Lightweight Deep Learning based Intelligent Edge Surveillance Techniques

Yu Zhao, *Student Member*, *IEEE*, Yue Yin, *Student Member*, *IEEE*, and, Guan Gui *Senior Member*, *IEEE*

***Abstract*—Decentralized edge computing techniques have been attracted strongly attentions in many applications of intelligent internet of things (IIoT). Among these applications, intelligent edge surveillance (LEDS) techniques play a very important role to recognize object feature information automatically from surveillance video by virtue of edge computing together with image processing and computer vision. Traditional centralized surveillance techniques recognize objects at the cost of high latency, high cost and also require high occupied storage. In this paper, we propose a deep learning-based LEDS technique for a specific IIoT application. First, we introduce depthwise separable convolutional to build a lightweight neural network to reduce its computational cost. Second, we combine edge computing with cloud computing to reduce network traffic. Third, we apply the proposed LEDS technique into the practical construction site for the validation of a specific IIoT application. The detection speed of our proposed lightweight neural network reaches 16 frames per second in edge devices. After cloud server fine detection, the precision of the detection reaches 89%. In addition, the operating cost at the edge device is only one-tenth of that of the centralized server.**

***Index Terms*—Intelligent surveillance systems, target detection, deep learning, edge computing, cloud computing, lightweight neural network.**

I. Introduction

The fifth-generation (5G) wireless communication networks provide great convenience for supporting big data video processing platforms [1]. High-bandwidth, low-latency, and large-scale low-power networks make video surveillance more mature. Video devices based internet of things (IoT) has expanded from personal applications to large-scale scenarios [2], such as smart cities [3] and Industry 4.0 [4]. With the development of communication networks, video surveillance technology has been widely used. It plays an important role in many areas such as national security [5], social security [6], traffic monitoring [7], because it can be more intuitive to obtain information from surveillance video. However, the approach of the obtaining information is usually low efficiency. At present, most video surveillance systems get information by playing back the saved surveillance video after an abnormal situation occurs. This method cannot timely detect and prevent the occurrence of abnormal events or accidents. There are also some video surveillance systems that set up a monitoring room to concentrate dozens or even hundreds of surveillance video together, which is watched by full-time workers. Once an abnormal event occurs, the relevant workers will make alarm. This method can prevent the occurrence of abnormal events to a certain extent, but it is limited by the number of monitors, and often ignores some key behavior information. A more serious problem is that due to the reduced attention caused by visual fatigue, monitoring personnel cannot keep focusing on monitors. Real-time observation and analysis of massive video data pose a big challenge to conventional video surveillance systems, and even post-recording queries are extremely labor intensive and difficult to obtain all important information.

Fortunately, with the development of computer vision technologies, machine learning empowered computers have the ability to analyze video intelligently. In the early years, people searched the area to be detected by sliding window method, and then detected the target by hand-designed features. Haar-like feature [8] combined with adaptive boosting (Adaboost) [9] and cascade [10] is its typical approach. However, this method has a high time complexity and redundancy. In addition, the hand-designed features lack robustness thus they cannot adapt to many real-world scenarios. In 2006, G. E. Hilton, *et al.* [11] proposed the concept of deep learning (DL), which has been utilized in many applications such as big data analysis [12]–[14], computer vision [15], [16], and wireless communications [17]–[25]. Motivated by the ubiquitous applications, many state-of-the-art deep neural networks (DNN) have been proposed, such as convolutional neural networks (CNN) [26]–[28]. In 2015, K. He, *et al.* [29] proposed deep residual network (ResNet) to solve the problem of vanishing gradient and exploding gradient when the number of network layers was deepened. In order to improve the accuracy of DNN, the number of layers of the network is increasing, and the network structure is becoming more and more complicate, which makes DNN only run on centralized GPU or cloud servers. However, the centralized GPU processing method has the following disadvantages, such as high computational cost, high transmission cost, and high system delay, and hence it is hard to utilize in IoT applications. In order to solve this problem, many researchers focus on lightweight neural networks, which allows these intelligent methods to run on embedded devices or edge devices [30]–[34], and its development process is shown in Fig. 1.

At present, there are four mainstream lightweight neural networks. J. Luo, *et al.* [35] proposed a parameter pruning and weight sharing method to reduce the redundancy of DNN.

This work was supported by the Project Funded by the National Science and Technology Major Project of the Ministry of Science and Technology of China under Grant TC190A3WZ-2, National Natural Science Foundation of China under Grant 61671253, the Jiangsu Specially Appointed Professor under Grant RK002STP16001, the Innovation and Entrepreneurship of Jiangsu High-level Talent under Grant CZ0010617002, the Six Top Talents Program of Jiangsu under Grant XYDXX-010, the 1311 Talent Plan of Nanjing University of Posts and Telecommunications. *(Corresponding author: Guan Gui.)*

The authors are with the College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (E-mails: {1018010409, 1018010408,guiguan}@njupt.edu.cn).

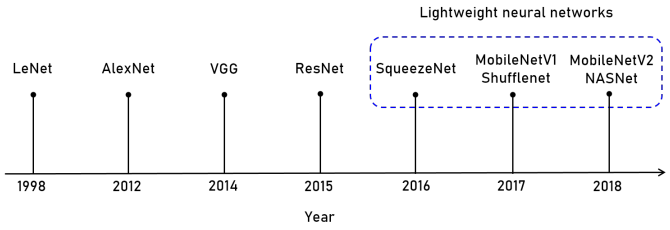


Fig. 1. The development process of neural networks.

