Peer-to-Peer Networking and Applications (2022) 15:950–972
https://doi.org/10.1007/s12083-021-01258-4

100+ FPS detector of personal protective equipment for worker safety: A deep learning approach for green edge computing

Xiao Ke[1,2] · **Wenyao Chen**[1,2] · **Wenzhong Guo**[1,2]

Received: 2 August 2021 / Accepted: 23 October 2021 / Published online: 15 November 2021

Abstract
In industrial production, personal protective equipment (PPE) protects workers from accidental injuries. However, wearing PPE is not strictly enforced among workers due to all kinds of reasons. To enhance the monitoring of workers and thus avoid safety accidents, it is essential to design an automatic detection method for PPE. In this paper, we constructed a dataset called FZU-PPE for our study, which contains four types of PPE (helmet, safety vest, mask, and gloves). To reduce the model size and resource consumption, we propose a lightweight object detection method based on deep learning for superfast detection of whether workers are wearing PPE or not. We use two lightweight methods to optimize the network structure of the object detection algorithm to reduce the computational effort and parameters of the detection model by 32% and 25%, respectively, with minimal accuracy loss. We propose a channel pruning algorithm based on the BN layer scaling factor γ to further reduce the size of the detection model. Experiments show that the automatic detection of PPE using our lightweight object detection method takes only 9.5 ms to detect a single video frame and achieves a detection speed of 105 FPS. Our detection model has a minimum size of 1.82 MB and a model size compression rate of 86.7%, which can meet the strict requirements of memory occupation and computational resources for embedded and mobile devices. Our approach is a superfast detection method for green edge computing.

Keywords Superfast detection · Personal protective equipment · Model light-weighting · Green edge computing

1 Introduction

Industrial production plays a major role in the economic development of countries around the world, which covers many areas such as construction, manufacturing, and mining. In the construction industry, for example, US annual spending in 2019 was $1.3 trillion, or approximately 6.3% of GDP [1]. The construction industry has a huge demand for workers, with a total of 7.2 million employees in 2019, accounting for about 5% of the total labor force [2]. However, while industrial production is the backbone of the nation's economy, it is also one of the most dangerous sectors in which to work. According to the U.S. Bureau of Labor Statistics (BLS), 991 fatal accidents occurred in the U.S. construction industry in 2016, accounting for approximately 19% of all other industries [3]. Furthermore, 2017 data show that 79,810 accidents and non-fatal illnesses occurred in the construction industry during the year [3]. The main causes of fatalities in workplace accidents fall from heights, falling objects on the head, etc. In 2017, nearly 50% of construction workers in the U.S. construction industry died from falls and object impacts to the head [4].

The U.S. Occupational Safety and Health Administration (OSHA) require all workers in industrial manufacturing to wear personal protective equipment (PPE) to minimize the occurrence of safety incidents or to reduce injuries resulting from safety incidents [5]. A report by the National Institute

This article is part of the Topical Collection: *Special Issue on Green Edge Computing*

Guest Editors: Zhiyong Yu, Liming Chen, Sumi Helal, and Zhiwen Yu

✉ Wenyao Chen
yao1079543838@gmail.com

Xiao Ke
kex@fzu.edu.cn

Wenzhong Guo
guowenzhong@fzu.edu.cn

[1] Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing, College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China

[2] Key Laboratory of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou 350003, China

for Occupational Safety and Health (NIOSH) showed that between 2003 and 2010, there were 2,210 traumatic brain injuries (TBI) deaths across the United States, accounting for 25% of all deaths in the construction industry during that period [6]. The most common cause of TBI accidents in industrial processes is a worker falling from a height or an object falling on the worker's head [5, 6]. Wearing a helmet can minimize injuries to the head. Similarly, since heavy equipment such as trucks, bulldozers, and graders often work near workers, they may hit and injure workers at night or when visibility is low. To prevent similar accidents from occurring, workers must wear safety vests. Not only that, but gloves are also an important PPE, and wearing gloves while working can effectively protect workers from injuries to their hands. Wearing gloves can also prevent electrocution when performing electricity-related work. Since the end of 2019, the new coronavirus (COVID-2019) has spread worldwide, and workers also need to wear masks to avoid the spread of the virus. Studies have shown that most safety incidents can be avoided if workers wear proper PPE, such as helmets, safety vests, masks, and gloves [5]. From the perspective of practical efficiency and benefits, the manual supervision and inspection approach is inefficient and cannot adequately meet the practical needs of safety supervisors. In recent years, with the development of deep learning and computer vision, more and more researchers are using machine vision-based methods for object detection [7–11].

For the safety monitoring of workers wearing PPE, there have been many studies on the automatic detection of helmets, while there are fewer studies on the wearing of workers' masks or safety vests, and almost no studies on the detection of gloves. Currently, machine vision-based PPE wear detection is still challenging. First, the great variability in the background and worker state is caused by various field conditions, so studies in specific scenarios are difficult to extend to other scenarios. Second, small targets that are far from the camera are difficult to distinguish from cluttered backgrounds and other overlapping targets. Moreover, multiple targets may exist in the same image region, partially occluding each other, which makes the detection of PPE wear difficult. In addition, deep learning network is too large and computationally intensive to be used directly on surveillance cameras, drones, and other Internet edge devices, and most existing detection models are slow to detect and cannot meet real-time detection needs. Finally, until now there is no publicly available dataset containing multiple PPEs for evaluating PPE detection algorithms in various situations.

In this paper, to solve the problem of automated detection of PPE wear by workers in industrial production, we collect relevant images and build a dataset FZU-PPE containing a variety of PPE. Meanwhile, we introduced deep learning and convolutional neural network (CNN), which has the powerful advantage of strong feature learning capability, using neural networks to automatically extract features from the original data and synthesize low-level features into high-level features [12, 13]. Moreover, CNN has a more powerful performance in the field of computer vision than traditional image processing-based methods. For real-time detection of PPE in industrial production, to solve the problem of complex network models of object detection algorithms, after conducting research and analysis, we proposed two lightweight methods to improve the network structure of object detection algorithms, which greatly compressed the model size, parameters, and computational effort of object detection network. In addition, to further meet the strict requirements of memory occupation and computational resources for Internet edge devices, we borrow the idea of the pruning algorithm Network Slimming, which trains the Batch Normalization (BN) layer scaling factor γ in the original model to sparse the network structure, then prunes the detection model by using γ as a measure of the importance of the convolutional channels and finally performs fine-tuning training to recover the detection accuracy. Our proposed lightweight approach can effectively reduce the model computational effort and parameters, compress the model size, and help the detection model to be used on embedded and mobile devices. In addition, our lightweight detection model can perform ultra-fast detection of video or images with a detection speed of over 100 FPS.

The major contributions of this work are summarized as follows:

1. To improve the complex network model of the object detection algorithm, we halve the number of output channels of Conv and C3 modules in the head part of the network and use a 1×1 convolutional kernel instead of 3×3 convolutional kernels. These two optimization methods can simplify the network structure of the object detection algorithm and greatly reduce the model size, parameters, and computational effort.
2. In order to meet the memory consumption and computational resource requirements of Internet edge devices, we sparse the BN layer scaling factor γ of the detection model to compress γ from a Gaussian distribution to a state close to 0. Then we use channel pruning to reduce the model size, parameters, and computational effort so that the detection model can meet the memory consumption requirements of embedded and mobile devices and can perform superfast detection.
3. At present, most of the PPE datasets in industrial production are only helmets or safety vests, and there is no glove dataset. To solve this problem, this paper produces a PPE detection dataset FZU-PPE for industrial production, containing 18,767 images and nearly

97,536 instances related to various PPE such as helmets, safety vests, masks, and gloves. These images cover different scenarios of PPE wear, including images of complex scenarios such as cover-ups and small targets. Each instance in the dataset is labeled with a class label and its bounding box information.

2 Related works

Most current machine vision-based PPE wears detection methods focus only on identifying helmets. Conventionally, the methods of hardhat-wearing detection can be divided into two categories: sensor-based detection and vision-based detection. Sensor-based detection methods [14, 15] mainly use remote location and remote tracking technologies such as radio frequency identification (RFID) and wireless local area networks (WLANs) for detection. Zhang et al. [14] designed a helmet detection system based on an Internet of Things (IoT) architecture with an infrared beam detector and a thermal infrared sensor in the helmet, and used radio frequency identification (RFID) triggers to detect whether a worker is wearing a helmet. Dong et al. [15] developed a real-time location system (RTLS) for worker location tracking with a pressure sensor in the helmet to determine whether the helmet is worn or not by transmitting pressure information via Bluetooth. However, sensor-based detection methods are hardly satisfactory for identifying helmets at construction sites, and the use of sensors increases production costs.

In recent years, with the development of deep learning, vision-based techniques have received increasing attention. Zhang et al. [16] proposed an improved weighted bi-directional feature pyramid network (BiFPN) to fuse multi-scale semantic features for helmet detection with good results. Wang et al. [17] employed the MobileNet model as the backbone network, proposed a top-down module for enhanced feature extraction, and used a residual-block-based prediction module for the helmet detection for multi-scale features. Filatov et al. [18] designed an automatic helmet monitoring system for surveillance cameras based on MobileNet, which can meet the demand for real-time detection but there is still room for improvement in detection accuracy. Wu et al. [19] proposed a reverse progressive attention mechanism (RPA) to fuse features from different layers of different scales into a new feature pyramid and used the Single Shot Multibox Detector (SSD) framework to predict the detection results of safety helmets. Mneymneh et al. [20] detected each worker in the video and then determined whether any helmet was located in the top region of the worker detection frame. Wójcik et al. [21] used a novel helmet detection algorithm, which combined three techniques of deep learning, object detection and head key point localization to achieve better detection results. Fang et al. [22] employed a Faster R-CNN [23] based approach to automatically detect helmet wearing (non-hardhat-use, NHU) of construction workers, and they collected a total of 81,000 images from various construction sites as a training dataset to train the Faster R-CNN model, but the model could only detect workers who were not The model could only detect workers who were not wearing helmets. In addition, the Faster R-CNN relies heavily on the information extracted from the upper-level features and cannot fully utilize the underlying feature details, which may affect the detection results of target objects at different scales in the images.

