or model compression. For designing lightweight neural network models, M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen proposed depthwise separable convolution [23] which greatly reduces the number of parameters and calculations of ANN model compared to the standard convolution. Li G, Yang Y and Qu X [24] combined depthwise separable convolution with YOLO to monitor pedestrians in foggy. Liu J, Wang X [25] used mobilenet to improve the model, and identify tomato leaf diseases on mobile devices. Lightweight neural networks such as mobilenet enable edge devices to gain intelligent computing capability.

For the method of compressing neural network, Song Han, Huizi Mao, Wallian J. Dally [26] used model pruning, weights quantization and Huffman coding to compress the model, reducing the model size by 35x to 49x and speedup the process of inference while maintaining the accuracy. Y. He, X. Dong, G. Kang, Y. Fu, C. Yan and Y. Yang [27] prune the convolutional filter of the model by the ASFP method, to solve the information loss caused by typical pruning algorithms. Li, Hao & Kadav, Asim & Durdanovic, Igor & Samet, Hanan & Graf [28] proposed a compression method for CNN to reduce the cost of computation. Jian-Hao Luo1, Jianxin Wu1, and Weiyao Lin [29] proposed Thinet framework to realize compress and speedup of CNN models. This framework decrease FLOPs by 3.31 times on VGG16, and decrease the size of the model by 16.63 times. Z. Wang, J. Zhang, Z. Zhao and F. Su [30] proposed the Efficient-YOLO based on YOLOv3, they compress the model size by layer-level and channel-wise pruning. As a result, the Efficient YOLO could deployed on embedded platform of NVIDIA Jetson TX2 with excellent accuracy and speed. Shi, Rui, Tianxing Li, and Yasushi Yamaguchi [31] prune the convolution kernels in channel-dimension to reduce the model size of YOLOv3-tiny to 5.3MB, and the pruning method decrease the computational cost of the model to 2.6 GFLOPs. So that the pruned YOLOv3-tiny model could deployed on ARM Cortex-A8 platform with accuracy of 94.4%. The above method is used to compress and speedup neural network models, but those methods need specialized computing strategy.

C. CLOUD-EDGE COOPERATION

Convergence of Cloud computing and Edge computing, edge computing provides users with low-latency, low-power services, while cloud computing is used to optimize the inference capability of edge computing. The cloud-edge cooperation method has already been applied in various field. Wang X, Yang L T, Xie X, Jin J and Deen M J [32] proposed a cloud-edge computing framework named CPSS (Cyber-Physical-Social Services). In the paper, cloud computing is used to process large-scale, long-term and global data, while edge computing is used to process small-scale, short-term and locality data. CPSS enables users to receive a higher quality and real-time service. Wang Y, Hong K, Zou J, Peng T and Yang H [33] proposed a cloud-edge computing environment to provide real-time picking services for factory-produced parts. Wang Y, Liu M, Zheng P, Yang H and Zou J [34] deployed R-CNN on edge devices and incorporate cloud computing for real-time monitoring of part surface defects. Ye L [35] used embedded devices to preliminary process the collected data, and then further analyzes the data through cloud computing to monitor the health of city pipes. Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik [36] proposed VideoEdge architecture, it identify the best tradeoff between resources and accuracy on cloud-edge collaboration framework for processing video stream. Meanwhile, they narrow the search space by identifying a "Pareto band" of promising configurations. Compared with the method of fair allocation of resources, this method improves accuracy by $25.4\times$. However, this method is difficult to apply to some areas of coal mines where the network environment is poor or even unable to connect network.

The above literature effectively resolved practical issues of industrial production and urban safety through cloud and edge computing. However, In the harsh coal mine environment, current object detection models and cloud-edge cooperation framework are difficult to perform effectively. To this end, the edge-cloud cooperation framework proposed in this paper is used to achieve real-time intelligent video surveillance, and it guarantee production safety in coal mine.

A summary of typical object detection methods is presented in TABLE I. Cloud computing or Edge-cloud are used in those methods of TABLE I. However, those methods in TABLE I are required smoothly and stable network environment. Meanwhile, the edge models in the proposed methods of TABLE I are also required high performance edge platform, it increases the cost of whole system.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

TABLE I
A SUMMARY OF TYPICAL HIGH SPEED OBJECT DETECTION METHODS

papers	Method	Implementation	FPS	mAP
8	YOLOv3	Titan X	35	55.3
8	YOLOv3-Tiny	Titan X	220	33.1
9	MD-YOLO	K40	200	79.5
11	IN-YOLO		25	88
24	MNPrioriBoxes-Yolo	GTX1080	151.9	86.6
25	MobileNetv2-TOLOv3	GTX1080TI	246	91.3
27	Faster-RCNN/ Cloud-Edge cooperation	i7-4790	0.67	100
28	Faster-RCNN/ Cloud-Edge cooperation	Titan X/ Xuelang Cloud/ Raspberry pi	17/ 8.5/ 1.1	68

III. CLOUD-EDGE COMPUTING COOPERATION FRAMEWORK OF COAL MINE

A. REAL-TIME EDGE SERVICE

Edge computing provides real-time intelligent processing services for coal mine video surveillance [37]. Edge services are composed of hardware layer, data interaction layer and service layer. It is show in Fig. 1.

Hardware layer is the basis of edge devices. It is composed of communication interface, control interface and image sensor. The data interaction layer communicates with cloud server through communication interface, which completes cloud-edge data transmission, control parameters transmission and model updating. Service layer is based on hardware layer, which acquires real-time images of the monitoring area, and processes them through MCU. The service layer controls coal mine equipment in real-time through the control interface according to the intelligent analysis results.

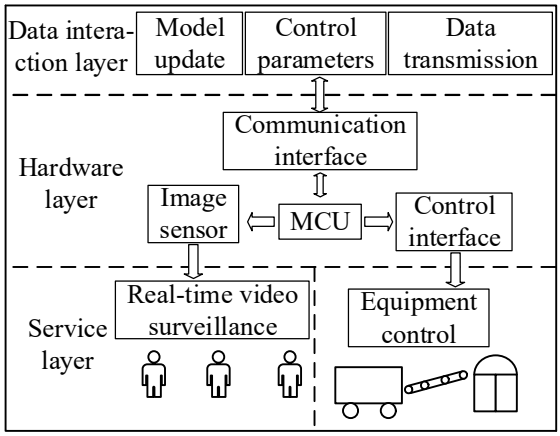


FIGURE 1. The Architecture of edge service. MCU control the edge service and process monitoring video in real-time.

Real-time edge service is the basis of entire edge-cloud cooperation system. It is used to process coal mine surveillance video in real-time, control coal mine equipment, and provide data for cloud servers.

B. NON-REAL-TIME CLOUD SERVICE

Cloud servers have strong computing and storage capabilities, but severe latency will occur during data transmission and process. Hence, cloud computing is not suitable for the tasks that require high real-time performance. Fig. 2. illustrates the work process of cloud-edge cooperation. In the cloud-edge cooperation system, cloud computing is primarily responsible for the following tasks:

(1) Integrating data from edge devices. The edge devices send data to cloud server through heterogeneous converged communication network. Then, cloud server classifies and stores those data to prepare for the optimization of edge models.

(2) Optimizing edge models. Cloud server has powerful computing capability, it continuously trains and optimize the edge models by the data transmitted from edge devices. Then, cloud server transmits the optimized models to the edge devices through heterogeneous converged communication network. It enables edge models to evolve constantly.

(3) Edge devices management. Edge devices are located in various scenarios. The efficiency of edge devices can promoted by centralized management through cloud servers.

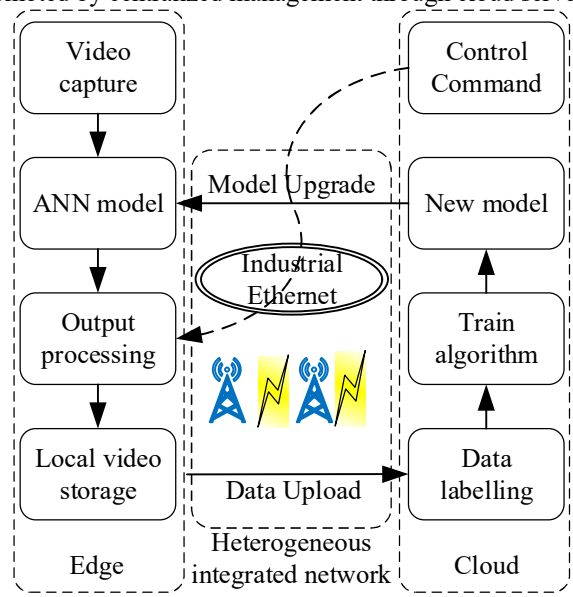


FIGURE 2. Process of cloud-edge cooperation framework. Cloud server receive the data transmitted by Edge, and use the data to train or improve model. Finally, the trained or improved models are transmitted to Edge devices by Heterogeneous integrated network.

In summary, cloud computing acts as a global orchestrator in cloud-edge cooperation system. Cloud server obtain data from edge devices and return the optimized models. Cloud computing enables cloud-edge cooperation system to form a virtuous cycle of data-model, which is an important guarantee for improving the quality of intelligent video monitoring systems.