W. Wen, *et al.* [36] proposed a low-rank factorization method to decompose the convolutional kernel in the original CNN. M. Sandler, *et al.* [33] proposed a compact convolutional computing unit method to compresses the storage of the model and reduces the computational complexity. J. Li, *et al.* [37] proposed a knowledge distillation method, which guides the training of small neural networks through large neural networks, to reduce the complexity of the network. Y. Wang *et al.* [38] proposed a light automatic modulation recognition method using DL and compressive sensing. All of above proposed light DL-based methods have made significantly contributions for realistic applications while there the related techniques are required to develop in the IoT applications.

In this paper, we propose a lightweight DL based intelligent edge surveillance (LEDS) method for intelligent internet of things (IIOT) applications. Based on the smart construction site system, this paper proposes a joint computing method that combine edge computing and central computing. Compared with the conventional DL methods, our proposed method has the following two advantages: **Less network resource occupancy:** Most of the existing intelligent surveillance systems directly process the video collected by the camera. Limited by the powerful computing capabilities required for DL, videos must be transmitted to a cloud server platform, which will consume a lot of network resources. The method of combine edge computing and cloud computing proposed in this paper reduces the occupation of network resources by sharing the computing burden to the edge notes. **Less system response delay**: Surveillance systems usually have hundreds of cameras. In addition to the occupation of network resources, cloud servers cannot process these videos in a timely manner, which will cause delays or even crashes to the system. Our proposed method can process video at the edge notes, and the cloud server is used for secondary confirmation and video storage, so that the system can achieve real-time response.

The rest of this paper is organized as follows. Section II introduces the smart construction site system, including system requirements, system principles, and system models. Section III provides the system operating environment, dataset production, and experimental results. Finally, we conclude this paper in Section IV.

II. System Model

In this section, we take smart construction site system as an example to introduce our LEDS. As shown in [16], smart construction site system needs to analyze the video captured by the camera to determine whether the worker wears a helmet. We have implemented this function in [16]. However, we have encountered many problems in practical applications. Since there are hundreds of cameras in the construction site, if the original central computing method is adopted, all the collected video data is transmitted to the cloud server for processing, which brings the following problems. First, in order to ensure the accuracy of neural network model detection, we need to transmit high-definition video, which takes up a lot of network resources. Second, transmitting so many videos can cause huge delay for the entire system. Third, even the cloud server cannot process the video from hundreds of cameras at the same time, causing the system to crash. Finally, transmitting the original video directly to the cloud server leaked users privacy. These are the motivations for us to study the new system. By adopting a combination of edge computing and cloud computing, we solve the above problems very well. The basic framework of the LEDS system is described in Fig. 2 and the functions of each part of the system are presented in Fig. 3.

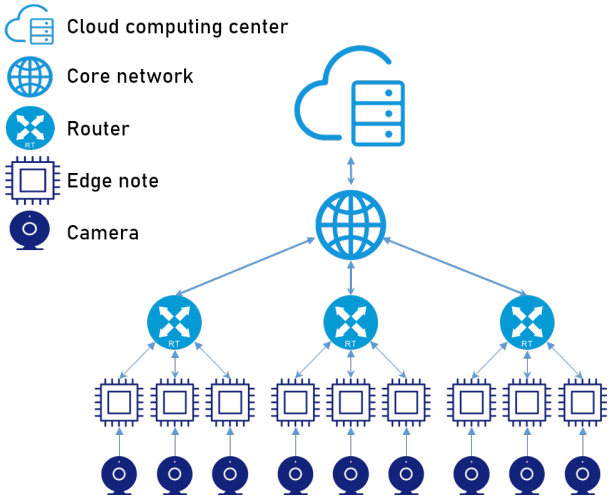


Fig. 2. Illustration of the basic framework of LEDS.

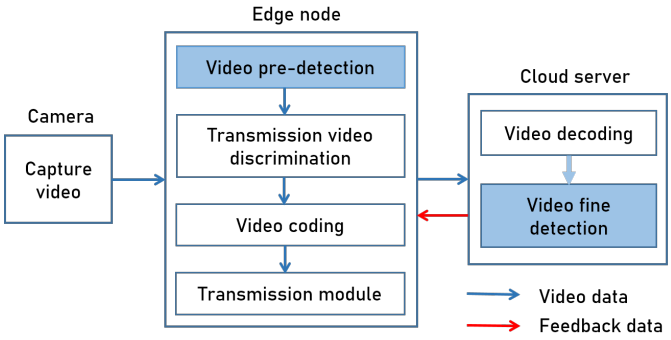


Fig. 3. Briefly introduction of the functions of each part of the LEDS, where three main parts are camera, edge node and cloud server.

A. System Framework

In this section, we introduce a framework of the system. The basic framework of the LEDS is shown in Fig. 2. It consists

of five components: camera, edge note, router, core network, and cloud computing center. According to its function, we can divide it into three parts, as shown in Fig. 3. First, the camera collects the video data of the construction site and transmits it to the edge node. At the edge node, we use a lightweight neural network to pre-detect the captured video. Based on the detection results, we only transmitted video clips containing un-wearing helmet workers to the cloud computing center. This not only can effectively save network resources, but also can effectively alleviate the computing burden of the cloud computing center. In order to occupy less network resources, we also encode the video at the edge node and select the user datagram protocol (UDP). On the cloud computing server side, we first decode the received video and then use the large neural network to fine-tune the video. Finally, the test result is passed back to the edge node. In the remainder of this section, we will describe in detail the construction of neural networks and the choice of transport protocols.