Recently, relatively few studies have been conducted for the detection of masks, safety vests, and gloves. Seong et al. [24] used different approaches for the detection of safety vests using a combination of five color spaces (RGB, nRGB, HSV, Lab, and YCbCr) and six classifiers (ANN, C4.5, KNN, LR, NB, and SVM). Yu and Zhang [25] improved the YOLOv4 algorithm to achieve better results in mask detection. In Ref. [26], the authors state that the combination of ResNet50 and SVM can achieve face mask detection with an accuracy of 99.64%. However, the computational cost of the algorithm is quite expensive and not suitable for practical applications. In addition, the combination of SSD and MobileNetV2 for mask detection is proposed in Ref. [27], but its model structure is too complex and computationally large.

Good algorithms are pursuing both high accuracy and high speed. For example, in the field of trajectory clustering, Li et al. [28] presents a multi-step trajectory clustering method for robust AIS trajectory clustering. Compared with other algorithms, the multi-step trajectory clustering method has higher accuracy and lower time complexity. However, To date, there have been few studies related to industrial production safety inspection that use deep learning techniques to detect worker glove wear and the few studies that detect multiple PPEs simultaneously. Among the limited studies that detect multiple PPEs simultaneously, a commercial software called smartvid.io applies AI-driven algorithms to detect multiple PPE components (e.g., helmets, goggles, and steel-toed shoes) [29], but possesses good detection results only in simple scenarios. Ref. [30] proposed three methods based on YOLOv3 to detect whether workers are wearing helmets or safety vests correctly, and the best performing method mAP reached 72.3% and detection speed reached 11 FPS. Although most deep learning-based detection models have better detection accuracy, they also have the disadvantages of complex network structure and high computational effort. Most of the current research on personal protective equipment is pursuing higher detection accuracy at the expense of the equally critical detection speed. The ability to detect video or images better and faster is the only way to implement the relevant research into practical applications. To meet the memory and computational resource requirements of Internet edge devices, a lightweight object

detection model requires only a small amount of computational resources and memory space to run on mobile cloudlet platforms [31], embedded devices or mobile devices (e.g., smartphones, tablets) or even lightweight UAVs is necessary [32].

3 Methodologies

Industrial production areas are often complex environments with many objects such as workers, machinery and equipment, construction materials, etc. Images collected at construction sites using cameras can have many challenging issues such as scale variations, perspective distortion, and partial occlusions. Traditional detection methods using manual extraction of image features are usually ineffective for real-time detection and guarantee generalization to various complex scenes, while deep learning-based methods have the drawbacks of excessive model size and computational effort, although the detection speed and detection accuracy are better than traditional methods. To solve this problem, in this paper, we propose a lightweight object detection method that can greatly compress the detection model size, while improving the model detection speed for high-speed detection.

3.1 Proposed PPE wearing detection framework

In computer vision, the task of object detection is to identify the target object in an image and locate the target object position with a detection frame [33]. Two-stage object detection algorithms such as Fast R-CNN [34], Faster R-CNN [23], Mask R-CNN [35], and CPNDet [36] have better detection accuracy, but the detection speed is very slow. While one-stage algorithms such as SSD (Single Shot Detector) [37], YOLOv4 (You-Only-Look-Once) [38], CenterNet [39] and YOLOF (You-Only-Look-One-Feature) [40] improve the detection speed at the cost of reducing the detection accuracy. However, the model size of one-stage methods still cannot meet the requirements of edge devices. In this paper, we design a lightweight object detection algorithm for Internet edge devices based on the idea of YOLOv5 object detection algorithm and optimize the network structure using two improvement methods to obtain a lightweight object detection network.

3.1.1 Network structure

Our object detection network uses CSPNet (Cross Stage Partial Networks) [41] as the Backbone and PANet (Path Aggregation Network) [42] as the Neck. When feature extraction is performed on the image, the detection network divides the input feature map into two parts, one part is used as the input feature map of the next convolution module, and the other part is merged with the output feature map of another network layer, thus realizing cross-stage feature fusion, which effectively alleviates the gradient disappearance problem of the deep network model and the gradient information duplication problem of network optimization in the backbone. The cross-stage feature fusion in the detection network can reuse image features and reduce the parameters and FLOPS (Floating-point Operations Per Second) of the detection model.

The structure of our original detection network is shown in Fig. 1, which can be divided into two parts: backbone and head. The backbone is responsible for extracting image features, while the head processes the output feature map of the backbone to predict the coordinates and the class of the object. The standard convolutional layers (Conv) module consists of a Conv2d, a BN layer, and a SiLU activation function, while the C3 module contains three Conv and X Bottleneck modules. The C3 processing of the feature map can be divided into two parts: one part uses one standard convolutional layer, Conv, and multiple Bottleneck modules to process the input feature map, while the other part uses only one standard convolutional layer and finally merges the two output feature maps.

Moreover, we add the Focus module to Backbone. When the original image is input to the detection network, the Focus module is used to slice it to obtain multiple feature maps. That is, pixels are extracted from the high-resolution image and reconstructed into a low-resolution image. Compared with the direct convolution of the input image to obtain the feature maps, the Focus module can effectively reduce the original information loss of the input image and the FLOPS of the model, and also improve the model inference speed.

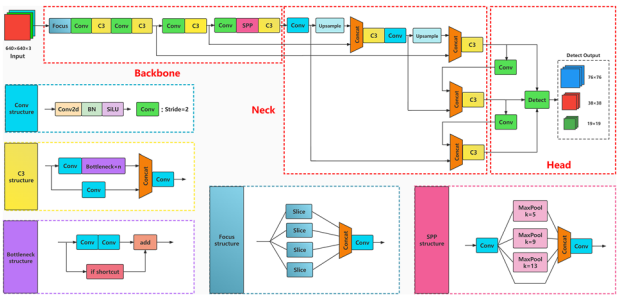


Fig. 1 The structure of the original detection network and some important modules. The input of the network is an image of size 640×640×3, the backbone part extracts features of the image, the neck part will integrate the features extracted from the backbone, form feature maps, and pass it to the head part. The head part makes predictions based on the feature maps, and finally generates bounding boxes and predict categories

3.1.2 Loss function

Our loss function consists of the loss of the center coordinates of the detection frame of the target object, the loss of the width and height coordinates, the loss of confidence, and the loss of classification, as shown in Eqs. (1), (2), (3), and (4), respectively. Coordinate loss, confidence loss and classification loss continuously update the network parameters by calculating the error between the model's predicted value of the target and the true value of the target, and the smaller the value of the loss function, the better the model is trained.

Our object detection network will divide the images into $S\times S$ grids in the prediction phase, and each grid generates B candidate frames, each candidate box contains 1 confidence value, 4 coordinate values, and C category probabilities. Where B is the number of anchor boxes in the output feature layer where each grid is located. And each candidate frame is processed by the network to get the corresponding prediction frame. Therefore, when the detection network detects the image, it will generate $S\times S\times B$ prediction frames. For a picture of size $S\times S$, the final output dimension is $S\times S\times B\times(4+1++C)$. In the prediction, the candidate frame with the largest Intersection over Union (IoU) with the ground truth of the target object in the grid is responsible for predicting the object.

$$loss_{center} = \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[(x_i-\hat{x}_i^j)^2+(y_i-\hat{y}_i^j)^2] \tag{1}$$

$$loss_{box} = \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[(\sqrt{w_i^j}-\sqrt{\hat{w}_i^j})^2+(\sqrt{h_i^j}-\sqrt{\hat{h}_i^j})^2] \tag{2}$$

$$\begin{aligned} loss_{conf} = &\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[\hat{C}_i^j\log(C_i^j)+(1-\hat{C}_i^j)\log(1-C_i^j)] \\ &+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}[\hat{C}_j^i\log(C_i^j)+(1-\hat{C}_i^j)\log(1-C_i^j)] \end{aligned} \tag{3}$$

$$loss_{class} = \sum_{i=0}^{S^2} I_{ij}^{obj}\sum_{c\in class}([\hat{P}_i^j\log(P_i^j)+(1-\hat{P}_i^j)\log(1-P_i^j)]) \tag{4}$$

For the sake of illustration, we refer to the region containing the target as the foreground and the region not containing the target as the background. In our loss function, only confidence loss is calculated for the predictor frame in the background, and the predictor frame in the foreground has classification loss and coordinate loss in addition to confidence loss. In the background, a threshold that ignores IoU thresh is set in the network, and the prediction frame in the background and the labeled frame ground truth calculate IoU one by one to get the maximum value max IoU. when max IoU > ignore IoU thresh, the confidence loss of the background prediction frame is ignored; when max IoU > ignore IoU thresh, the confidence loss of the background frame is added to the calculation of the loss function.

In the above equation, x, y, w, h are the predicted coordinates of the detection network of the target object, $\hat{x}, \hat{y}, \hat{w}, \hat{h}$ are the true coordinates of the detection frame for that object, $\lambda_{coord}, \lambda_{noobj}$ are the loss weights. The current model's ability to detect the coordinate information of the target object is judged by calculating the relative error between the predicted frame coordinates of the model and the true coordinates of the detected frame. The smaller the value of the coordinate loss function is, the better the model can detect the coordinate information of the target object. I_{ij}^{obj} and C_i^j indicate whether the j-th candidate frame of the i-th grid is responsible for predicting the current detected object, and are 1 if yes, and 0 otherwise. In contrast to I_{ij}^{obj}, I_{ij}^{noobj} is 1 when the candidate frame is not responsible for predicting the object and 0 otherwise. In Eq. (4), P_i^j denotes the classification probability, i.e. the probability that the target in the current prediction frame belongs to a certain category.