C. THE HETEROGENEOUS CONVERGED NETWORK OF CLOUD-EDGE COOPERATION

Cloud-edge cooperation system provides real-time intelligent video surveillance at the edge. However, the process of data transmission and model training can be considered as non-real-time tasks. Depending on the coal mine network environment, edge nodes managed by cloud servers can be divided into two types. The first type of edge computing nodes are located in unblocked network environment. Those

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

nodes are able to provide real-time intelligent video surveillance and transmit the monitoring images to cloud server immediately. The second type of edge nodes are located in blocked network environment, so that it cannot transmit monitoring images immediately. But, those nodes also provide real-time intelligent video surveillance and equipment controlling at edge to ensure the safety of workers.

We proposed a heterogeneous converged network for the two types of edge computing nodes in coal mine. For the first type of edge nodes, the existing coal mine network is used to interact with cloud servers. However, the coverage of existing coal mine network is limited, which cannot support the second type of edge nodes communicating with cloud server directly. To this end, we merge various existing wired network and wireless network, and to build mobile opportunity networks based on mobile workers and vehicles [38]. Ultimately, a heterogeneous converged network is composed of wired network + wireless network, fixed communication nodes + mobile communication nodes, traditional network and opportunity network [39]. The heterogeneous converged network provides a channel for edge nodes to interact with cloud server. The heterogeneous converged network is shown in Fig. 3.

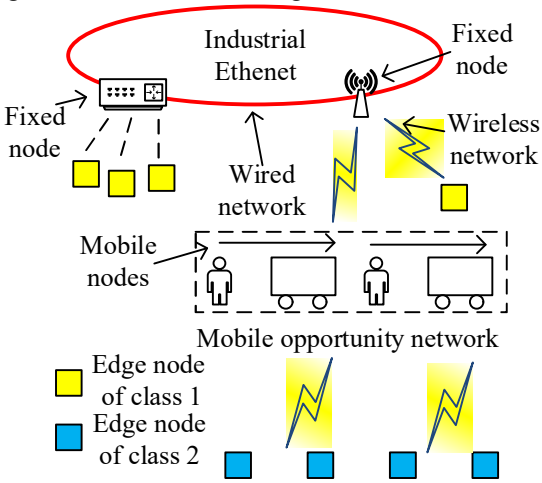


FIGURE 3. Heterogeneous converged network. Workers and vehicles compose the mobile opportunity network. Edge node of class2 transmit data through mobile opportunity network and Industrial Ethenet.

IV. THE PROPOSED METHOD FOR REAL-TIME INTELLIGENT VIDEO SURVEILLANCE

A. ANALYSIS OF THE OBJECT DETECTION MODEL OF Tiny-YOLOv3

Tiny-YOLOv3 is a lite version of YOLOv3. Compared with YOLOv3, Tiny-YOLOv3 is smaller and faster, with fewer parameters and calculations. Therefore, Tiny-YOLOv3 is easy to deploy on embedded platforms, and it has high real-time performance due to its low computational complexity.

Tiny-YOLOv3 divides the input image into $S \times S$ grids. Each grid contains 3 Bounding boxes, and each Bounding box contains 6 predicting parameters, which is $(x, y, w, h, I_{object}, class)$. As shown in Fig. 4, among those parameters, (x, y) is the distance of grid's border to the center of Bounding box. (w, h) is the ratio of the width and height of the Bounding box to the width and height of entire image. I_{object} is the confidence score of Bonding box. $class$ is the category of object. (X, Y) is the distance of grid's border that contains Bounding box to the border of image.

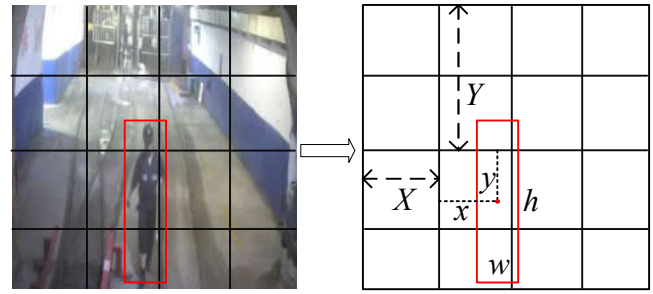


FIGURE 4. Predicting bounding box of Tiny-YOLOv3. Tiny-YOLOv3 predicting size and location of object in surveillance image.

In the Bounding box of Tiny-YOLOv3, the content of $class$ is shown in (1).

$$class = [p_1, p_2 ... p_j] \tag{1}$$

where, p_j is the confidence of the prediction for jth category.

B. ANALYSIS OF DEPTHWISE SEPARABLE CONVOLUTION

Depthwise separable convolution greatly reduces the number of model parameters and calculations with only a small loss of accuracy [23]. Therefore, a model consisting of depthwise separable convolution is well suited for intelligent computing at the edge.

Depthwise separable convolution consists of depthwise convolutions and pointwise convolutions. It is computed using depthwise convolution filters for each channel of the input image, and followed by pointwise convolution, while standard convolution is done in one step. The calculation process of the depth separable convolution is shown in Fig. 5.

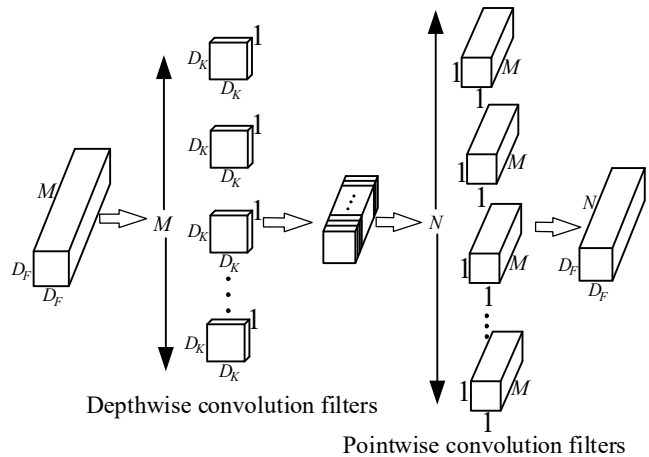


FIGURE 5. Calculation process of the depth separable convolution

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

Depth separable convolution decompose convolutional operation into Depthwise convolutional and Pointwise convolutional.

Where, D_F is the height and width of the input image. M is the number of the image channels. D_K is the size of the filters of the depthwise convolution. N is the number of pointwise convolution and output channels.

For an input image of size $D_F \times D_F \times M$, the ratio of computational cost between depthwise separable convolution and standard convolution [23] is shown in (2):

$$\frac{\mathrm{Cos}t_{secnn}}{\mathrm{Cos}t_{stcnn}} = \frac{D_F \times D_F \times M \times D_K \times D_K + D_F \times D_F \times M \times N}{D_F \times D_F \times M \times N \times D_K \times D_K} = \frac{1}{N} + \frac{1}{D_K^2} \tag{2}$$

Where, $\mathrm{Cos}t_{stcnn}$ is the computational cost of standard convolution; $\mathrm{Cos}t_{secnn}$ is the computational cost of depthwise separable convolution.

The ratio of the parameters number is shown in eq. 3:

$$\frac{Pnum_{secnn}}{Pnum_{stcnn}} = \frac{D_K \times D_K \times M + N \times M}{D_K \times D_K \times N \times M} = \frac{1}{N} + \frac{1}{D_K^2} \tag{3}$$

Where, $Pnum_{secnn}$ is the parameters number of depthwise convolution filters and pointwise convolution filters; $Pnum_{stcnn}$ is the parameters number of standard convolution filters.

The formula (2) and (3) show that, with the increase of N and D_K , the computational cost and parameters of depthwise separable convolution is decrease respect to standard convolution.

C. DOWN-SAMPLING INVERTED RESIDUAL BLOCK

The backbone of Tiny-YOLOv3 is used to extracts features from image, and those features down-sampled by standard convolution or max-pooling. However, Tiny-YOLOv3 is unable to extract the deeper features of the image due to the limited number of convolutional layers. In addition, max-pooling will cause information lost. In this paper, we down-sampled the input image by depthwise separable convolution. Compared with max-pooling, depthwise separable convolution can hold more information. Moreover, depthwise separable convolution is able to increase the depth of CNN model while maintain the size and computational cost. Hence, for the same size of two CNN-based models, depthwise separable convolution can improve the feature extraction ability of the model. However, with the increases of model depth, the model is prone to gradient disappearance and overfitting. Fortunately, residual structure can effectively solve those problems [40][41].

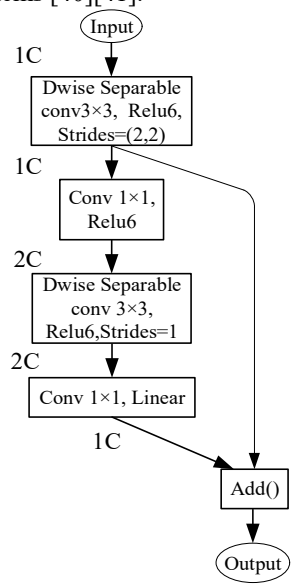


FIGURE 6. Down-sampling inverted residual block. Firstly, the Dwise Separable convolution down-sample and extract features of input data. Secondly, the residual block is used to extract deeper features.