B. Lightweight Neural Network at the Edge Node

This section introduces the lightweight neural networks which are used in the smart construction site system. As we all know, it is difficult to make a fair comparison between the two detection networks. The quality of the detection neural network often depends on the actual requirement and it is a trade-off between accuracy and detection speed. In the smart construction site system, the speed of detection is paramount, so we choose the one-stage detection neural network. You only look once (YOLO) and single shot multiBox detector (SSD) are the most famous in the one-stage detection network, with fast detection speed and high detection accuracy. By simplifying their backbone network structure, this paper proposes Tiny-YOLO and MobileNetV2-SSD. The detailed introduction is as follows.

1) The structure of MobileNetV2-SSD: In image processing, the standard convolutional layer plays a crucial role in extracting image features. It is widely used in [26]–[28]. The structure of standard convolutional layer is shown in Fig. 4. Suppose the input of standard convolutional layer is a feature map of size $D_i \times D_i \times M$, and the output is a feature map of size $D_o \times D_o \times N$. The computational cost of the standard convolutional is

$$D_i \times D_i \times D_k \times D_k \times M \times N \tag{1}$$

where D_i is the size of the input feature map; M, N are the number of channels of the input and output feature map, respectively; D_k is the size of the convolutional kernels.

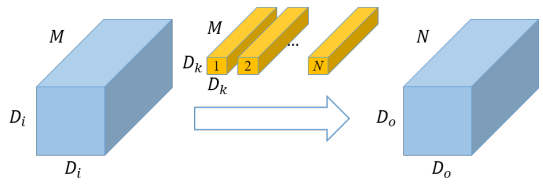


Fig. 4. The structure of standard convolutional layer with three parts: 1) Input: feature map of size $D_i \times D_i \times M$; 2) Convolutional kernels of size $D_k \times D_k \times M$; 3) Output: feature map of size $D_o \times D_o \times N$.

In order to reduce the computational cost and speed up the operation of the network, depthwise separable convolutional is proposed in [31]. The structure of depthwise separable convolutional layer is shown in Fig. 5. It integrates the standard convolutional into two steps: depthwise convolutional and pointwise convolutional.

The structure of the depthwise convolutional is shown in the left half of Fig. 5. A feature map of size $D_o \times D_o \times M$ can be obtained by depthwise convolutional. The computational cost of the depthwise convolutional is,

$$D_i \times D_i \times D_k \times D_k \times M \tag{2}$$

The structure of the pointwise convolutional is shown in the right half of Fig. 5. The computational cost of the pointwise convolutional is,

$$D_o \times D_o \times 1 \times 1 \times M \times N \tag{3}$$

The computational cost of the depthwise separable convolutional is,

$$D_i \times D_i \times D_k \times D_k \times M + D_o \times D_o \times 1 \times 1 \times M \times N \tag{4}$$

By comparing the computational cost of the standard convolutional and the depthwise separable convolutional, we can get the following ratio,

$$\begin{aligned} Ratio &= \frac{D_i \times D_i \times D_k \times D_k \times M + D_o \times D_o \times M \times N}{D_i \times D_i \times D_k \times D_k \times M \times N} \\ &\approx \frac{1}{N} + \frac{1}{D_k^2} \approx \frac{1}{D_k^2}, \end{aligned} \tag{5}$$

suppose that $D_i = D_o$, and N is much larger than D_K^2. The size of the convolutional kernel is usually 3×3, so the computational cost of the depthwise separable convolutional is one-ninth of the standard convolutional.

On the basis of MobileNetV1 [31], MobileNetV2 [33] further improves performance by introducing an Inverted residual block structure and a linear activation function. Due to the limitation of its own calculation, depthwise separable convolutional cannot determine the number of channels. If the number of output channels in the upper layer is small, depthwise separable convolutional can only extract features in low-dimensional space, which affects its performance. So MobileNetV2 adds a layer of pointwise convolutional before the depthwise convolutional, its purpose is to improve the dimension. In addition, MobileNetV2 changed the non-linear activation function of the second pointwise convolutional layer to a linear activation function. Because the non-linear activation function can effectively increase non-linearity in a high-dimensional space, but it can destroy features in a low-dimensional space, and its performance is not as good as a linear activation function. The structure comparison of standard convolutional, MobileNetV1, MobileNetV2 is shown in Fig. 6.

The network structure of MobileNetV2-SSD is shown in Fig. 7. Unlike SSD, MobileNetV2-SSD replaced VGG16 as the backbone network with MobileNetV2. Compared with other one-stage neural network, the biggest bright spot of SSD is that it uses shallow layers to detect small targets and deep

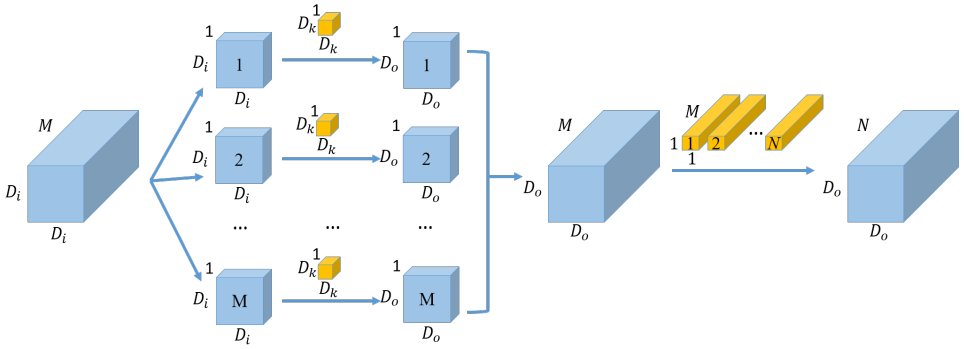


Fig. 5. The structure of depthwise separable convolutional layer with four parts: 1) Input: feature map of size $D_i \times D_i \times M$; 2) Depthwise convolutional layer with convolutional kernels of $D_k \times D_k \times 1$; 3) Pointwise convolutional layer with convolutional kernels of $1 \times 1 \times M$; 4) Output: feature map of size $D_o \times D_o \times N$.