The division of foreground and background makes the training of the model more targeted. The predictions of coordinates, confidence and categories are obtained by means of regression, which allows the overall optimization of the network loss function until convergence. In summary, the loss function of our lightweight object detection algorithm is shown in Eq. (5).

$$\begin{aligned} Loss &= loss_{center}+loss_{box}-loss_{conf}-loss_{class} \\ &= \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[(x_i-\hat{x}_i^j)^2+(y_i-\hat{y}_i^j)^2] \\ &+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[(\sqrt{w_i^j}-\sqrt{\hat{w}_i^j})^2+(\sqrt{h_i^j}-\sqrt{\hat{h}_i^j})^2] \\ &-\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[\hat{C}_i^j\log(C_i^j)+(1-\hat{C}_i^0)\log(1-C_i^j)] \\ &-\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}[\hat{C}_j^i\log(C_i^j)+(1-\hat{C}_i^j)\log(1-C_i^j)] \\ &-\sum_{i=0}^{S^2} I_{ij}^{obj}\sum_{c\in class}([\hat{P}_i^j\log(P_i^j)+(1-\hat{P}_i^j)\log(1-P_i^j)]) \end{aligned} \tag{5}$$

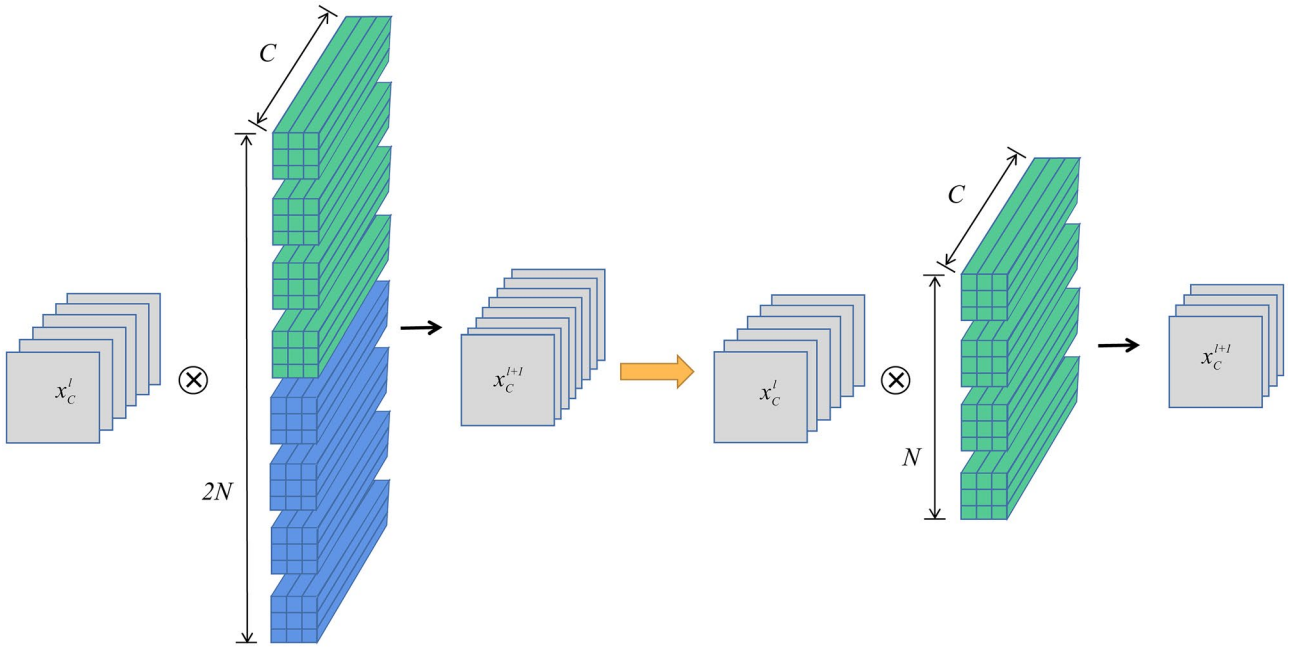


Fig. 2 Halve the number of output channels of Conv and C3 in the head part and neck part. With the reduced number of output channels, the number of parameters and the amount of computation of the model will be reduced accordingly

3.1.3 Two optimization method for network structure

After extensive research and analysis, we concluded that the head part of the detection network has some redundancy, so we decided to optimize for the head part and streamline the network structure. In this regard, we propose two optimization methods to simplify the detection network, reduce the model size, parameters, and computational effort, and obtain a lightweight object detection network, so that the detection model can meet the requirements of Internet edge devices in terms of model size and computational resource consumption.

The first improvement is to halve the number of output channels of Head's Conv and C3 in this paper. The C3 module uses two different ways to extract features from the input feature map and finally merges the two intermediate feature maps in the output feature map with strong feature extraction capability. When the input image enters the detection network, the backbone part is responsible for abstracting the underlying features into higher-level features, and the neck part is responsible for integrating and upsampling the incoming feature maps from the backbone part into the head part, whose main role is to make predictions based on the feature maps. Due to the excellent feature extraction capability of the backbone part, it is difficult to achieve a more perfect result by continuing to extract features from the output feature maps of the backbone part in the neck part and the head part. Therefore, we decided to reduce the number of convolutional kernels of Conv and C3 in the neck part and head part to half of the original one. The reduction in model size and computational effort comes at the cost of weakening the feature extraction capability for features of lower importance. The improvement method is shown in Fig. 2.

PeleeNet [43] uses 1×1 convolution kernels in the head part to predict the class confidence and detection box offset of the detected objects. Experiments show that the prediction using 1×1 convolution kernels is almost as effective as using 3×3 convolution kernels, while the computational effort is reduced by 21.5%. Inspired by PeleeNet, our second improvement is to modify the convolution kernel of the head part of Conv from 3×3 to 1×1. The amount of parameters of the Conv kernel is reduced to 1/9 of the original one, while the prediction ability of the head part is almost no weaker. The improvement method is shown in Fig. 3.

3.2 Channel pruning algorithm based on scaling factor

A paper by Liu et al. [44] published in ICCV proposed a pruning algorithm called Network Slimming, which prunes neural networks in a simple but quite effective way. For VGGNet, Network Slimming gives a $20\times$ reduction in model size and a $5\times$ reduction in computing operations, with no significant accuracy degradation in the

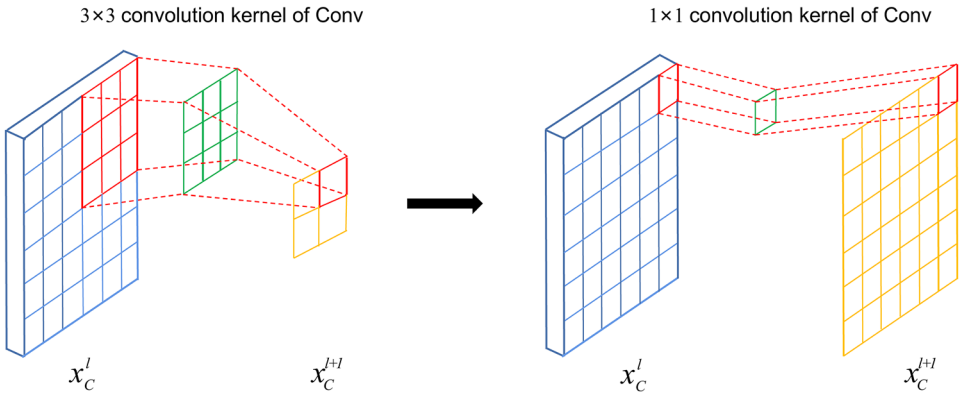


Fig. 3 3×3 convolution and 1×1 convolution

pruned network model. Drawing on the idea of Network Slimming, we design a pruning algorithm for lightweight object detection networks, which prunes the convolutional channels in the detection network and can be directly used in convolutional neural network-based object detection networks. The model obtained after pruning does not require the use of specialized hardware or underlying libraries and can be directly used for fast detection tasks.

In a study related to model pruning, Ref. [45] suggests pruning the unimportant connections after the neural network training is completed. At this point, most of the weights in the network are 0, so the model size can be reduced by storing the model in the form of a sparse matrix. However, this approach can only speed up model inference with a dedicated sparse matrix operation library or hardware and has a very limited reduction in running memory. Ref. [46] achieves high compression rates by imposing sparse constraints on each weight with additional gate variables and by pruning the joins with zero gate values. This approach achieves a better compression rate than [45], but again requires a dedicated sparse matrix operation library/or hardware to accelerate model inference. Recently, Li et al. [47] pruned the convolutional kernel channels with smaller weight values to achieve a reduced model size after the model training was completed. The study [48] spars the network by randomly suppressing the channel connections in the convolutional layers before training, but the accuracy of the models generated by this approach is not satisfactory.

The training granularity of sparse training is divided into weight level, channel level, and network layer level. As shown in Fig. 4. Sparse training at a fine granularity (e.g., weight level) has the highest compression rate and flexibility of the model, but usually requires specialized hardware or underlying libraries to accelerate the inference model. The Sparse granularity for the network layer is the coarsest; this granularity of sparsity does not require special hardware or underlying libraries but is less compressible and flexible for the model. In addition, sparse training for network layers is only fully effective when the depth of the network model exceeds 50 layers. In contrast, channel-level sparse training strikes a good balance between flexibility and ease of implementation, and it can be applied to any convolutional neural network or fully connected network. For these reasons, our pruning algorithm will perform channel-level sparse training on the model.

The flow of our proposed pruning algorithm is shown in Algorithm 1 and Fig. 5. We perform channel-level sparsification of the Initial network obtained from normal training. In the sparse training, we choose to apply a simple L1 regularization on the channel scaling factor γ of Batch Normalization to sparse the network at channel granularity, which achieves a good compression rate without the need for specialized hardware or underlying libraries. For sparsification training, we train both the network weights and the scaling factor γ and apply sparse regularization to the scaling factor γ. After the sparse training is completed, the convolutional channels in the model with smaller scaling factor γ will be pruned. Finally, we will fine-tune the pruned model to restore accuracy. The objective function of sparse training is as follows:

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma) \tag{6}$$

where (x, y) denotes the training inputs and outputs, W denotes the trainable weights, the first accumulation term is the normal training loss of the convolutional neural network, g (γ) is the sparsity penalty on the scaling factor, and λ is the balancing factor. We choose the L1 parameterization as the penalty term, as shown in Eq. (7).

$$g(\gamma) = |\gamma| \tag{7}$$

Batch Normalization is a very common optimization in convolutional neural networks, which generally acts before the activation layer, and it enables the network to converge