Inspired by MobilenetV2 [23], we proposed down-sampling inverted residual block based on depthwise separable convolution. The structure is shown in Fig. 6. Firstly, the block uses depthwise separable convolution to down-sample the input image. Secondly, the number of channels of the input feature is expanded from 1C to 2C through 1×1 convolution. Thirdly, depthwise separable convolution is used to extract the features. Finally, the number of channels is restored to 1C through 1×1 convolution, and the features are accumulated with the output features of the secondly step. In order to reduce the loss of accuracy caused by float16 inference on embedded platforms, RELU6 is used as the activation function of the first three convolutional layers [42]. At the same time, the linear activation function is used as the output of the last layer. The linear activation function can avoid information destruction caused by the nonlinearity of RELU.

D. FAST-LIGHTWEIGHT YOLO

The safety of workers is an important factor of production safety in coal mine. Pedestrian detection gives timely alarm or shut down the equipment in operation based on the location of the pedestrian, which can effectively prevent workers from being injured.

We proposed a novel object detection model to detect coal mine workers in real-time at edge. The model is named FL-YOLO (Fast-Lightweight YOLO). The backbone of FL-YOLO is composed of down-sampling inverted residual blocks. The framework of FL-YOLO is shown in Fig. 7.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

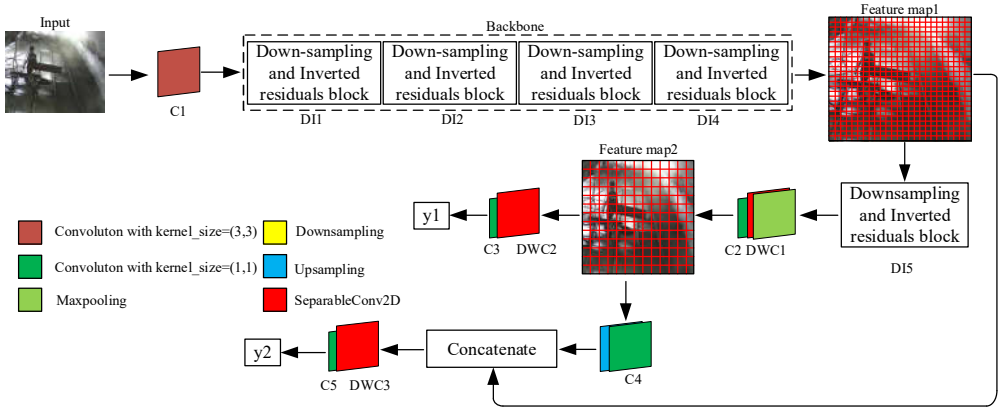


FIGURE 7. Framework of FL-YOLO. Down-sampling and Inverted residuals blocks compose the backbone of FL-YOLO, and multi-scale detection enables the model to detect objects of different sizes.

Where, $Ci\ (i=1,2,3,4,5)$ denotes ith convolution layer. $DWCj\ (j=1,2,3)$ denotes jth depthwise separable convolution layer. $DIn\ (n=1,2,3,4,5)$ denotes nth down-sampling and inverted residual block.

FL-YOLO is a multi-scale object detection method. It has excellent detection results for object of different sizes. Considering the versatility of the model, FL-YOLO uses a fully convolutional layer as the output layer, which allows FL-YOLO has the ability of process the input image with different size.

For an image with an input size $416\times416\times3$, firstly, FL-YOLO extracts the feature map1 with size 26×26 by Backbone. Secondly, after down-sampling and convolution operations, a feature map2 of size 13×13 is extracted. Thirdly, FL-YOLO up-samples feature map2 and fuses it with feature map1 to form a new feature map of size 26×26. Finally, FL-YOLO outputs $3\times13\times13$ bounding boxes y1, and $3\times26\times26$ bounding boxes y2.

Compared to Tiny-YOLOv3, the depth of FL-YOLO is greatly improved due to down-sampling inverted residual block and depthwise separable convolution, which improves the feature extraction capability of FL-YOLO. And The size of FL-YOLO is only 16.1MB, which is much smaller than Tiny-YOLO's 34MB and YOLOv3's 237MB. The details of the FL-YOLO model are shown in TABLEII.

TABLE II
DETAILS OF THE FL-YOLO

Layer	Filter size	channels	Feature size
C1	3×3	16	416×416
DI1	/	32	416×416
DI2	/	64	208×208
DI3	/	128	104×104
DI4	/	256	52×52
DI5	/	512	26×26
DWC1	3×3	1024	13×13
C2	1×1	256	13×13
DWC2	3×3	512	13×13
C3	1×1	18	13×13
C4	1×1	128	13×13
DWC3	3×3	256	26×26
C5	1×1	18	26×26

E. COMPLEXITY ANALYSIS OF FL-YOLO

Inference speed is largely influenced by model complexity. Model complexity includes computational complexity and spatial complexity. Because the model size of FL-YOLO is only 16.1MB, the spatial complexity of FL-YOLO is less than YOLOv3 and Tiny-YOLOv3. The computational complexity is determined by floating point operations (FLOPs). We analyze the computational complexity of FL-YOLO by calculating FLOPs during inference.

The backbone of FL-YOLO is composed of down-sampling inverted residual block. The FLOPs of the block is shown in (4):

$$FLOPs_{DIRB} = FLOPs_{DS1} + FLOPs_{COV1} + FLOPs_{DS2} + FLOPs_{COV2} + FLOPs_{ADD} \quad (4)$$

Where, $FLOPs_{DIRB}$ is the FLOPs of down-sampling inverted residual block; $FLOPs_{DS1}$ is the FLOPs of first depthwise separable convolution operation in the block; $FLOPs_{COV1}$ is the FLOPs of first convolution operation in the block; $FLOPs_{DS2}$ is the FLOPs of second depthwise separable convolution operation in the block; $FLOPs_{COV2}$ is the FLOPs of second convolution operation in the block; $FLOPs_{ADD}$ is the addition operation in the block.

For an input image with size of $D_F\times D_F\times M$, by Eq. (2), the FLOPs of first depthwise separable convolution is shown in (9):

$$FLOPS_{DS1} = 4.5D_F{}^2M + 0.5D_F{}^2M^2 \quad (5)$$

As Fig .6 shows that the first convolution changes the feature depth from 1M to 2M. By Eq. (3), $FLOPs_{COV1}$ can be calculated as:

$$FLOPS_{COV1} = 0.5D_F{}^2M^2 \quad (6)$$

The FLOPs of second depthwise separable convolution operation is shown in (7):

$$FLOPS_{DS2} = 4.5D_F{}^2M + D_F{}^2M^2 \quad (7)$$

$FLOPs_{COV2}$ and $FLOPs_{ADD}$ can be calculated as follows:

$$FLOPS_{COV2} = 0.5D_F{}^2M^2 \quad (8)$$

$$FLOPS_{ADD} = D_F{}^2M \quad (9)$$

From (5) to (9), the FLOPs of down-sampling inverted residual block is:

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

$$FLOPS_{DIRB} = 10D_F^{\ 2}M + 2.5D_F^{\ 2}M^2 \quad (10)$$

By (10), the FLOPs of an inference process is 0.92Bn for an image of size $416\times416\times3$. The FLOPs of FL-YOLO is much less than that of GradAM (2.6GFLOPs)[31], YOLOv3 (65.86Bn FLOPs) and Tiny-YOLOv3 (5.56Bn FLOPs)[8]. Therefore, FL-YOLO is more suitable for embedded platforms than YOLOv3 and Tiny-YOLOv3.

The size and computational cost of FL-YOLO is less than other object detection models such as YOLOv3 and Faster-RCNN. Furthermore, the lightweight characteristics of FL-YOLO allows it to be deployed on resource-constrained platform, even embedded systems. Therefore, FL-YOLO can be used in real-time control systems at edge or other scenario. To sum up the above, FL-YOLO has excellent scalability.

F. ALGORITHM OF FL-YOLO

FL-YOLO outputs a number of bounding boxes that contain objects information. However, the information in bounding boxes does not directly represent the position and the type of the objects in the image. The output bounding boxes of FL-YOLO need to be decoded.

The output bounding boxes of FL-YOLO:

$$Box^k = (x^k, y^k, w^k, h^k, I^k_{object}, class^k) \quad (11)$$

Where, k denotes the number of bounding boxes, $k \in [0, 2535)$.

The object's position in bounding boxes is decoded as follows:

$$b_x^k = \sigma(x^k) + X^k \quad (12)$$

$$b_y^k = \sigma(y^k) + Y^k \quad (13)$$

$$b_w^k = p_w e^{w^k} \quad (14)$$

$$b_h^k = p_h e^{h^k} \quad (15)$$

Where, σ denotes the function of sigmoid; X^k, Y^k are the distance of grid's border that contains Bounding box to the border of entire image; p_w, p_h are the width and height of anchor boxes respectively [8][43], b_x^k, b_y^k, b_w^k, b_h^k are the center point position and size of the bounding boxes.