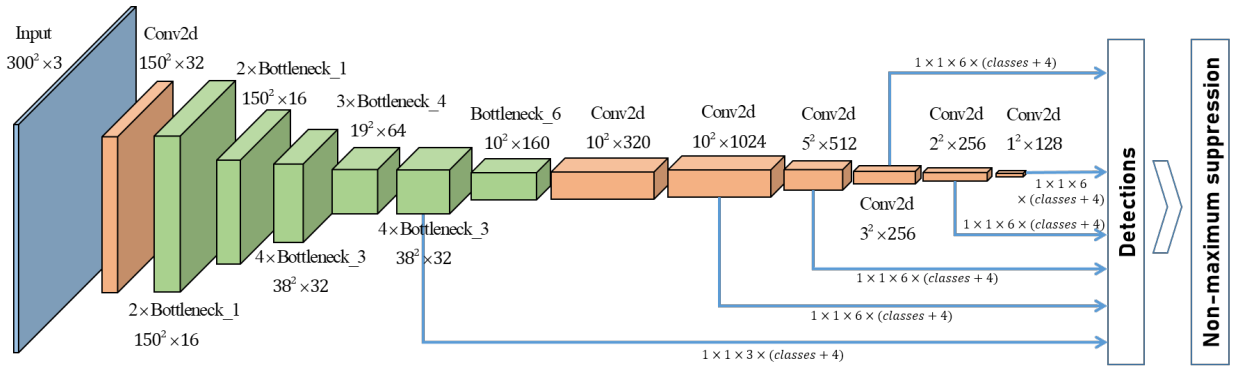


Fig. 7. The structure of MobileNetV2-SSD. The feature extraction network of MobileNetV2-SSD is composed of MobileNetV2, which composed of bottleneck layers. The additional convolutional layer is used to deepen the neural network to increase detection accuracy. MobileNetV2-SSD detects images from six different feature scales. In addition, it uses non-maximum suppression (NMS) to eliminate redundant prediction boxes.

TABLE I
THE STRUCTURE OF BOTTLENECK LAYER

Input	Operator	Output
$h \times w \times c$	1×1 Conv2d,ReLU6	$h \times w \times tc$
$h \times w \times tc$	3×3 Dwise,ReLU6	$\frac{h}{s} \times \frac{w}{s} \times tc$
$\frac{h}{s} \times \frac{w}{s} \times tc$	Linear 1×1 Conv2d	$\frac{h}{s} \times \frac{w}{s} \times tc'$

layers to detect large targets. Superficial neurons have more detailed information is more effective for small targets. Deep neurons have larger receptive fields and more abstract semantic information is more effective for large targets. SSD proposes to use both low-level feature maps and high-level feature maps for detection. As shown in Fig. 7, MobileNetV2-SSD detect on six different scales feature maps, thereby improving the accuracy of detection. In order to avoid using too low-level features, six layers of convolutional layers are added behind MobileNetV2. In addition, MobileNetV2-SSD uses the concept of anchor box in region proposal network (RPN) [39] to avoid using region proposals, which greatly reduces the amount of calculation.

Fig. 7 shows the structure of MobileNetV2-SSD, which consists of an input layer, 7 convolutional layers, and 16 bottleneck layers. The structure of each bottleneck layer is shown in Table I, where h and w are the size of the input feature map; c is the number of channels of the input feature map; c' is the number of channels of the output feature map; t is the magnification of the inverse residual (6 is selected here), and s is the stride of convolutional. The activation function of the first pointwise convolutional and depthwise convolutional is ReLU6, as shown in Eq. 6. The second pointwise convolutional uses a linear activation function,

$$f_{ReLU6}(x) = \min(\max(x, 0), 6) \in [0, 6] \tag{6}$$

Suppose that the dataset of MobileNetV2-SSD is represented as $T = \{g_i, class_i\}_{i=1}^{N}$, where g_i is the position parameter of ground truth, $class_i$ is the class of ground truth and N is the number of training samples for each picture. The loss function of MobileNetV2-SSD is given as follows

$$L_{MobileNetV2-SSD}(T; \Theta) = \frac{1}{N}(L_{loc}(T; \Theta) + \lambda L_{conf}(T; \Theta)) \tag{7}$$

where $L_{loc}(T; \Theta)$ is the localization loss of ground truth as shown in Eq. 8 ; $L_{conf}(T; \Theta)$ is the confidence loss of ground truth as shown in Eq. 10; Θ is the trainable neural network parameter and λ is used to adjust the importance of localization loss and confidence loss,

$$L_{loc}(T; \Theta) = \sum_{i=1}^{N} f_{smooth_{L1}}(g_i - \widehat{g}_i) \tag{8}$$