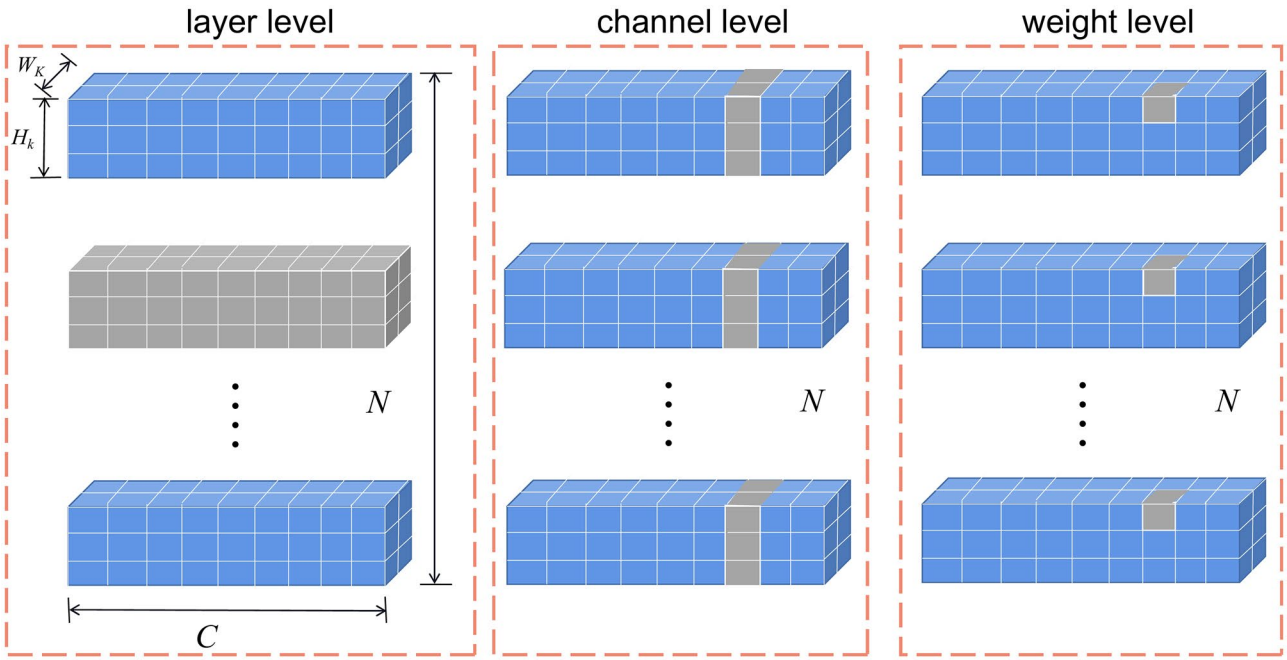


Fig. 4 The visualization of different types of pruning. The gray parts represent pruning granularity

quickly and increases the generalization performance. The formula is as follows:

$$\hat{z} = \frac{z_{in}\mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}; z_{out} = \gamma\hat{z} + \beta \tag{8}$$

where B denotes the current mini-batch, z_{in} and z_{out} are the inputs and outputs of the BN layer, μ_B and δ_B are the mean and standard deviation of B, and γ and β are the trainable affine transform parameters.

We use the scaling factor γ of the BN layer as a measure of model pruning, and prune the convolutional channels

Algorithm 1 Channel pruning algorithm based on scaling factor

Input: The initial Network obtained by normal training

Output: Compact network with small model size and low computational effort

Initialize Epochs=200, bn_weights = $\gamma_1, \gamma_2, \gamma_3, \ldots \gamma_n$

// start sparse training

while epoch ≤ Epochs do

  g(γ) ←|γ|

  objective function $L \leftarrow \sum_{(x,y)} l(f(x,W),y) + \lambda \sum_{\gamma\in\Gamma} g(\gamma)$

  output of bn $z_{out} \leftarrow \gamma\hat{z} + \beta$

  update $\gamma_1, \gamma_2, \gamma_3, \ldots \gamma_n$

  bn_weights ← $\gamma_1, \gamma_2, \gamma_3, \ldots \gamma_n$

  end

  Initialize prune_ratio=0.9

  sort channels by $\gamma_1, \gamma_2, \gamma_3, \ldots \gamma_n$

  if $c_1, c_2, c_3, \ldots, c_i$ channel_id < len(channels) × prune_ratio

    prune channel $c_1, c_2, c_3, \ldots, c_i$

  end

  fine tune pruned network

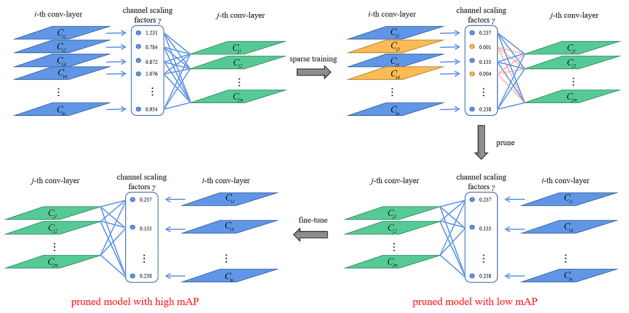


Fig. 5 The flow of the channel pruning algorithm. After sparse training, the convolutional channels with lower γ (yellow in the figure) will be pruned. Since the network structure of the pruned model has changed compared with the original model, but the neural network parameters learned according to the original network structure have not changed, the detection ability of the pruned model for objects is weakened and the mAP of the pruned model is low. The pruned model is fine-tuned for training, and the model can relearn the neural network parameters based on the current network structure to recover the detection accuracy of the model and improve the mAP

whose γ is lower than the pruning threshold to reduce the model size, and finally fine-tune the pruned model to recover the accuracy to obtain a lightweight detection network.

4 Experiment

4.1 PPE detection dataset

After a lot of research and searching, we found that most of the current relevant datasets on PPE detection only contain one category of helmet or safety vest, and the datasets that contain multiple PPEs at the same time are quite rare. Therefore, we established the PPE detection dataset FZU-PPE, which includes four types of PPEs: helmet, mask, safety vest, and glove, covering different scenarios of PPE wear, and also includes examples of complex situations such as cover-up and small target. In addition to the four PPEs, we also added the annotation of three objects, namely fire extinguisher, flame, and grounding rod, to the dataset in order to further improve the usefulness and practical value of the FZU-PPE dataset. The model obtained by training with the FZU-PPE dataset can detect not only different kinds of PPEs but also fire extinguishers, flames and grounding rods to improve safety on construction sites. After screening and data cleaning, there are 18,767 images in the FZU-PPE dataset, which are obtained by shooting at the construction site and searching the web using keywords. We divided the FZU-PPE into a training set and test set, where the training set contains 13,334 images and the test set contains 5,433 images. 97,536 instances of FZU-PPE are included in 11 categories, and each instance is labeled with category labels and bounding boxes. The number of instances per category and some images are shown in Table 1 and Fig. 6.

4.2 Evaluation metrics

In this paper, the detection performance of the object detection model is evaluated using the basic metrics of Precision, Recall, Average Precision (AP), mean Average Precision (mAP), and detection speed. The Precision and Recall are calculated as follows.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

where TP denotes the number of positive samples judged correctly, FP denotes the number of positive samples judged incorrectly, and FN denotes the number of negative samples judged incorrectly among all detected samples.

The calculation of AP is defined as the integral of the recall rate for each category with upper and lower limits of 1 and 0, respectively, and is calculated as Eq. (11).

$$AP = \int_0^1 p(r)dr \tag{11}$$

$$mAP = \frac{\sum_{i=0}^{k} AP(i)}{k} \tag{12}$$

Table 1 The number of instances in the FZU-PPE dataset

Label	Number		
	Trainval	Test	Total
Helmet	20850	8329	29179
No_helmet	11804	4989	16793
Face_mask	4800	2016	6816
No_face_mask	1829	557	2386
Safety_vest	5801	2290	8091
No_safety_vest	11423	4917	16340
Gloves	1876	726	2602
No_gloves	8066	3911	11977
Extinguisher	1284	509	1793
Fire	576	232	808
Groundrod	519	232	751

Fig. 6 Dataset image example, including small targets, dense crowds and partial occlusion, etc

mAP is the mean value of *AP* values for all categories, which is calculated as shown in Eq. (12), where k denotes the number of all detection categories in the dataset, and *AP(i)* denotes the *AP* value of category *i*.

The evaluation index of detection speed uses FPS in frames/s, which indicates the number of images that the model can detect per unit time, and is used to determine whether the object detection model can meet the requirements of real-time detection.

4.3 Model hyperparameters

In the paper, the experimental running environment is Ubuntu 16.04 and the GPU is RTX 2080 Ti. Our experimental steps can be roughly divided into four steps: normal training of the model, sparse training, model pruning, and fine-tuning training to recover the detection accuracy of the model. The hyperparameters we used in the four steps are shown in Table 2.

Table 2 The hyperparameters used in the four steps of our experiment

Hyperparameter	Meaning	Normal training	Sparse training	Model pruning	Fine-tuning
lr0	initial learning rate	0.01	0.01	-	0.032
lrf	final OneCycleLR learning rate	0.2	0.2	-	0.12
momentum	SGD momentum/Adam beta1	0.937	0.937	-	0.843
weight_decay	optimizer weight decay	0.0005	0.0005	-	0.00036
warmup_epochs	warmup epochs	3.0	3.0	-	2.0
warmup_momentum	warmup initial momentum	0.8	0.8	-	0.5
warmup_bias_lr	warmup initial bias lr	0.1	0.1	-	0.05
fliplr	image flip left-right	0.5	0.5	-	0.5
mosaic	image mosaic	1.0	1.0	-	1.0
mixup	image mixup	0.5	0.5	-	0.243
s	scale sparse rate	-	0.0007	-	-
global_percent	global channel prune percent	-	-	0.9	-
layer_keep	channel keep percent per layer	-	-	0.6	-

The hyperparameter lr0 is the initial learning rate for training. We use the OneCycleLR method to vary the learning rate. Instead of monotonically decreasing the learning rate during training, we let the learning rate vary back and forth between a set maximum and minimum value. Throughout the training process, the learning rate first increases from the initial value to the maximum value, and then decreases from the maximum value to a size below the initial value, with the final OneCycleLR learning rate being lr0*lrf.

Fliplr, mosaic and mixup are hyperparameters about data enhancement, for example, during normal training, we set fliplr to 0.5, which means 50% of the training set images will be flipped left and right during training. When performing sparse training, the value of the sparse rate is crucial. Too large a sparse rate can seriously degrade the accuracy of the model, while too small a sparse rate can affect the pruning effect of the model. The sparse rate when performing sparse training varies with different data sets. After extensive experiments, we obtained a suitable sparse rate for the FZU-PPE dataset, which is 0.0007.