Define the confidence that the predicted bounding box contains object as:

$$score^k = I^k_{object} \times p^k \quad (16)$$

Where, $score^k$ is the confidence for detecting the object in the Kth bounding box.

Split the information in the bounding box output by FL-YOLO, let:

$$b = (x, y, w, h) \quad (17)$$

So,

$$Box^k = (b^k, score^k) \quad (18)$$

For the large number of Bounding Boxes output by FL-YOLO, it is necessary to select the prediction box for each target. In this paper, we use the NMS (Non-Maximum Suppression) algorithm to eliminate repetitive and low-confidence prediction bounding boxes. The NMS algorithm pseudocode is shown in Algorithm 1.

Algorithm 1:NMS

1. **Input**: $B = \{b^1, b^2 ... b^k\}$; SC=$\{\text{score}^1, \text{score}^2, ..., \text{score}^k\}$;NMS threshold N_t ;
2. **Output**: Bounding boxes *D*, score *SC*, Index *indexes*
3. define function $NMS(B, SC, N_t)$

// indexes is the number of output Bounding Boxes.

4. $D \leftarrow \{\}$, $indexes \leftarrow \{1, 2, \ldots, k\}$
5. while $B \neq empty$ do

// Find the number of maximum score and assigned to m.

6. $m \leftarrow \arg\max(SC)$

//M denotes the Bounding Box with maximum score.

7. $M \leftarrow b^m$

//D is the candidate output of FL-YOLO.

8. $D \leftarrow D \cup M$

//Delete M(b^m) from B.

9. $B \leftarrow B - M$

// Delete the Bounding boxes whose iou value with M are //larger than the threshold N_t .

10. for b^i in B do
11. if $iou(M, b^i) \geq N_t$ then

//Delete the Bounding box from B.

12. $B \leftarrow B - b^i$

//Delete the score from SC.

13. $SC \leftarrow SC - score^i$

//Delete the number from indexes.

14. $indexes \leftarrow indexes - i$
15. end if
16. end for

//Until $B = empty$,ending the loop

17. end while
18. return *D*, *SC*, *indexes*

The FL-YOLO object detection algorithm pseudocode is shown in Algorithm 2.

Algorithm 2: The object detection algorithm of FL-YOLO

1. **Input**: image *I*
2. **Output**: boxes *D*, score *SC*, *classes*
3. Use FL-YOLO to predict:

$y1 \cup y2 = \{x_{ij}, y_{ij}, w_{ij}, h_{ij}, c_{ij}, p_{ij}\}$, $i \in [0,3), j = [0, 5S^2)$

4. Reshape:

$y1 \cup y2$ to $box\_xy = \{x_k, y_k\}$,

$box\_wh = \{w_k, h_k\}$, $confidence = \{c_k\}$, $class\_prob = \{p_k\}$,

$k \in [0, 15S^2)$

// Calculate the position and size of the object in the input image.

5. $box\_xy \leftarrow \begin{cases} x_k = \sigma(x_k) + g_{x_k} \\ y_k = \sigma(y_k) + g_{y_k} \end{cases}$, $box\_wh \leftarrow \begin{cases} w_k = p_{w_k} e^{w_k} \\ h_k = p_{w_k} e^{w_k} \end{cases}$

6. Concatenate *box_xy*, *box_wh* to *boxes*
7. $score \leftarrow \{c_k \times p_k\}$

//NMS algorithm is used to select Bounding Boxes.

8. Obtaining *D*, *SC* and *indexes* by $NMS(boxes, score, N_t)$

//Obtain the object category.

9. for *j* in *indexes* do
10. $classes \leftarrow \arg\max(p_j)$
11. end for
12. Return *D*, *SC*, *classes*

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

F. LOSS FUNCTION

The loss function plays an important role in training process. FL-YOLO predicts the position, size, and type of objects, and its loss function should contain these items.

(1) Position loss function

$$Loss_1 = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} (2 - w_i \times h_i)[(w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2] \quad (19)$$

Where, s^2 is the number of grids in the image; B is the number of bounding boxes predicted for each grid; When the IOU between the j-th prediction box of the i-th grid and the ground truth box is the largest, $I_{ij}^{obj} = 1$, otherwise $I_{ij}^{obj} = 0$; x_i, y_i, w_i, h_i are the predicted center point position and length and width of the object; $\hat{x}_i$, $\hat{y}_i$, $\hat{w}_i$, $\hat{h}_i$ are the ground truth center position and length and width of the object; We set $\lambda_{coord} = 5$ to balance the object position loss function.

(2) Object confidence loss function

$$Loss_2 = -\sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} [\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] - \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} [\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] \quad (20)$$

Where, C_i denotes the predicted confidence; $\hat{C}_i$ represents the truth confidence; We set $\lambda_{noobj} = 0.5$ to balance the loss function of the area without objects in the image.

$$Loss_3 = -\sum_{i=0}^{s^2} I_{ij}^{obj} \sum_{c \in classes}^{B} [\hat{p}_i^j(c) \log(p_i^j(c)) + (1 - \hat{p}_i^j(c)) \log(1 - p_i^j(c))] \quad (21)$$

Where, classes indicates the number of categories; p_i^j denotes the probability of the prediction category; $\hat{p}_i^j$ represents the classification probability of ground truth boxes.

The loss function of FL-YOLO is obtained by overlaying (16)-(18):

$$Loss = Loss_1 + Loss_2 + Loss_3 \quad (22)$$

G. MODEL OPTIMIZED FOR SCENE

In the cloud-edge cooperation system, FL-YOLO needs to be deployed on edge to ensure real-time performance. However, due to the limitation of computing capacity and storage resources of edge, lightweight model has poor data generalization capabilities and low object detection accuracy. Considering the characteristics of coal mine video surveillance, the monitoring area of each monitoring device remains unchanged for a long time. Therefore, an object detection model deployed on edge device only needs to exhibit excellent performance in corresponding monitoring scenarios. In this paper, we classify the FL-YOLO model into generic model and dedicated model. The generic model uses refers to a model trained with pictures of various scene in coal mine. So that the generic model can be monitored in any area. Based on the transfer learning [44] [45], the dedicated model is optimized for the dataset with single scene. The single scene denotes the area which monitored by an edge device with general model. The parameters of dedicated model are adjusted to make the edge model more suitable for the monitored scene. Therefore, the accuracy of the model is improved by optimization.

After the generic model is deployed to the surveillance scene, the edge device continuously transmit the images of monitored area to cloud server. These images are used to further train the edge model which deployed in the scene. Finally, these optimize models are transmitted to the edge. The process is shown in Fig. 8.

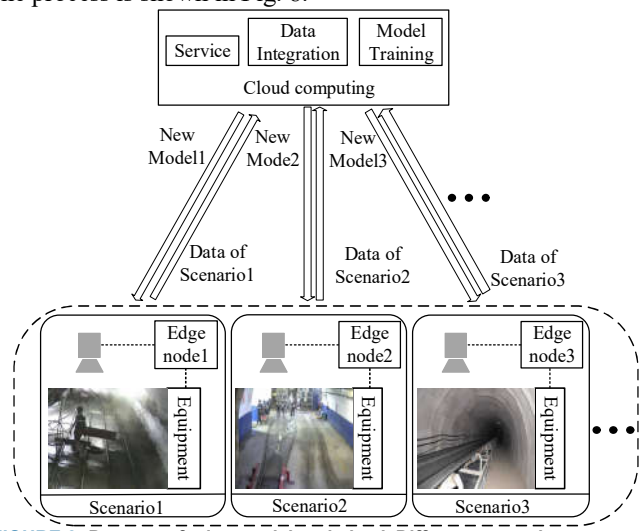


FIGURE 8. Process of edge model optimized. Different scenario transmit different data to cloud server, and the cloud server optimizes the model based on those data for different scenario.

V. EXPERIMENTAL RESULTS

A. COAL MINE PEDESTRIAN DATASET

(1) Multi-scene pedestrian dataset

In order to train FL-YOLO and verify the optimization effect of cloud-edge cooperation framework, we collected 6,000 coal mine surveillance pictures. After sorting, it includes 2,598 pictures of pedestrians and 3,402 pictures of nobody. Four augmentation skills are used to process those pictures, including src, processed by CLAHE, crop and flip. Fig. 9 shows the examples of augmented images. Moreover, we randomly added Gaussian noise and Gaussian blur on images. Finally, the multi-scene pedestrian dataset has 24000 pictures, it contains 10392 pictures with pedestrian and 13608 pictures with nobody.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

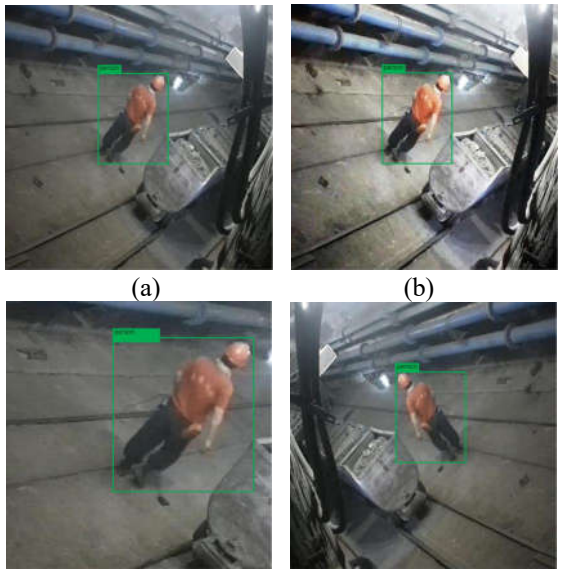


FIGURE 9. Examples of augmented images. (a) Source (b) Processed by CLAHE (c) Crop (d) flip

(2) Single-scene pedestrian dataset.