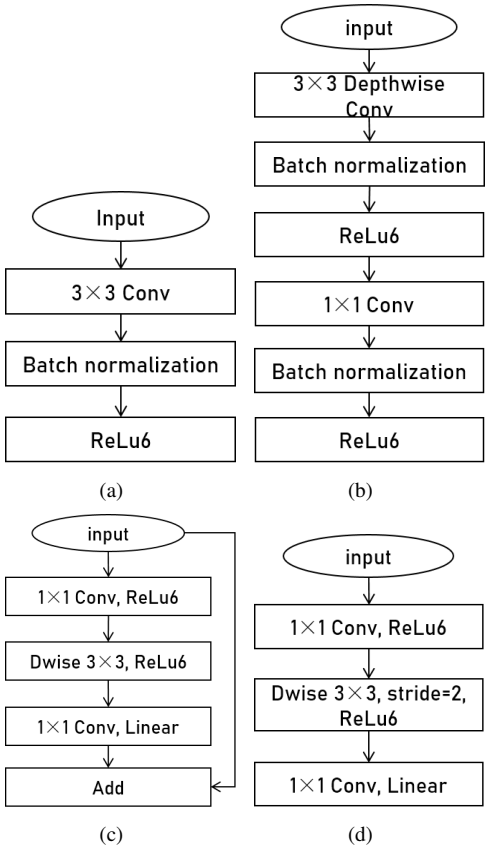


Fig. 6. (a) Standard convolutional layer with batch normalization and ReLU6; (b) Depthwise separable convolutional with batch normalization and ReLU6 for MobileNetV1; (c) Convolutional blocks for MobileNetV2 with stride = 1; (d) Convolutional blocks for MobileNetV2 with stride = 2.

$$f_{smooth_{L1}}(x) = \begin{cases} \frac{1}{2}x^2, & if\ |x| < 1 \\ |x| - \frac{1}{2}, & if\ |x| \geq 1 \end{cases} \tag{9}$$

where g_i is the position parameter of ground truth; $\widehat{g_i}$ is the prediction of the position parameter. We use $f_{smooth_{L1}}$ to calculate the loss between the ground truth and the prediction as

$$L_{conf}(T;\Theta) = -\sum_{i=1}^{N} log(f_{softmax}(c_j^i)) \tag{10}$$

$$f_{softmax}(x) = \frac{exp(x)}{\sum_{j=0}^{S} exp(x_j)} \tag{11}$$

where c_j^i is the class confidence that the i-th ground truth belongs to the j-th class; S is the number of class and c_0 represents the background. We use softmax function $f_{softmax}$ to calculate the class confidence, as shown in (11).

2) The structure of Tiny-YOLO: At the edge notes, we also tried another lightweight neural network called Tiny-YOLO, which network structure is shown in Fig. 8. There are 24 layers in Tiny-YOLO. Compared with 107 layers in YOLOV3, the number of layers is greatly reduced. Its network structure is simple, with a small amount of calculation, which is very suitable for running on the edge notes. Tiny-YOLO detects from two feature maps of different scale, which improves the detection accuracy of the network. It is worth noting that in the detection phase, it uses the convolutional layer to take the fully connected layer, which greatly reduces the calculation amount.

Algorithm 1 Algorithm description of the MobileNetV2-SSD.

Input: Annotated datasets for different construction site;
Output: The MobileNetV2-SSD for smart construction site system;
1: Data cleaning and data augmentation for datasets;
2: Randomly assign training dataset and validation dataset according to 7 : 3;
3: Build the botttleneck layer according to Table I for feature extraction;
4: Build the MobileNetV2-SSD structure according to Fig. 7 for multi-scale feature fusion;
5: Initialize the neural network and train the network weights;
6: Set a proper λ, calculate the loss according to (7) and minimize it;
7: Save weights and verify;
8: **return** MobileNetV2-SSD.

Type	Filters	Size	Output
Convolutional	16	3×3	416×416
Maxpool		2×2/2	208×208
Convolutional	32	3×3	208×208
Maxpool		2×2/2	104×104
Convolutional	64	3×3	104×104
Maxpool		2×2/2	52×52
Convolutional	128	3×3	52×52
Maxpool		2×2/2	26×26
Convolutional	256	3×3	26×26
Maxpool		2×2/2	13×13
Convolutional	512	3×3	13×13
Maxpool		2×2/1	13×13
Convolutional	1024	3×3	13×13
Convolutional	256	1×1	13×13
Convolutional	512	3×3	13×13
Convolutional	255	1×1	13×13
Avgpool		Global	
Connected		1000	
Softmax			

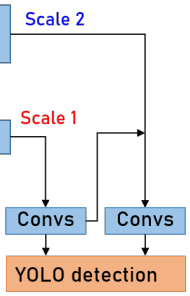


Fig. 8. The structure of Tiny-YOLO. The convolutional layers are used to extract features; The pooling layers are used to reduce the size of the feature map; The connected layer is used to classification; Softmax is used to calculate confidence. In addition, Tiny-YOLO detects images from two different feature scales.