When pruning, we use two key hyperparameters: global_percent and layer_keep. Hyperparameter global_percent is the ratio of the number of channels pruned to the total number of channels in the model. In order to prevent all the channels of some network layers from being deleted, we add the hyperparameter layer_keep. When all the channels of a certain network layer will be deleted, we will keep some channels of the network layer, and the hyperparameter layer_keep is the ratio of the number of reserved channels to the total number of channels in the network layer. In our experiments, we set the hyperparameter global_percent to 0.9, and 90% of the convolutional channels will be removed during pruning. Meanwhile, we set the values of hyperparameter layer_keep to 0.3, 0.4, 0.5 and 0.6 to obtain four pruning models, YOLOE-P3, YOLOE-P4, YOLOE-P5 and YOLOE-P6, respectively.

4.4 Results and comparisons

4.4.1 Results of PPE wearing detection framework

In Sect. 3.1, we design a lightweight object detection algorithm for edge devices by borrowing ideas from the YOLOv5 object detection algorithm and optimize the network using two improvement methods to obtain a lightweight object detection network. We reduce the number of output channels of Conv and C3 in the head part of the detection network to half of the original one, which greatly reduces the parameters and computational effort of the model. At the same time, inspired by PeleeNet, we change the convolutional kernel size, of the last two Conv modules in the head part from 3×3 to 1×1. We call the initial network PPENet and add the improved method to PPENet for ablation experiments, and the experimental results are shown in Table 3.

As can be seen in Table 3, both of our improvement methods for PPENet achieve better results. The first improved method is to reduce the number of convolutional kernels

Table 3 Results of ablation experiments using different improved methods

Method	mAP	Model Size	Parameter	GFLOPS
PPENet	87.1	13.7 MB	7,080,880	16.4
PPENet + Half	86.1	9.75 MB	4,990,832	12.5
PPENet + 1×1	87.0	12.5 MB	6,425,520	15.6
PPENet + Half + 1×1	85.8	9.43 MB	4,826,992	12.2

of Conv and C3 in the Head section to half of the original one. This method reduces the model size and the parameters to 12.5 MB and 4,990,832 at the cost of a 1% reduction in mAP, which is 71% and 75% of the original size, respectively. The second improvement is to change the size of the convolutional kernels of the last two Conv modules in the head part from 3×3 to 1×1. This method reduces the mAP by only 0.1%, while the model size is reduced from 13.7 MB to 12.5 MB, 91% of the original size, and the parameters and computational effort are reduced by 9% and 5%, respectively, resulting in a better light-weighting result at very small cost. After combining the two methods, the mAP is reduced by 1.3%, and the model size is reduced from 13.7 MB to 9.43 MB.

It can be seen from Table 3 that the second improved method has the least impact on mAP, reducing the model size and the parameter amount by nearly 10% while the detection accuracy is almost unchanged. In contrast, the effect of the first improved method is more obvious, the model size, parameter amount, and computational effort are reduced by nearly 30%, and the impact on mAP is also greater than that of the first improved method, from 87.1% to 86.1%, a decrease of 1%. However, considering the significance of the lightweight effect, such a decline is acceptable. Figure 7 shows the comparison of various indicators between the original model PPENet and the improved model. The three metrics of model size, parameters, and GFLOPS (Gigaflops Per Second, computational effort of model) are benchmarked against the data from PPENet, and the data from the other methods are percentages relative to PPENet. Figure 7 visually shows the changes in various data of the improved model. It can be clearly seen that the two improved methods we proposed have almost no effect on mAP, while the values of the three indicators of model size, parameter amount, and GFLOPS are significantly reduced. Figure 8 shows the comparison between the AP value of each category of the PPENet + Half + 1×1 model and the original PPENet model. The AP value of most categories has decreased by 1% to 2%.

For the PPENet network model, the two improved methods proposed in this paper can effectively reduce the model size, parameters, and the GFLOPS. The model size and parameters of PPENet + Half + 1×1 are 68% of the PPENet model and the GFLOPS is 74%. We call the improved PPENet + Half + 1×1 network YOLOE.

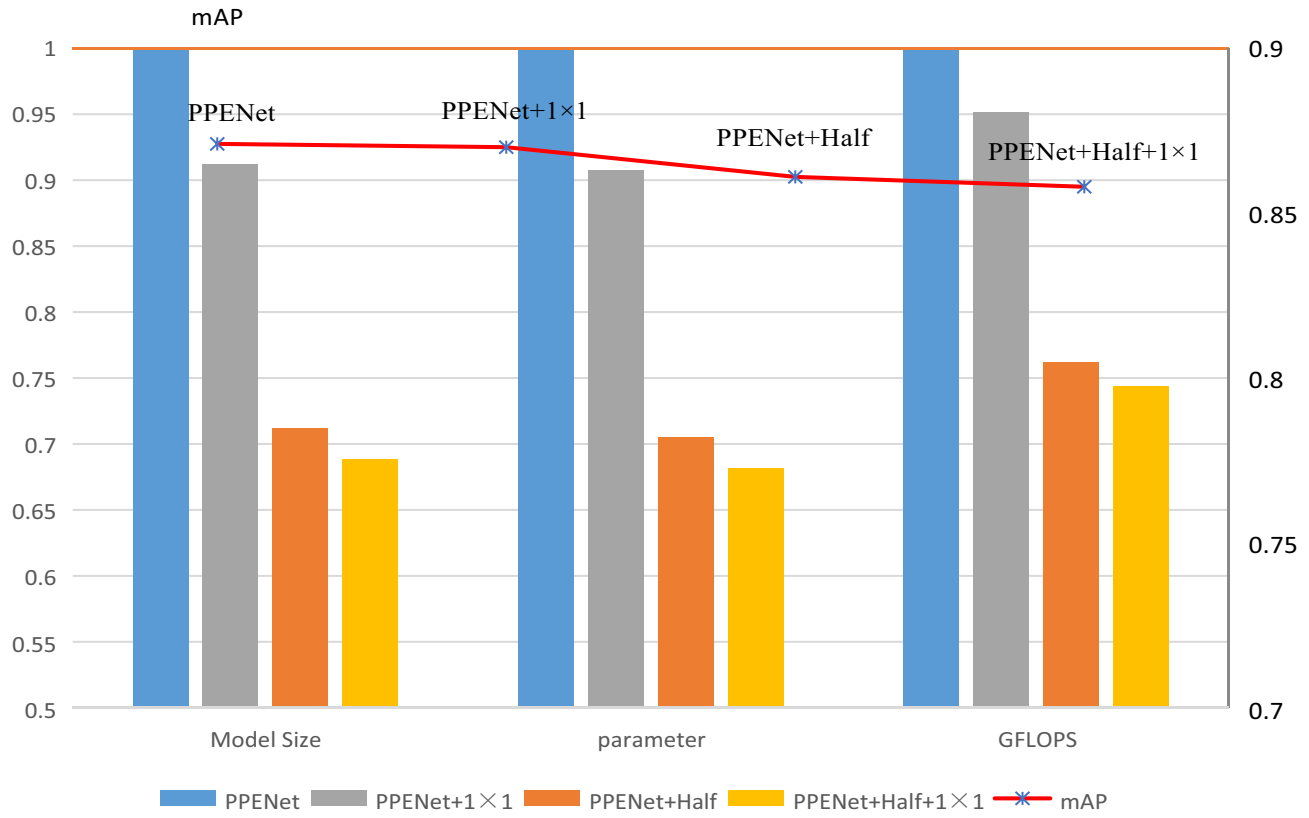


Fig. 7 Comparison of the results of the two improvement methods

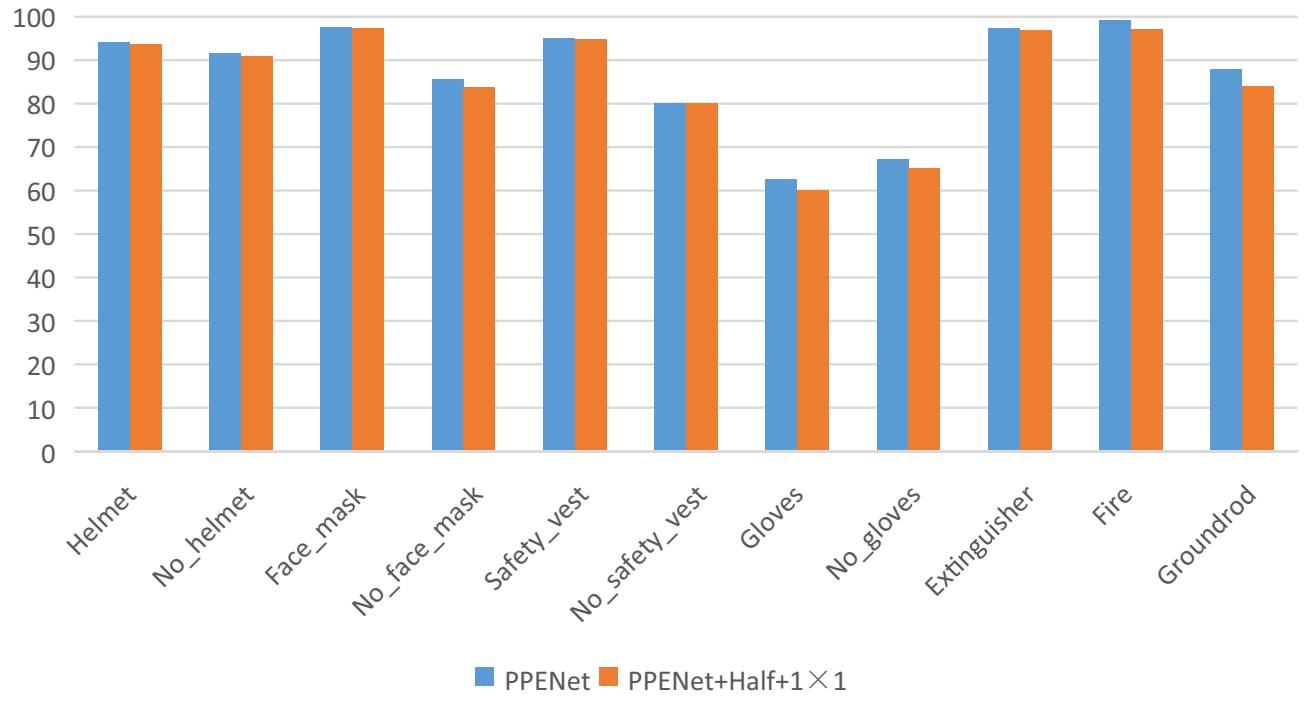


Fig. 8 Comparison of the AP of PPENet and PPENet + Half + 1 × 1 for each category

4.4.2 Results of sparse training

The YOLOE model obtained in Sect. 4.4.1 is pruned using the Network Slimming-based pruning algorithm. The sparsification training is performed first, and the sparsification scale factor is obtained while training the network, which is used as a measure for the network pruning to trim the convolutional channels. The sparse training requires setting the sparse rate. If the sparse rate is set too large, it will speed up the process of model sparse, and the compression of the model is higher after sparse training, but at the same time, it will lead to a significant decrease in the detection accuracy of the model after the sparse training is completed and cannot be recovered. On the contrary, if a smaller sparse rate is set, the sparse process is slower, but the accuracy of the model decreases less after the sparse training. The sparse process is a game process, we don't only want a high compression degree, but also want to recover enough accuracy after sparse, and the final sparse results are different when setting different sparse rates, and it often takes a high time cost to find a suitable sparse rate. After repeated experiments and tests, we finally set the sparse rate to 0.0007, and the change of mAP when the model is tested on the validation set during sparse training is shown in Fig. 9, and the change of BN layer scaling factor γ during normal training and sparse training is shown in Figs. 10 and 11, respectively.