To verify the impact of further training, we collected 501 images of a particular situation in coal mine. Those images include 500 images of pedestrians, and one image of no pedestrian. We processed the above 501 images with same way as A, forming a Single-scene pedestrian dataset of 2000 pedestrian images and 4 images without pedestrians. Then, random noise is superimposed on the non-pedestrians images, to expand it to the same number of pedestrians images. As a result, the Single-scene dataset contains 4,000 images. Fig. 10 shows the examples of Single-scene dataset.

FIGURE 10. Examples of Single-scene pedestrian dataset. (a) Including pedestrians (b) Non-pedestrian

We process those images and train models on Intel-i7 9700K (4.9 Ghz) with NVIDIA GTX 1080Ti. After training, we deployed the model on NVIDIA GTX 1080Ti and NVIDIA Jetson TX1 to test the performance of the model.

B. GENERIC MODEL PERFORMANCE VALIDATION

In order to verify the performance of FL-YOLO, we analyze the training process of FL-YOLO and other object detection models. Furthermore, the detection speed and accuracy of FL-YOLO is also compared with other models.

Training process reflects the performance of the model to some extent. The faster the loss function converges and the lower of the loss value is, the higher the accuracy of the model is. To train the FL-YOLO model, we randomly divide the dataset into training set and test set, where the test set accounts for 20 percent of the multi-scene pedestrian dataset. Fig. 11 shows the Loss-Epochs curves of FL-YOLO, YOLOv3 and Tiny-YOLOv3. From the training process, it can be seen that the loss function of Tiny-YOLOv3 decreases unsteadily, converges slowly. The final loss value of Tiny-YOLOv3 is higher than that of TOLOv3 and FL-YOLO. The size of YOLOv3 is the largest among the three models, with better feature extraction capabilities, and the convergence process is faster and more stable than Tiny-YOLOv3. For FL-YOLO, the feature extraction capability is enhanced by depthwise separable convolution. However, the size of FL-YOLO is lighter, and the residual structure allows FL-YOLO has better data generalization capability. Thereby, the training process of FL-YOLO performs better than Tiny-YOLOv3 in terms of stability and speed, and the final loss value of the former is also smaller than that of the latter.

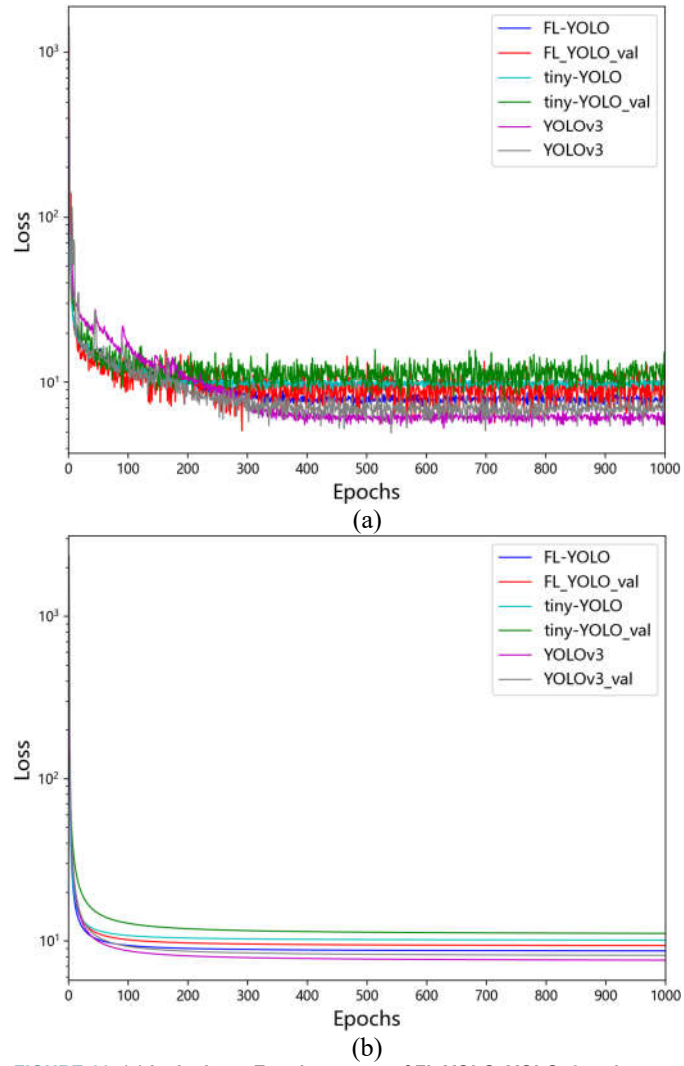


FIGURE 11. (a) is the Loss-Epochs curves of FL-YOLO, YOLOv3 and Tiny-YOLOv3. In order to make the curve more intuitive, (b) is the curve fitting in (a) to make it more smooth and reflect the trend of training.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

AP and mAP are important metric to reflect detection accuracy of object detection models. The higher value of AP or mAP is, the higher accuracy of the model will make. In this paper, we only detect coal mine underground pedestrians. Therefore, AP and mAP are equal in this case. FPS (Frame Per Second) reflects the detection speed of models, and the real-time performance is positively correlated with FPS. FL-YOLO and other models are deployed on GTX 1080ti GPU and NVIDIA Jetson TX1 respectively, to measure the value of AP and FPS.

TABLE III shows the results of various object detection models on Multi-scene pedestrian dataset. A higher AP value indicates a higher accuracy of the model, and the value of FPS indicates the real-time performance of the model. Fig. 12 shows the PR curves of these models. Experiment results show, YOLOv3 has the best performance for AP, P, and R values, and it also runs faster than SSD and Faster R-CNN. Then, mobilenetV2 is used to construct the YOLOv3-mobilenetv2. YOLOv3-mobilenetv2 has better real-time performance than YOLOv3, but it is still unable to be applied on NVIDIA Jetson TX1. Tiny-YOLOv3 is a lite version of YOLOv3 with considerably reduced computational cost and model size. It runs more than five times faster than YOLOv3 and nearly three times faster than YOLOv3-mobilenetv2. However, the accuracy of Tiny-YOLOv3 is the worst of these models, so that it is difficult to applied. The AP value of FL-YOLO is behind only YOLOv3 and Faster R-CNN, but the speed of FL-YOLO is the fastest among all these models including the Efficient YOLO(401.21 ms/frame on NVIDIA Jetson TX2)[30]. The excellent accuracy and real-time performance of FL-YOLO make it suitable for real-time pedestrian detection in coal mine.

TABLE III
RESULTS OF SEVERAL OBJECT DETECTION MODEL

Model	AP	P	R	FPS-1080Ti	FPS-TX1
Faster R-CNN	79.3	83.4	81.7	9.4	1.6
SSD	72.9	77.9	73.6	15.3	3.4
SSD-mobilenetV2	59.3	68.3	63.9	32.6	6.9
YOLOv3	**82.4**	**88.6**	**84.5**	35.5	6.2
YOLOv3-mobilenetV2	70.3	75.1	71.8	76.1	9.8
Tiny-YOLOv3	52.8	55.3	61.7	197	27.3
FL-YOLO	76.7	79.2	81.3	**217.7**	**36.7**