Due to different detection principles, the loss functions of Tiny-YOLO and MobileNetV2-SSD are different. The loss function of Tiny-YOLO is given as follows.

$$\begin{aligned} L_{Tiny-YOLO}(T;\Theta) = L_{loc}(T;\Theta) \\ + L_{obj}(T;\Theta) + L_{conf}(T;\Theta) \end{aligned} \tag{12}$$

$$f_{SSE}(x) = \sum_{i=1}^{N}(x_i - \widehat{x_i})^2 \tag{13}$$

where $L_{loc}(T;\Theta)$ is the localization loss of ground truth as shown in (14) ; $L_{obj}(T;\Theta)$ is the object confidence loss of ground truth as shown in (15); $L_{conf}(T;\Theta)$ is the class confidence loss of ground truth as shown in (16); We use

sum of the squared errors function f_{SSE} to calculate the loss between the ground truth and the prediction, as shown in (13).

$$L_{loc}(T;\Theta) = \sum_{i=1}^{N}(g_i - \widehat{g_i})^2 \tag{14}$$

$$L_{obj}(T;\Theta)) = \sum_{i=1}^{N}(C_i - \widehat{C_i})^2 \tag{15}$$

$$L_{conf}(T;\Theta) = \sum_{j=0}^{S}(c_j - \widehat{c_j})^2 \tag{16}$$

where g_i is the position parameter of ground truth; C_i is the object confidence; c_j is the class confidence; $\widehat{g_i}$, $\widehat{C_i}$, $\widehat{c_i}$ are their prediction, respectively; S is the number of class and c_0 represents the background.

Algorithm 2 Algorithm description of the Tiny-YOLO.

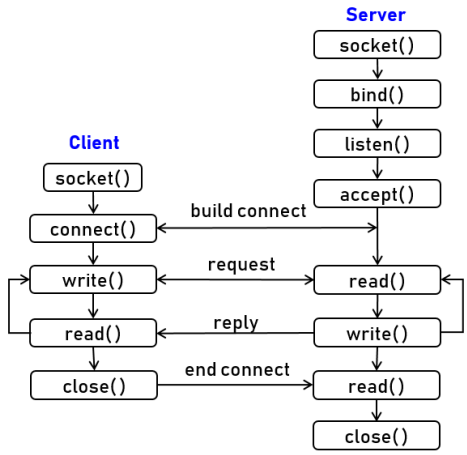
Input: Annotated datasets for construction site;
Output: The Tiny-YOLO for smart construction site system;
1: Data cleaning and data augmentation for datasets;
2: Datasets format conversion;
3: Randomly assign training dataset and validation dataset according to 7 : 3;
4: Build the Tiny-YOLO structure according to Fig. 8;
5: Initialize the neural network and train the network weights;
6: Calculate the loss according to Eq. 12 and minimize it;
7: Save weights and verify;
8: **return** Tiny-YOLO.

After the lightweight neural network completes the pre-detection, the edge node will determine which piece of video needs to be transmitted. The specific method is that the lightweight neural network only needs to detect three frames of pictures per second. If all three frames are detected that the worker is not wearing a helmet, the video in this period of time is transmitted. In order to further reduce the transmission burden of the network traffic, the video is also compressed and encoded before transmission.

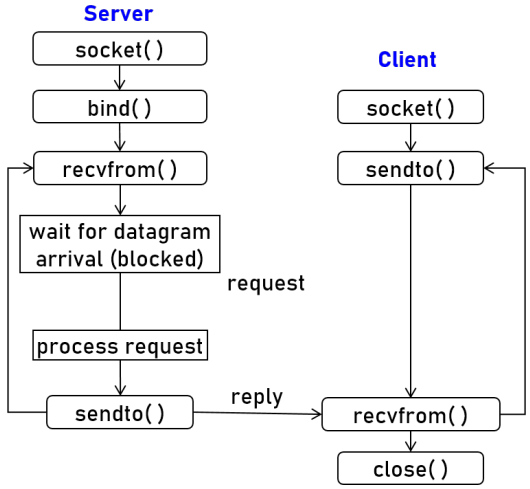
C. Transmission Protocol Selection

In this section, we introduce the choice of transmission protocol. As all we know, if you want to transfer information between different networks, you must abide by the transport layer protocol. The transport layer protocols mainly include two protocols, transmission control protocol (TCP) and user datagram protocol (UDP). Their respective flowcharts are shown in Fig. 9.

The Fig. 9(a) is the TCP flowchart. TCP provides connection-oriented services. It is a reliable transport layer communication protocol. To ensure its stability, TCP needs a three-way handshake to establish a connection and a four-way handshake to terminate the connection. In addition, it also has functions such as congestion control, timeout retransmission, and flow control. The Fig. 9(b) is the flowchart of UDP. UDP provides datagram-oriented services, which is an unreliable transport layer communication protocol. Therefore, it has the characteristics of simple structure, and does not perform any splitting and splicing operations on data messages. It always



(a) TCP



(b) UDP

Fig. 9. The flowchat of the transmission process of two different protocols: (a) TCP; (b) UDP.

sends data at a constant speed, which is very conducive to video transmission. Through the above analysis, UDP is selected for the smart construction site system.

D. Neural Network at the Server Side

On the server side, when the video transmitted from the edge side is received, the video is decompressed and decoded first. According to the requirements of smart construction site systems, a neural network with high detection accuracy needs to be selected. We chose YOLOV3 neural network, which network structure is shown in Fig. 10. The principle and network structure of YOLOV3 have been analyzed in detail in [16]. Its excellent detection accuracy and detection speed meet the system requirements.