According to Fig. 10, the scaling factor γ of the BN layer converges from a scattered, irregular distribution to a Gaussian distribution centered at 1 as the training epoch increases during the normal training of the model. From Figs. 9 and 11, we can see that the accuracy of the model decreases, and the value of γ is continuously compressed in the first 20 epochs after the sparse training starts. By the 20th epoch of sparse training, most of the γ values are compressed to close to 0, and the mAP of the model for the validation set drops to about 70% at this time. After the 20th epoch, the mAP of the model on the validation set starts to recover gradually. By the 110th epoch of sparse training, the mAP converges to 84%, and the sparse training is completed, by which time γ has been fully compressed, and the accuracy of the compressed model returns to the normal level. The value of the scaling factor γ is close to 0, which means that the importance of the convolution channel to the whole model is very low, so the channel can be directly cut off during pruning and will not have a large impact on the accuracy of the model.

The BN layer scaling factor γ obtained after sparse training will be used as a measure to prune the model. The algorithm will rank the individual convolution channels

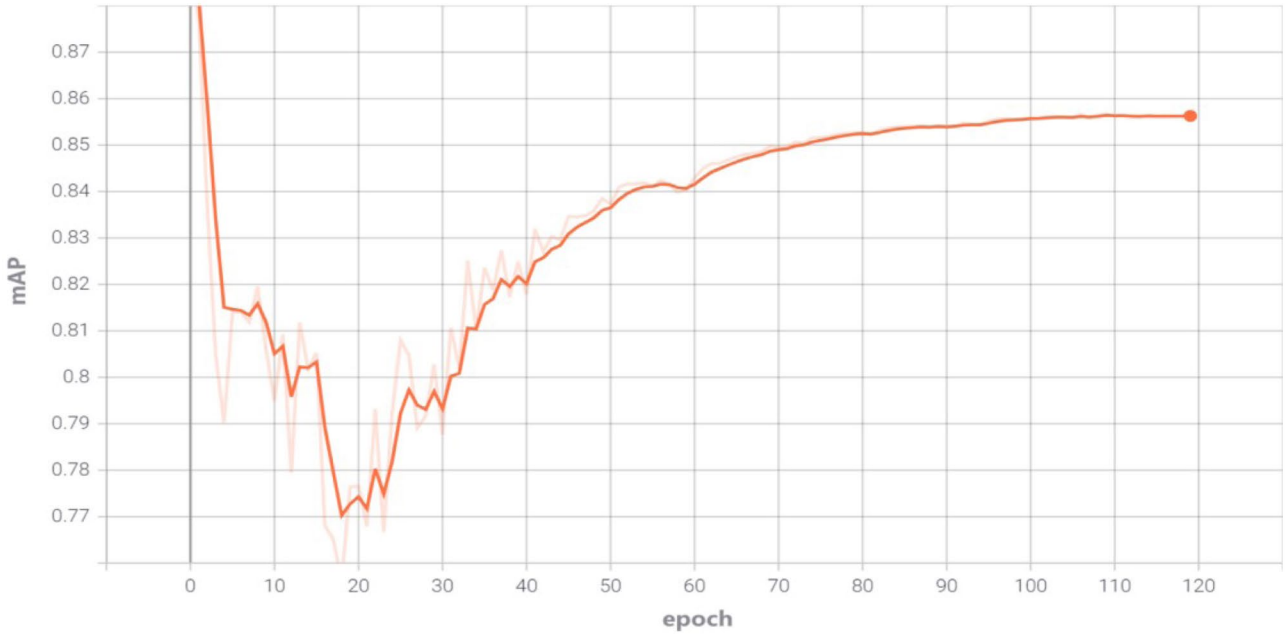


Fig. 9 Variation of mAP of the model for the validation set during sparse training

according to the magnitude of γ and then determine the number of convolution channels to be removed from the entire network model based on the set pruning rate. If the BN layer scaling factor γ of all convolutional channels in a particular convolutional layer is very low, it will result in the deletion of the entire convolutional layer. Compared with deleting some of the convolutional channels in a convolutional layer, deleting the whole convolutional layer may have a larger impact on the model accuracy. Therefore, in this experiment, to prevent the entire layer of a convolutional layer from being deleted, we set the layer_keep hyperparameter to protect the convolutional layer. When all channels of a convolutional layer need to be deleted, we keep some channels with the highest importance according

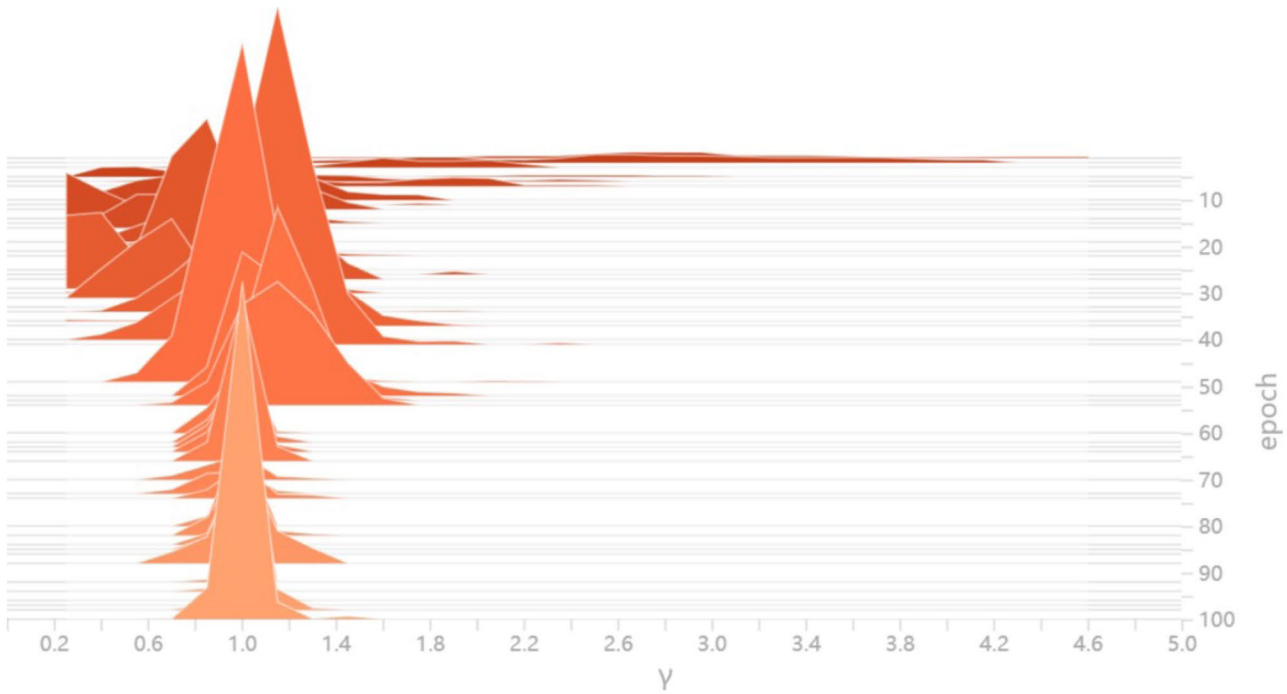


Fig. 10 Variation of scaling factor γ with the epoch during normal training. The vertical coordinate represents the epoch and the horizontal coordinate represents the value γ

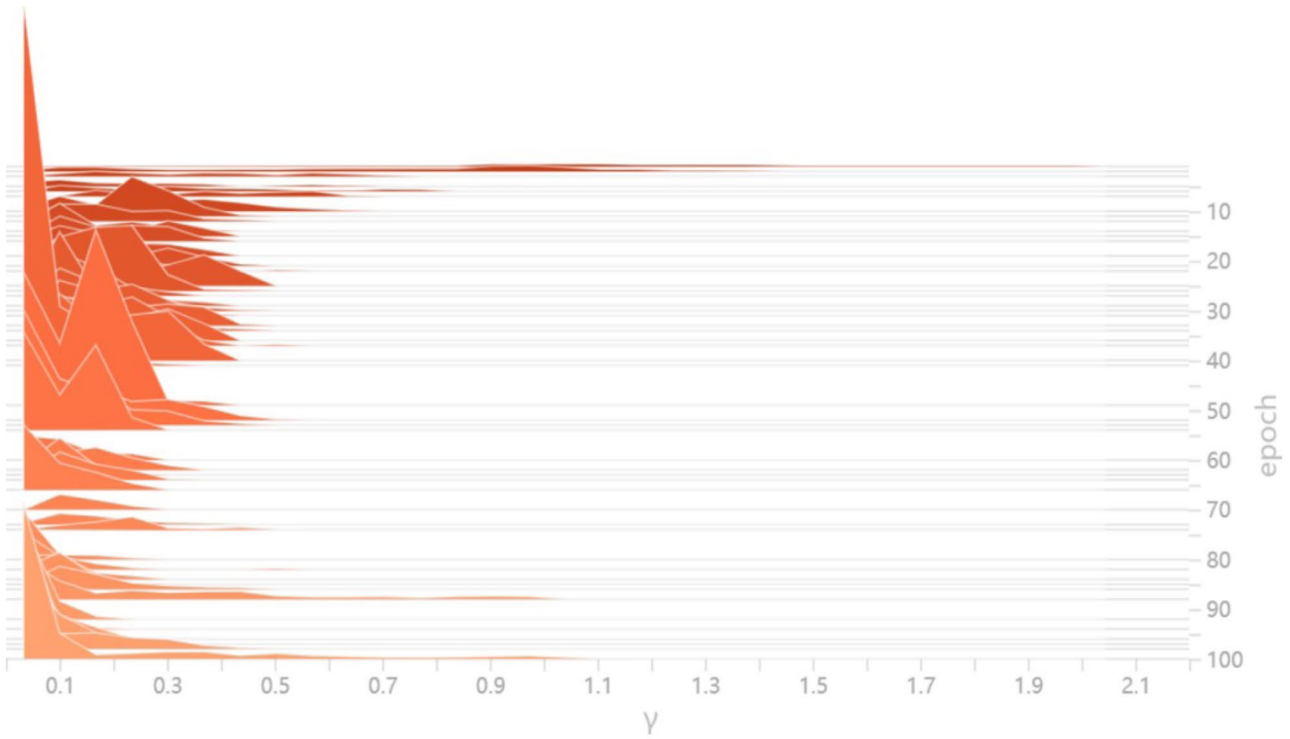


Fig. 11 Variation of scaling factor γ with epoch during sparse training

to the layer_keep hyperparameter. This prevents the whole convolutional layer from being deleted by the pruning algorithm.