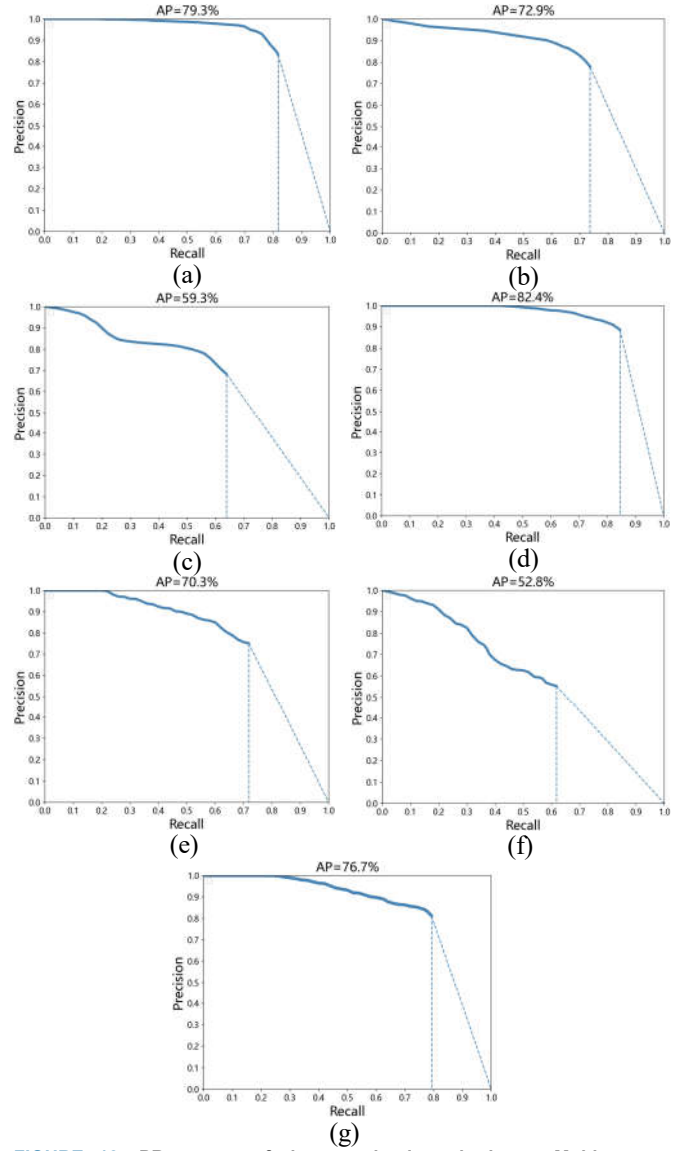


FIGURE 12. PR curves of the examined methods on Multi-scene pedestrian dataset. (a) Faster R-CNN (b) SSD (c) SSD-mobilenetV2 (d) YOLOv3 (e) YOLOv3-mobilenetV2 (f) Tiny-YOLOv3 (g) FL-YOLO

In order to verify the performance of FL-YOLO more comprehensively, we randomly divided Multi-scene pedestrian dataset into 5 groups of 4800 images each. Then, we apply the method of k-fold [15], with k=5, and train the FL-YOLO, YOLOv3, Tiny-YOLOv3 until convergence at each fold. At the same time, we test the AP values of each fold. The results of k-fold cross validation are show in TABLE IV.

TABLE IV
RESULTS OF K-FOLD CROSS VALIDATION

The times of Fold	AP of FL-YOLO(%)	AP of Tiny-YOLOv3(%)	AP of YOLOv3(%)
1-fold	79.3	48.4	80.3
2-fold	76.7	56.7	86.6
3-fold	75.4	57.9	85.3
4-fold	82.0	58.1	78.9
5-fold	74.6	49.3	80.1
Mean	77.6	53.8	82.2
σ	3.04	4.34	3.46

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

TABLE III and TABLE IV show that the AP value of YOLOv3 are higher than that of FL-YOLO and Tiny-YOLOv3. However, the AP value of FL-YOLO is close to YOLOv3 and much higher than Tiny-YOLOv3. Moreover, the real-time performance of FL-YOLO is superior to that of YOLOv3 and Tiny-YOLOv3.

The above results show that FL-YOLO has excellent performance, and it can ensure the accuracy of detecting pedestrians in coal mine. The reasons for the outstanding performance of FL-YOLO can be summarized as follows:

(1) Depthwise separable convolution reduces the number of parameters and computing cost, and increases the speed of FL-YOLO.

(2) The down-sampling inverted residual block greatly improves the capability of feature extraction and data generalization.

(3) Multi-scale detection allows FL-YOLO to detect objects with different sizes effectively.

C. SCENE OPTIMIZATION MODEL PERFORMANCE VALIDATION

There is a large demand for intelligent video surveillance in coal mine, and the surveillance scenes are different. Therefore, the generic model is not able to achieve the ideal accuracy, because the lightweight model is lack of data generalization ability. Considering that FL-YOLO is deployed in different regions, the model parameters can be adjusted according to the environment to suit different monitoring area. For a given surveillance scene, the viewpoint tends not to change in the short term, so the background of the surveillance scene can be considered as fixed. However, the recognition mistake of background may occur when using a generic model for intelligent surveillance.

According to the transfer learning, FL-YOLO takes a Multi-scene pedestrian dataset as the source domain of the model, while single-scene pedestrian dataset can be considered as the target domain. Therefore, we optimized the model's parameters by instance-based transfer learning. On the basis of the generic model, we further train it using the single-scene pedestrian dataset. Consider that only the Tiny-YOLO and FL-YOLO have the speed to meet the practical application on embedded platforms. Hence, we just analyze the further training process of Tiny-YOLOv3 and FL-YOLO.

To optimize the FL-YOLO, we divide the single-scene pedestrian dataset into a training set and a test set, where the test set takes up 20 percent of the single-scene pedestrian dataset. As shown in Fig. 13, the loss functions of FL-YOLO and Tiny-YOLO are further reduced.

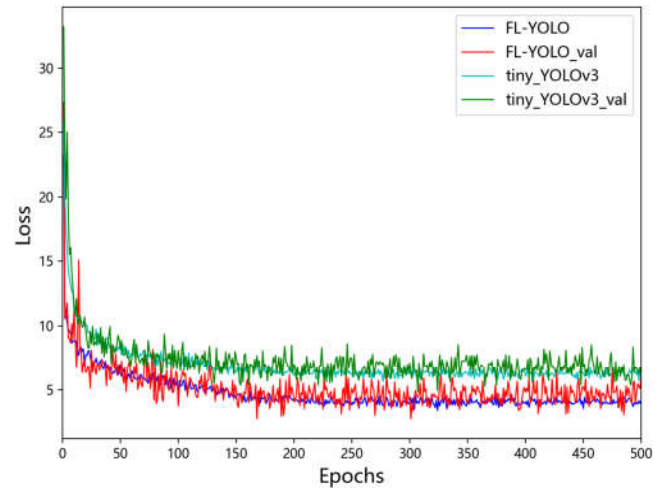


FIGURE 13. (a) is the Further training Loss-Epochs curves of FL-YOLO and Tiny_YOLOv3. In order to make the curve more intuitive, (b) is the curve fitting in (a) to make it more smooth and reflect the trend of training.

The P-R curve of optimized FL-YOLO and Tiny-YOLOv3 is shown in Fig. 14. Fig. 13 and Fig. 14 show that the FL-YOLO and Tiny-YOLOv3 models become more adaptable to the scene, and the performance is improved in this scenario.

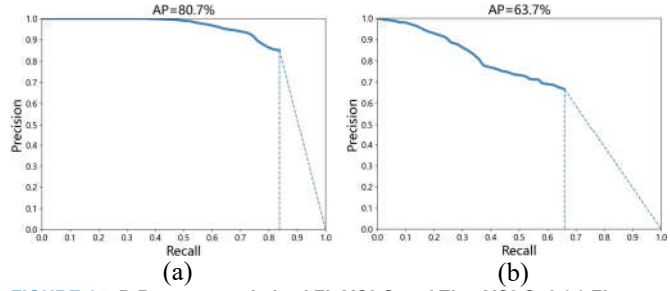


FIGURE 14. P-R curves optimized FL-YOLO and Tiny-YOLOv3 (a) FL-YOLO (b) Tiny-YOLOv3

We also use the method of k-fold (k=5) to further validate the effect of scene optimization. First, we randomly divided the single-scene pedestrian dataset into 5 groups of 800 images each. Then, we optimize FL-YOLO and Tiny-

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

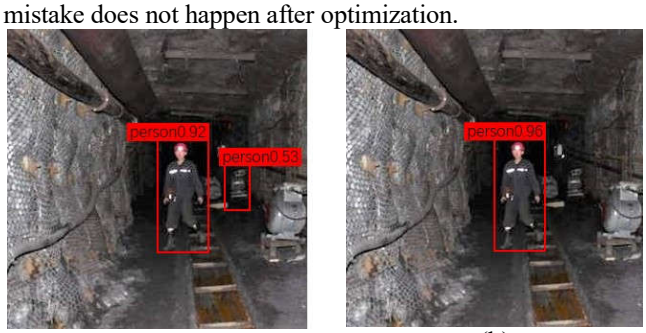
YOLOv3 until convergence at each fold. Finally, we test the AP values of FL-YOLO and Tiny-YOLOv3 at each fold. The results of optimization k-fold cross validation are show in TABLE V.

TABLE V
RESULTS OF K-FOLD CROSS VALIDATION

The times of Fold	AP of FL-YOLO(%)	AP of Tiny-YOLOv3(%)
1-fold	80.3	67.5
2-fold	82.9	66.9
3-fold	81.2	60.8
4-fold	80.1	59.8
5-fold	83.7	61.4
Mean	81.6	63.3
σ	1.60	3.63

TABLE V shows that the average values of AP for FL-YOLO and Tiny-YOLOv3 were significantly improved. Meanwhile, the Standard Deviation of FL-YOLO and Tiny-YOLOv3 is also decreased. The above results show that the optimization leads to improve accuracy and generalization of models in a single scenario.

As the Fig. 15 shows, YOLO incorrectly identifies objects in the background as pedestrians before optimized, but this mistake does not happen after optimization.



(a) (b)

FIGURE 15. Example of optimization effect. (a) Before optimization, FL-YOLO misidentify a machine as person. (b) After optimization, FL-YOLO solve the misidentification problem and increases the confidence level of correct identification.