III. Experimental Results

In this section, the experimental environment, experimental process, and experimental results are described. At the edge node, we chose NVIDIA Jetson TX2 as the edge device. At the server side, our calculations are accelerated by the

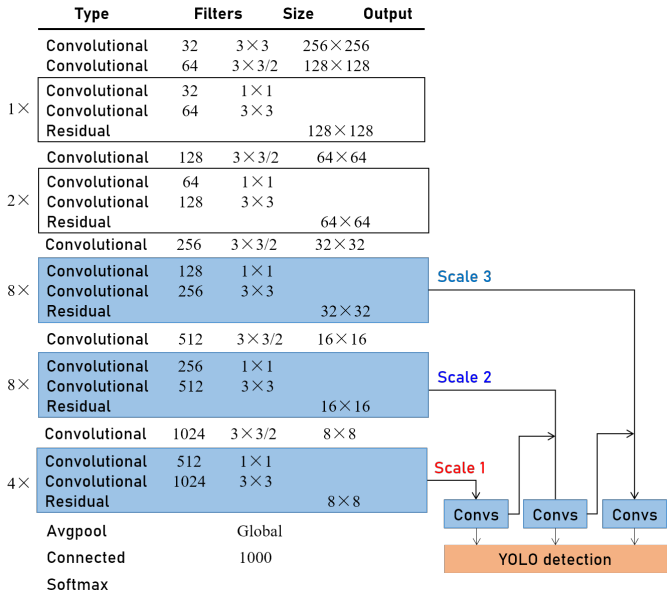


Fig. 10. The structure of YOLOV3. The feature extraction network of YOLOV3 is DarkNet53, which composed of residual modules. It can eliminate vanishing gradient and exploding gradient. YOLOV3 detects images from three different feature scales. In the detection phase, YOLOV3 uses the convolutional layers instead of the connected layer to reduce the amount of parameters.

NVIDIA GTX 1080Ti graphics card. Their specific parameters are shown in Table II. Both the server and edge operating systems are ubuntu16.04 and the programming language is Python.

TABLE II
THE EDGE NOTE AND SERVER SIDE DEVICE PARAMETERS

	Edge note	Server side
GPU	NVIDIA Pascal^{TM}, 256 CUDA cores	NVIDIA Pascal^{TM}, 3584 CUDA cores
CPU	HMP Dual Denver 2/2 MB L2 + Quad ARM A57/2 MB L2	Intel Xeon E5-2630
Memory	8 GB LPDDR4	128 GB LPDDR3
Data storage	32 GB	3 TB
Power	7.5 W	600 W

A. Dataset preparation

Due to the requirements of smart construction site systems, we need datasets of pedestrians and helmets, but existing public datasets do not include helmets. Therefore, we adopt the method of manual labeling, collecting site video through the camera and picking typical scenes. After data cleaning, we have adopted some data augmentation methods such as flip change, random crop, color jittering, shift transformation, scale transformation, contrast transformation, add gaussian noise and reflection transformation. Through the above processing, we have built a huge dataset of $100,000$ levels.

B. Experimental results

The model loss during MobileNetV2-SSD and Tiny-YOLO training is shown in Fig. 11. Due to the different definitions of the loss function, we cannot compare their accuracy directly by the loss value. But through the loss graph, we can find that the convergence speed of MobileNetV2-SSD is faster than Tiny-YOLO. When batchsize $= 512$, MobileNetV2-SSD has completed convergence after about 600 epochs, while Tiny-YOLO requires $1,000$ epochs. The detection speed and detection accuracy of YOLOV3, MobileNetV2-SSD and Tiny-YOLO are shown in Table III.

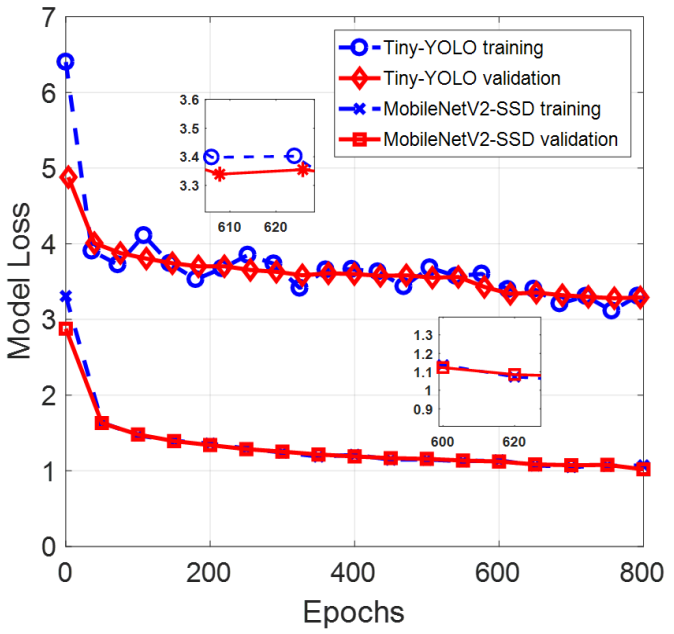


Fig. 11. The model loss during MobileNetV2-SSD and Tiny-YOLO training.

Fig. 12. Some detection results of smart construction site system.

TABLE III
THE DETECTION SPEED AND DETECTION ACCURACY

Operating equipment	NVIDIA GTX 1080Ti		NVIDIA Jetson TX2	
Neural network	YOLOV3	MobileNetV2-SSD	Tiny-YOLO	MobileNetV2-SSD
mAP	0.89	0.68	0.73	0.68
Frame rate(f/s)	18	83	12	16

IV. CONCLUSION

In this paper, we have proposed a lightweight DL based LEDS for the IIoT applications, which combine edge computing and cloud computing. This system has the advantages of low system delay and low network resource

occupancy. In addition, this method is ten times cheaper than the centralized method. We use the smart construction site system as an example to display how to implement the system. However, the experimental results show that there is still room for improvement in detection accuracy. In the future, we will combine federated learning (FL) [40] to solve the problem of weak system robustness and improve detection accuracy.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] Y. Chen, C. Lin, and J. Huang, "Energy efficient scheduling and management for large-scale services computing systems," *IEEE Trans. Serv. Comput.*, vol. 10, no. 2, pp. 217–230, 2017.