4.4.3 Results of channel pruning algorithm based on scaling factor

When pruning the sparsely trained model, we set the pruning rate to 0.9, which means that 90% of the convolutional channels in the whole model will be pruned when pruning. In addition, for a pruning rate of 0. 9, we set four different hyperparameter layer_keep thresholds for the number of channels per layer, 0.3, 0.4, 0.5, and 0.6, representing that if all convolutional channels in a convolutional layer are to be removed, then 30%, 40%, 50%, or 60% of the convolutional channels in each convolutional layer will be retained. Finally, four different sizes of detection models YOLOE-P3, YOLOE-P4, YOLOE-P5, and YOLOE-P6 are obtained. The pruned models need to be fine-tuned to recover the detection accuracy, and the data of the four models after fine-tuning are shown in Table 4.

As can be seen from Table 4, when the pruning rate is 0.9, if the reservation threshold of the number of channels per layer is 0.3, the model size, parameters and GFLOPS of YOLOE-P3 are the smallest, 1.82 MB, 850,144, and 4.3, respectively, which are 81%, 82%, and 65% less than the model YOLOE before pruning, and the pruning effect is quite remarkable. Such a model can fully meet the requirements of various embedded or mobile devices that are very sensitive to memory consumption. Of course, the higher the pruning degree is, the more the accuracy of the model is affected after pruning. With the dramatic increase of model lightness, the mAP decreases from 85.8% to 80.0%, a reduction of 5.8%. If the number of channels per layer retention threshold is increased, the model size, parameters, and GFLOPS of YOLOE-P4 are reduced by 73%, 74%, and 57%, respectively, when the threshold is 0.4, and the mAP value decreases by 5.1%. At a threshold of 0.5, the model size, parameters, and GFLOPS for YOLOE-P5 are reduced by 65%, 67%, and 53%, respectively, and the mAP value decreases by 3.2%. At a threshold of 0.6, the YOLOE-P6 model size, parameters, and calculation amount were reduced by 53%, 54%, and 42%, respectively, and the mAP value decreased by 1.6%.

The comparison of each datum of PPENet, YOLOE and the four pruning models is shown in Fig. 12, and each data

Table 4 Comparison of the results of the pruned model and the original model

Model	mAP	Model Size	Parameter	GFLOPS
PPENet	87.1	13.7 MB	7,080,880	16.4
YOLOE	85.8	9.43 MB	4,826,992	12.2
YOLOE-P6	84.2	4.47 MB	2,234,736	7.1
YOLOE-P5	82.6	3.28 MB	1,610,756	5.7
YOLOE-P4	80.7	2.59 MB	1,250,200	5.2
YOLOE-P3	80.0	1.82 MB	850,144	4.3

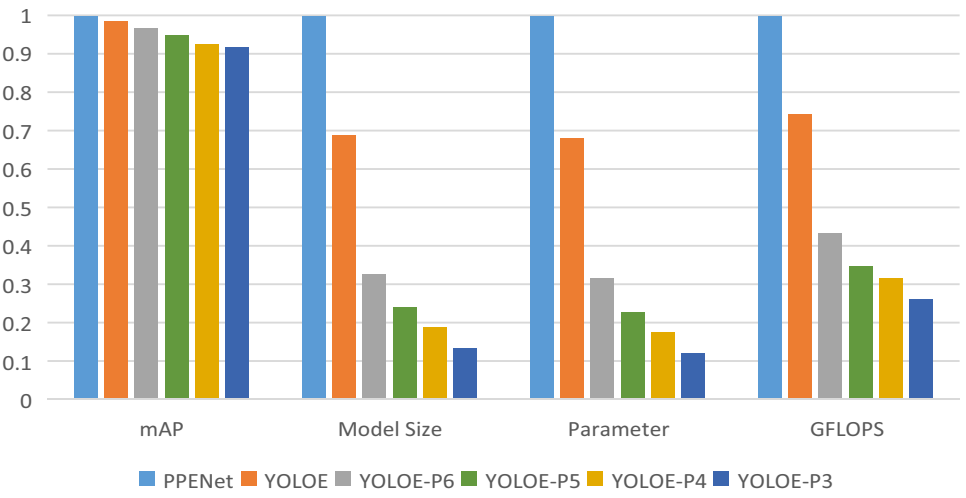


Fig. 12 Visualization of the pruned model compared to the original model

of PPENet is 100% for the four types of metrics. Among the four pruning models, YOLOE-P3 has the highest degree of pruning, but also the highest decrease in accuracy, while YOLOE-P6 has the largest model size but only a 1.6% decrease in detection accuracy. Therefore, if the device is very sensitive to memory occupation, you can choose to use YOLOE-P3 for PPE real-time detection in your own device, and if you have higher requirements for detection accuracy, you can choose YOLOE-P6. YOLOE-P4 and YOLOE-P5, on the other hand, strike a better balance between model size and detection accuracy, and we consider them to be the best models among the four models.

4.4.4 Comparisons

To verify the effectiveness of the object detection algorithm for green edge computing proposed in this paper, YOLOv3, YOLOv4, Scaled-YOLOv4 [49], CenterNet and YOLOX [50], the most popular object detection algorithms, are used to compare with our proposed YOLOE-P6, and the results are shown in Table 5. In terms of model size, both YOLOv4 and YOLOv3 have a model size of over 230 MB, which is nearly 50 times larger than that of YOLOE-P6. Scaled-YOLOv4, CenterNet and YOLOX have a model size of 100 MB, 77 MB and 68 MB, respectively, which are much smaller than YOLOv3 and YOLOv4. However, Scaled-YOLOv4, CenterNet and YOLOX still cannot meet the storage space requirements of the edge devices, which are 22 times, 17 times and 15 times larger than the model size of YOLOE-P6, respectively. It can be seen that the YOLOE-P6 model has a great advantage in model size after network improvement and pruning, which can fully meet the demanding requirements of embedded and mobile devices for deep learning model memory consumption. In terms of detection accuracy and detection

Table 5 Comparison of the results of our lightweight detection model with several of the most popular detection models

Method	mAP	Model Size	FPS
YOLOv3	79.1	235 MB	38
YOLOv4	83.3	244 MB	35
Scaled-YOLOv4	86.3	100 MB	43
CenterNet	82.4	77 MB	36
YOLOX	81.9	68 MB	69
YOLOE-P6	84.2	4.47 MB	105

Table 6 The number of instances in the PASCAL VOC dataset

Label	Number		
	Trainval	Test	Total
Aeroplane	1333	311	1644
Bicycle	1255	389	1644
Bird	1870	576	2446
Boat	1457	393	1850
Bottle	2195	657	2852
Bus	957	254	1211
Car	4136	1541	5677
Cat	1666	370	2036
Chair	4488	1374	5862
Cow	1127	329	1456
Diningtable	1110	299	1409
Dog	2136	530	2666
Dorse	1209	395	1604
Motorbike	1191	369	1560
Person	22,848	5227	28,075
Pottedplant	1827	592	2419
Sheep	1437	311	1748
Sofa	1266	396	1662
Train	1032	302	1334
Tvmonitor	1260	361	1621

Fig. 13 Example of some images from the PASCAL VOC dataset

speed, Scaled-YOLOv4 has the highest detection accuracy of 86.3% mAP, but the detection speed is only 43 FPS, much slower than YOLOE-P6. Among the five models we have listed for comparison, YOLOX has the fastest detection speed of 68 FPS. However, in terms of both speed and accuracy, YOLOX is far inferior to YOLOE-P6. With an mAP of 84.2% and FPS of 105, YOLOE-P6 has a detection accuracy just below Scaled-YOLOv4, while the detection speed is far faster than the remaining five comparison models, allowing for ultra-fast object detection.

To validate the generalization of our designed lightweight object detection algorithm for green edge computing, we also conducted experiments on two open datasets, PASCAL VOC, GDUT-HWD, and a private dataset, FZU-CND. FZU-CND is a small dataset for bank card number recognition with 136 images, 109 images in the training set, and 27 images in the test set.

The PASCAL VOC dataset, a public dataset introduced by the PASCAL VOC Challenge, contains 20 different target classes such as person, cat, bus, etc. It is one of the most used public datasets in the field of target detection today. In the target detection task, almost all SOTA methods need to use the PASCAL VOC dataset to prove the accuracy and effectiveness of their models. In our experiments, the PASCAL VOC dataset contains 22,077 images, of which 17,125 images are used in the training and validation sets and 4952 images are used in the test set. The number of instances and some images for each category are shown in Table 6 and Fig. 13.

GDUT-HWD is a public dataset for helmet detecting proposed by Wu et al. in 2019. Wu et al. collected a large number of images about helmet wearing to form

Table 7 The number of instances in the GDUT-HWD dataset

Label	Number		
	Trainval	Test	Total
Blue	1234	1378	2612
Yellow	1953	1945	3898
White	1963	1818	3781
Red	2301	1930	4231
None	2123	2248	4371

Fig. 14 Example of some images from the GDUT-HWD dataset

the GDUT-HWD dataset and classified helmets into five categories based on different colors: red, blue, white, yellow, and non-helmet. GDUT-HWD contains a total of 3174 images, of which 1588 images are used as the training and validation set and another 1586 images are used as the test set. The number of instances and some images for each category are shown in Table 7 and Fig. 14.