Through the cooperation of edge computing and cloud computing, the edge is used to complete real-time intelligent video monitoring, while the cloud computing is responsible for model optimization. Experimental results show that FL-YOLO has excellent performance. After optimization, the real-time performance and accuracy of FL-YOLO are all superior to those of other mainstream object detection models. Meanwhile, the Robustness of FL-YOLO is also improved, and the influence of the noise and adversarial attacks acting on edge model is reduced by further train of edge-cloud framework.

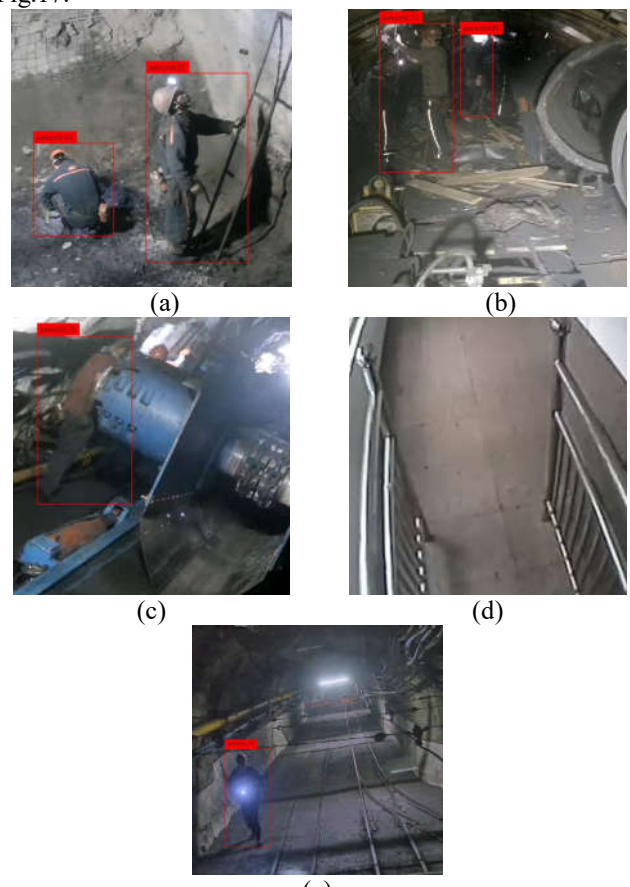
D. SYSTEM TEST AND COMPARISON

(1) System implementation: A Surveillance Video Real-time Analysis System was implemented on coal mine. We use the NVIDIA Jetson TX1 as edge node, and a computer with Intel i7-9700k @4.9Ghz CPU, NVIDIA GTX1080Ti GPU was used as a cloud server. The deployment locations of the edge nodes include Rock roadway, machine lane, Transfer Point of Outer Lane, Ground entry channel and haulage tunnel. The surveillance screens are show in Fig. 16.

In the above five scenarios, Ground Entry Channel and Haulage Tunnel have access to Industrial Ethenet. They have a low-latency and stable network. The edge nodes implemented at Ground Entry Channel and Haulage Tunnel can be seen as the edge node of class 1.

However, the Industrial Ethenet does not cover the Machine Lane and Transfer Point of Outer Lane. Hence, wireless communication is used to access network. Therefore, the edge nodes deployed on the Machine Lane and Transfer Point of Outer Lane can be considered the edge node of class 1.

The edge node deployed on Rock Roadway can been seen as a temporary node added during construction. Therefore, those edge nodes cannot be connected to Industrial Ethnet and wireless network. So, the edge node implemented at Rock Roadway can be seen as the edge node of class 2. However, there is frequent movement of workers and vehicles in Rock Roadway. Hence, we use the mobility of workers and vehicles to build heterogeneous converged networks to complete the transmission tasks of key videos, images and models. The information transmission method of the edge node implemented as Rock Roadway is shown in Fig.17.



(a) (b) (c) (d) (e)

FIGURE 16. The surveillance screens (a) Rock Roadway (b) Machine Lane (c) Transfer Point of Outer Lane (d) Ground Entry Channel (e) Haulage Tunne

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

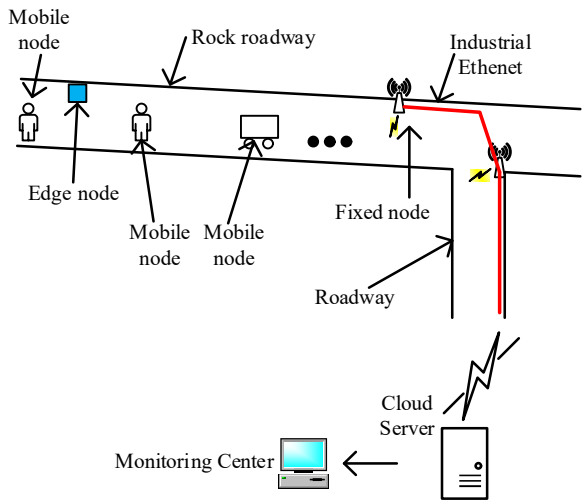


FIGURE 17. Information transmission method of Rock Roadway. Edge nodes deployed on the Rock Roadway transmit the information of the nodes by mobbing vehicles or workers.

As the Fig.17 shows, we install mobile information transmission nodes on workers and vehicles. Then, the information of the edge node is carried by mobile nodes, and those information will be forwarded to the fixed nodes through the movement of workers and vehicles. Finally, the fixed nodes send information to cloud server or monitoring center through Industrial Ethenet.

(2) Comparison of Edge-cloud cooperation and cloud computing

We deploy edge nodes at the above five sites. On the one hand, Surveillance video is processed at edge in the edge-cloud cooperation system. On the other hand, surveillance video is sent to cloud, and the cloud server is used to process the surveillance video. The latency of cloud computing and edge-cloud cooperation to process surveillance video is shown in TABLE VI.

TABLE VI
LATENCY OF PROCESS SURVEILLANCE VIDEO

Position	Cloud computing	Edge-Cloud Cooperation
Rock Roadway		28.73ms
Machine Lane	5964.79ms	28.62ms
Transfer Point of Outer Lane	11471.94ms	28.79ms
Ground Entry Channel	60.13ms	28.65ms
Haulage Tunnel	69.31ms	28.81ms

The latency times shown in TABLE VI are the average times used by the two systems for intelligent processing of 5000 images respectively. As shown in TABLE VI, the latency of Edge-cloud cooperation is lower than that of Cloud computing. Moreover, the latency of Cloud computing is different due to the network conditions in different regions.

In the Edge-cloud cooperation environment, the latency of video upload is still affected by network conditions. However, the video has been processed by edge nodes, it is able to respond to the video events in real time at the surveillance sites. In addition, with the growth of system deployment time, the edge model is continuously optimized. Hence, the edge-cloud cooperation surveillance analysis system has an excellent performance of real-time and accuracy, it ensures the production safety of coal mine to a great extent.

VI. CONCLUSION

In this paper, we have introduced a cloud-edge cooperation framework for real-time intelligent video surveillance in underground coal mine environment. The main work of this paper are as follows:

(1) A new cloud-edge cooperation framework is proposed. In the real-time intelligent video surveillance, this framework realize model optimization. For the scenarios with poor network environments, this framework still enables data interaction between edge and cloud.

(2) On the basis of YOLO, FL-YOLO real-time object detection model is proposed.

(3) Pedestrian dataset is built to train FL-YOLO, and validated the performance of FL-YOLO.

Compared with the traditional video surveillance method, cloud-edge computing framework has excellent performance of real-time and accuracy.

In the future, the main work is further optimization the cloud-edge cooperation framework by two steps. Firstly, we will further optimize the object detection model with the goals of lightweight, high accuracy, and high speed. Secondly, we will optimize the data transmission method between edge and cloud, to improve the coverage of real-time intelligent video surveillance and the speed of data transmission.

REFERENCES

[1] S. S. Thomas, S. Gupta and V. K. Subramanian, "Smart surveillance based on video summarization," in 2017 IEEE Region 10 Symposium (TENSYMP), Cochin, India, 2017, pp. 1-5, doi: 10.1109/TENCONSpring.2017.8070003.

[2] T. Akiyama, Y. Kobayashi, J. Kishigami and K. Muto, "CNN-Based Boat Detection Model for Alert System Using Surveillance Video Camera," in 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), Nara, Japan, 2018, pp. 669-670, doi: 10.1109/GCCE.2018.8574704.

[3] A. Mhalla, T. Chateau, S. Gazzah and N. E. B. Amara, "An Embedded Computer-Vision System for Multi-Object Detection in Traffic Surveillance," IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 11, pp. 4006-4018, Nov. 2019, doi: 10.1109/TITS.2018.2876614.

[4] A. Shahbaz and K. Jo, "Deep Atrous Spatial Features based Supervised Foreground Detection Algorithm for Industrial Surveillance Systems," in IEEE Transactions on Industrial Informatics, doi: 10.1109/TII.2020.3017078.

[5] 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.

[6] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[7] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

[8] J. Redmon and A. Farhadi, ''YOLOv3: An incremental improvement,'' in Proc. CVPR, 2018, pp. 1–6.