[3] L. Duan, Y. Lou, S. Wang, W. Gao and Y. Rui, "AI-oriented large-scale video management for smart city: Technologies, standards, and beyond," *IEEE MultiMedia*, vol. 26, no. 2, pp. 8–20, 2019.

[4] M. Wollschlaeger, T. Sauter and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[5] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive Survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[6] C. Stergiou, K. E. Psannis, A. P. Plageras, G. Kokkonis, and Y. Ishibashi, "Architecture for security monitoring in IoT environments," in *IEEE International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 1382–1385.

[7] A. Celesti, A. Galletta, L. Carnevale, M. Fazio, A. Lay-Ekuakille, and M. Villari, "An IoT cloud system for traffic monitoring and vehicular accidents prevention based on mobile sensor data processing," *IEEE Sens. J.*, vol. 18, no. 12, pp. 4795–4802, 2018.

[8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. 511–518.

[9] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[10] Q. Zhu, S. Avidan, M. C. Yeh, and K. T. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 1491–1498.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[12] B. Mao, Z. Md. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.

[13] G. Gui, F. Liu, J. Sun, J. Yang, Z. Zhou, and D. Zhao, "Flight delay prediction based on aviation big data and machine learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 1065–1069, 2020.

[14] B. Mao, F. Tang, Z. Md. Fadlullah, and N. Kato, "An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems," *IEEE Trans. Emerg. Topics Comput.*, in press, doi: 10.1109/TETC.2019.2899407.

[15] J. Pan, Y. Yin, J. Xiong, W. Luo, G. Gui, and H. Sari, "Deep learning-based unmanned surveillance systems for observing water levels," *IEEE Access*, vol. 6, no. 1, pp. 73561–73571, 2018.

[16] Y. Zhao, Q. Chen, W. Cao, J. Yang, and G. Gui, "Deep learning for risk detection and trajectory tracking at construction sites," *IEEE Access*, vol. 7, pp. 30905–30912, 2019.

[17] J. Zhang, X. Tao, H. Wu, N. Zhang, and X. Zhang, "Deep reinforcement learning for throughput improvement of uplink grant-free NOMA system," *IEEE Internet Things J.*, in press, doi: 10.1109/JIOT.2020.2972274

[18] H. Huang et al., "Deep learning for physical-layer 5G wireless techniques: Opportunities, challenges and solutions," *IEEE Wirel. Commun. Mag.* doi 10.1109/MWC.2019.1900027, pp. 1–9, 2019.

[19] G. Gui, H. Huang, Y. Song, and H. Sari, "Deep learning for an effective nonorthogonal multiple access scheme," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8440–8450, Sept. 2018.

[20] X. Liu, S. Wu, Y. Wang, N. Zhang, J. Jiao, and Q. Zhang, "Exploiting error-correction-CRC for polar SCL decoding: A deep learning based approach," *Trans. Cogn. Commun. Netw.*, 2019, doi: 10.1109/TCCN.2019.2946358

[21] H. Huang, Y. Peng, J. Yang, W. Xia, and G. Gui, "Fast beamforming design via deep learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 1065–1069, 2020.

[22] L. Ale, N. Zhang, H. Wu, D. Chen, and T. Han, "Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5520–5530, 2019.

[23] N. Kato, B. Mao, F. Tang, Y. Kawamoto, and J. Liu, "Ten challenges in advancing machine learning technologies towards 6G," *IEEE Wirel. Commun. Mag.*, in press, doi: 10.1109/MNET.001.1900476

[24] J. Sun, W. Shi, Z. Han, J. Yang, and G. Gui, "Behavioral modeling and linearization of wideband RF power amplifiers using BiLSTM networks for 5G wireless systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 10348–10356, 2019.

[25] J. Sun, J. Wang, L. Guo, J. Yang and G. Gui, "Adaptive deep learning aided digital predistorter considering dynamic envelope," *IEEE Trans. Veh. Technol.*, in press, doi: 10.1109/TVT.2020.2974506

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference of Learning Representation (ICLR)*, 2015, pp. 2058–2072.

[28] C. Szegedy et al., "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[30] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1–9.

[31] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1C9.

[32] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6848–6856.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[34] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8697–8710.

[35] J. H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5068–5076.

[36] W. Wen, C. Xu, C. Wu, Y. Wang, Y. Chen, and H. Li, "Coordinating filters for faster deep neural networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 658–666.

[37] J. Li, K. Fu, S. Zhao, and S. Ge, "Spatiotemporal knowledge distillation for efficient estimation of aerial video saliency," *IEEE Trans. Image Process.*, vol. 29, pp. 1902–1914, 2019.

[38] W. Wang, J. Yang, M. Liu, and G. Gui, "LightAMC: Lightweight automatic modulation classification using deep learning and compressive sensing," *IEEE Trans. Veh. Technol.*, in press, doi: 10.1109/TVT.2020.2971001

[39] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[40] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization spproach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, 2019.