[9] L. Xie, T. Ahmad, L. Jin, Y. Liu and S. Zhang, "A New CNN-Based Method for Multi-Directional Car License Plate Detection," IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 2, pp. 507-517, Feb. 2018, doi: 10.1109/TITS.2017.2784093.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

[10] L. Zhou, W. Min, D. Lin, Q. Han and R. Liu, "Detecting Motion Blurred Vehicle Logo in IoV Using Filter-DeblurGAN and VL-YOLO," IEEE Transactions on Vehicular Technology, vol. 69, no. 4, pp. 3604-3614, April 2020, doi: 10.1109/TVT.2020.2969427.

[11] D. Sadykova, D. Pernebayeva, M. Bagheri and A. James, "IN-YOLO: Real-Time Detection of Outdoor High Voltage Insulators Using UAV Imaging," IEEE Transactions on Power Delivery, vol. 35, no. 3, pp. 1599-1601, June 2020, doi: 10.1109/TPWRD.2019.2944741.

[12] C. Zhang, Q. Cao, H. Jiang, W. Zhang, J. Li and J. Yao, "A Fast Filtering Mechanism to Improve Efficiency of Large-Scale Video Analytics," IEEE Transactions on Computers, vol. 69, no. 6, pp. 914-928, 1 June 2020, doi: 10.1109/TC.2020.2970413.

[13] W. Fang, L. Wang and P. Ren, "Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments," IEEE Access, vol. 8, pp. 1935-1944, 2020, doi: 10.1109/ACCESS.2019.2961959.

[14] Z. Shao, L. Wang, Z. Wang, W. Du and W. Wu, "Saliency-Aware Convolution Neural Network for Ship Detection in Surveillance Video," IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 3, pp. 781-794, March 2020, doi: 10.1109/TCSVT.2019.2897980.

[15] G .Olague, D. Hernández, P. Llamas and E Clemente. "Brain programming as a new strategy to create visual routines for object tracking," Multimed Tools Applications ,vol.78, pp. 5881–5918, 2019. https://doi.org/10.1007/s11042-018-6634-9

[16] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[17] M. Satyanarayanan, "The Emergence of Edge Computing," Computer, vol. 50, no. 1, pp. 30-39, Jan. 2017, doi: 10.1109/MC.2017.9.

[18] J. Ren, Y. Guo, D. Zhang, Q. Liu and Y. Zhang, "Distributed and Efficient Object Detection in Edge Computing: Challenges and Solutions," IEEE Network, vol. 32, no. 6, pp. 137-143, November/December 2018, doi: 10.1109/MNET.2018.1700415.

[19] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in 2016 10th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, 2016, pp. 1-8, doi: 10.1109/ISCO.2016.7727082.

[20] N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, "Mobile Edge Computing: A Survey," IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450-465, Feb. 2018, doi: 10.1109/JIOT.2017.2750180.

[21] L. Xu, J. Han, T. Wang and L. Bai, "An Efficient CNN to Realize Speckle Correlation Imaging Based on Cloud-Edge for Cyber-Physical-Social-System," IEEE Access, vol. 8, pp. 54154-54163, 2020, doi: 10.1109/ACCESS.2020.2979786.

[22] H. El-Sayed et al., "Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment," IEEE Access, vol. 6, pp. 1706-1717, 2018, doi: 10.1109/ACCESS.2017.2780087.

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[24] G. Li, Y. Yang and X. Qu, "Deep Learning Approaches on Pedestrian Detection in Hazy Weather," IEEE Transactions on Industrial Electronics, vol. 67, no. 10, pp. 8889-8899, Oct. 2020, doi: 10.1109/TIE.2019.2945295.

[25] J. Liu and X. Wang, "Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model," Plant Methods. vol. 16. 2020, doi: 10.1186/s13007-020-00624-2.

[26] S. Han, H. Mao and W. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in 4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2017.

[27] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan and Y. Yang, "Asymptotic Soft Filter Pruning for Deep Convolutional Neural Networks," IEEE Transactions on Cybernetics, vol. 50, no. 8, pp. 3594-3604, Aug. 2020, doi: 10.1109/TCYB.2019.2933477.

[28] H. Li, A. Kadav, I. Durdanovic, H. Samet and H.P Graf, "Pruning Filters for Efficient ConvNets," in 5th International Conference on Learning Representations (ICLR), Toulon, France, 2016.

[29] J. Luo, J. Wu and W. Lin, "ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression," in 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 5068-5076, doi: 10.1109/ICCV.2017.541.

[30] Z. Wang, J. Zhang, Z. Zhao and F. Su, "Efficient Yolo: A Lightweight Model For Embedded Deep Learning Object Detection," in 2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), London, UK, 2020, pp. 1-6, doi: 10.1109/ICMEW46912.2020.9105997.

[31] S. Rui, T. Li, and Y. Yamaguchi. "An attribution-based pruning method for real-time mango detection with YOLO network," Computers and Electronics in Agriculture, vol. 169, 2020.

[32] X. Wang, L. T. Yang, X. Xie, J. Jin and M. J. Deen, "A Cloud-Edge Computing Framework for Cyber-Physical-Social Services," IEEE Communications Magazine, vol. 55, no. 11, pp. 80-85, Nov. 2017, doi: 10.1109/MCOM.2017.1700360.

[33] 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 Transactions on Industrial Informatics, vol. 16, no. 7, pp. 4726-4735, July 2020, doi: 10.1109/TII.2019.2947539.

[34] 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," Advanced Engineering Informatics, vol. 43, 2020, https://doi.org/10.1016/j.aei.2020.101037.

[35] L Ye, "Study on embedded system in monitoring of intelligent city pipeline network," Computer Communications, vol. 153, pp. 451-458, 2020, https://doi.org/10.1016/j.comcom.2020.02.004.

[36] C. Hung, A. Ganesh and P. Bodik., "VideoEdge: Processing Camera Streams using Hierarchical Clusters," in 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, USA, 2018, pp. 115-131, doi: 10.1109/SEC.2018.00016.

[37] G. Ananthanarayanan et al., "Real-Time Video Analytics: The Killer App for Edge Computing," Computer, vol. 50, no. 10, pp. 58-67, 2017, doi: 10.1109/MC.2017.3641638.

[38] S. Trifunovic, S. T. Kouyoumdjieva, B. Distl, L. Pajevic, G. Karlsson and B. Plattner, "A Decade of Research in Opportunistic Networks: Challenges, Relevance, and Future Directions," IEEE Communications Magazine, vol. 55, no. 1, pp. 168-173, January 2017, doi: 10.1109/MCOM.2017.1500527CM.

[39] S. Wang, X. Wang, J. Huang, R. Bie, Z. Tian and F. Zhao, "The Potential of Mobile Opportunistic Networks for Data Disseminations," IEEE Transactions on Vehicular Technology, vol. 65, no. 2, pp. 912-922, Feb. 2016, doi: 10.1109/TVT.2015.2401605.

[40] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016 pp. 770-778. doi: 10.1109/CVPR.2016.90

[41] L. Chen, Q. Ding, Q. Zou, Z. Chen and L. Li, "DenseLightNet: A Light-Weight Vehicle Detection Network for Autonomous Driving," IEEE Transactions on Industrial Electronics, vol. 67, no. 12, pp. 10600-10609, Dec. 2020, doi: 10.1109/TIE.2019.2962413.

[42] Q. Mao, H. Sun, L. Zuo and R. Jia, "Finding every car: a traffic surveillance multi-scale vehicle object detection method," Applied Intelligence, vol. 50, pp. 3125–3136 ,2020, Available: https://doi.org/10.1007/s10489-020-01704-5

[43] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Industrial Electronics, vol. 67, no. 12, pp. 10600-10609, Dec. 2020, doi: 10.1109/TIE.2019.2962413.

[44] W Pan and Q Yang, "Transfer learning in heterogeneous collaborative filtering domains," Artificial Intelligence, vol. 197, pp. 39-55, 2013, Available: https://doi.org/10.1016/j.artint.2013.01.003.

[45] Y. Jiang et al., "Seizure Classification From EEG Signals Using Transfer Learning, Semi-Supervised Learning and TSK Fuzzy System," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 25, no. 12, pp. 2270-2284, Dec. 2017, doi: 10.1109/TNSRE.2017.2748388.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3077499, IEEE Access

Zhi XU received the Bachelor's degree of Eletrical Engineering in the college of Electrical and Information Engineering, Anhui University of Science and Technology (AUST), Huainan, China, in 2018. Now he is pursuing ph.D degree in Mechanical engineering in AUST. His current research interests are in Edge computing and Artificial Intelligence.

Jingzhao Li received his M.A. degree from China University of Mine and Technology in 1992, and PhD. degree in the key Lab of Power Electronics and Power Drives at 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. His research interests include Computer Control, Internet of Things Technology and Embedded Systems. 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 are cited more than 100 times by others.

MEI ZHANG was born in Suzhou, Anhui, China, in 1979. She received her bachelor's degree in electrical engineering from Anhui University of Science and Technology in 2002, and the master's degree in electrical engineering from Anhui University of Science and Technology in 2005. Now she is currently an associate professor in School of Electrical and Information Engineering, Anhui University of Science and Technology. Her research interests include intelligent control, the Internet of Things (IoT) technology, and embedded systems.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/