The experimental results of the three datasets are shown in Table 8. In GDUT-HWD and FZU-CND, the model size of YOLOE-P2 is 1.16 MB and 1.66 MB, respectively, compared to PPENet, which is 92% and 88% reduction respectively, while the mAP is almost unchanged. In the PASCAL VOC, the model size of YOLOE is 70% of the initial model PPENet, and the mAP

Table 8 Comparison of the results of pruned model and original model on different datasets

	Model	GDUT-HWD	FZU-CND	PASCAL VOC
mAP	PPENet	88.7	99.5	82.0
	YOLOE	88.5	99.5	80.6
	YOLOE-P2	86.3	99.5	71.8
Mode Size	PPENet	13.7 MB	13.7 MB	13.8 MB
	YOLOE	9.46 MB	9.45 MB	9.46 MB
	YOLOE-P2	1.16 MB	1.66 MB	1.79 MB
Parameter	PPENet	7,064,698	7,078,183	7,105,153
	YOLOE	4,818,874	4,825,639	4,839,169
	YOLOE-P2	507,378	764,587	834,461
GFLOPS	PPENet	16.4	16.4	16.5
	YOLOE	12.2	12.2	12.3
	YOLOE-P2	3.3	3.8	4.5

Fig. 15 Pruning model results for the four PPEs, both worn and unworn

is reduced by only 1.4%. The experiments demonstrate that our lightweight target detection algorithm works on different datasets.

We find that on the difficult datasets PASCAL VOC and FZU-PPE, which contain various complex scenes and a large number of images, the mAP decreases sharply when the model size is compressed to 90%. In contrast, for the simpler datasets GDUT-HWD and FZU-CND with fewer images, our lightweight object detection algorithm can compress the model size to 90% with almost no accuracy loss.

4.5 Visualization

In Figs. 15 and 16, we show some detection examples using the YOLOE-P6 model on the FZU-PPE dataset. These examples cover a wide range of scenarios that may affect detection accuracy, such as visual range, light changes, human pose, and occlusion. These examples show that our proposed lightweight detection model has good detection performance with good generalization and robustness to different scenarios and can be extended to different industries in the industrial production field.

Fig. 16 Samples include both worn and unworn detection results of the same image

5 Conclusion and discussion

Enhancing on-site individuals' safety is an essential requirement for developing intelligent industrial production. The use of a deep learning-based PPE intelligent detection system provides an effective means of reducing the risk of safety accidents occurring in industrial production activities.

In this paper, we propose a lightweight object detection algorithm for green edge computing that can detect whether workers are wearing PPEs in industrial scenarios. First, we construct a dataset containing four PPEs, FZU-PPE. In this paper, extensive experiments on the FZU-PPE dataset validate the effectiveness of our proposed lightweight PPE object detection algorithm. Second, to reduce the model size and GFLOPS, we optimize the network structure of the algorithm by two methods: reducing the number of output channels of the Conv and C3 modules in the head part of the detection network and using a 1×1 convolution kernel to predict the detection results. The model size and GFLOPS of the optimized network PPENet are reduced by nearly 30%. Third, to address the limitations of embedded and mobile devices in terms of memory occupation as well as computational resources, we propose a channel pruning algorithm based on the BN layer scaling factor γ. The minimum model size after pruning is only 1.82 MB, and the detection speed exceeds 100 FPS. Finally, we conduct experiments on open datasets PASCAL VOC, GDUT-HWD, and a private dataset FZU-CND, all of which achieve good results and prove that our lightweight target detection algorithm has good generalizability.

In future research, we will further improve our FZU-PPE dataset by adding relevant images and marking data about gloves and masks, and adding more real data images of different site scenarios to improve generalization. In addition, we will improve the model pruning algorithm to improve the detection accuracy of the model under the condition that the model size and GFLOPS meet the requirements of Internet edge devices.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grant 61972097 and U1705262, in part by the Natural Science Foundation of Fujian Province under Grant 2021J01612, 2020J01494 and 2018J07005, in part by the Major Project of Fujian Province under Grant 2021HZ022007, in part by the Industry-Academy Cooperation Project of Fujian Province under Grant 2018H6010, in part by the University Production Project of Fujian Province under 2017H6008, in part by the Fujian Collaborative Innovation Center for Big Data Application in Governments, and in part by the Fujian Engineering Research Center of Big Data Analysis and Processing. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers.

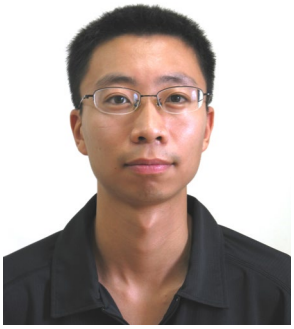
Declarations

Conflict of interest All authors in this work declare that they have no conflict of interest.

References

1. United States Census Bureau. Construction spending. https://www.census.gov/construction/c30/prpdf.html. Accessed 1 Jul 2021
2. Bureau of Labor Statistics. Industries at a glance: Construction. https://www.bls.gov/iag/tgs/iag23.htm. Accessed 1 Jul 2021
3. Bureau of Labor Statistics. Fatal occupational injuries counts and rates by selected industries. https://www.bls.gov/news.release/cfoi.t04.htm. Accessed 1 Jul 2021
4. OSHA. Commonly used statistics. https://www.osha.gov/oshstats/commonstats.html. Accessed 1 Jul 2021
5. OSHA. Worker safety series: construction. https://www.osha.gov/Publications/OSHA3252/3252.html. Accessed 1 Jul 2021
6. Centers for Disease Control and Prevention. Traumatic brain injuries in construction. https://blogs.cdc.gov/niosh-science-blog/2016/03/21/constructiontbi/. Accessed 1 Jul 2021
7. Kim J, Mesmakhosroshahi M (2015) Stereo-based region of interest generation for real-time pedestrian detection. Peer-to-Peer Netw Appl 8(2):181–188
8. Sudha M (2021) Traffic sign detection and recognition using RGSM and a novel feature extraction method. Peer-to-Peer Netw Appl 1–12
9. Jin S, Gao Y, Chen L (2020) Improved deep distance learning for visual loop closure detection in smart city. Peer-to-Peer Netw Appl 13(4):1260–1271
10. Sun Z, Tang S, Huang H et al (2017) SOS: Real-time and accurate physical assault detection using smartphone. Peer-to-Peer Netw Appl 10(2):395–410
11. Yang F, Li F, Zhang K et al (2021) Influencing factors analysis in pear disease recognition using deep learning. Peer-to-Peer Netw Appl 14(3):1816–1828
12. Meng T, Wolter K, Wu H et al (2018) A secure and cost-efficient offloading policy for mobile cloud computing against timing attacks. Pervasive Mob Comput 45:4–18
13. Lu Y, Yi S, Zeng N et al (2017) Identification of rice diseases using deep convolutional neural networks. Neurocomputing 267:378–384
14. Zhang H, Yan X, Li H et al (2019) Real-time alarming, monitoring, and locating for non-hard-hat use in construction. J Constr Eng Manag 145(3):04019006
15. Dong S, He Q, Li H et al (2015) Automated PPE misuse identification and assessment for safety performance enhancement. In ICCREM 2015 204–214
16. Zhang C, Tian Z, Song J et al (2021) Construction worker hardhat-wearing detection based on an improved BiFPN. 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, p 8600–8607
17. Wang L, Xie L, Yang P et al (2020) Hardhat-wearing detection based on a lightweight convolutional neural network with multi-scale features and a top-down module. Sensors 20(7):1868
18. Filatov N, Maltseva N, Bakhshiev A (2020) Development of hard hat wearing monitoring system using deep neural networks with high inference speed. International Russian Automation Conference (RusAutoCon), IEEE, p 459–463
19. Wu J, Cai N, Chen W et al (2019) Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset. Autom Constr 106:102894
20. Mneymneh BE, Abbas M, Khoury H (2019) Vision-based framework for intelligent monitoring of hardhat wearing on construction sites. J Comput Civ Eng 33(2):04018066
21. Wójcik B, Żarski M, Książek K et al (2021) Hard hat wearing detection based on head keypoint localization. arXiv preprint: arXiv:2106.10944

22. Fang Q, Li H, Luo X et al (2018) Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. Autom Constr 85:1–9
23. Ren S, He K, Girshick R et al (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst 28:91–99
24. Seong H, Son H, Kim C (2018) A comparative study of machine learning classification for color-based safety vest detection on construction-site images. KSCE J Civ Eng 22(11):4254–4262
25. Yu J, Zhang W (2021) Face mask wearing detection algorithm based on improved YOLO-v4. Sensors 21(9):3263
26. Loey M, Manogaran G, Taha MHN et al (2021) A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. Measurement 167:108288
27. Nagrath P, Jain R, Madan A et al (2021) SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. Sustain Cities Soc 66:102692
28. Li H, Liu J, Liu RW, Xiong N, Wu K, Kim T (2017) A dimensionality reduction-based multi-step clustering method for robust vessel trajectory analysis. Sensors 17(8):1792
29. Smartvid.io, Inc. Smartvid.io. https://www.smartvid.io. Accessed 1 Jul 2021
30. Nath ND, Behzadan AH, Paal SG (2020) Deep learning for site safety: Real-time detection of personal protective equipment. Autom Constr 112:103085
31. Fang W, Yao X, Zhao X, Yin J, Xiong N (2016) A stochastic control approach to maximize profit on service provisioning for mobile cloudlet platforms. IEEE Trans Syst Man Cybern Syst 48(4):522–534
32. Kyrkou C, Plastiras G, Theocharides T et al (2018) DroNet: Efficient convolutional neural network detector for real-time UAV applications. 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, p 967–972
33. Szegedy C, Toshev A, Erhan D (2013) Deep neural networks for object detection
34. Girshick R (2015) Fast r-cnn. Proceedings of the IEEE international conference on computer vision, p 1440–1448
35. He K, Gkioxari G, Dollár P et al (2017) Mask r-cnn. Proceedings of the IEEE international conference on computer vision, p 2961–2969
36. Duan K, Xie L, Qi H et al (2020) Corner proposal network for anchor-free, two-stage object detection. Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16, Springer International Publishing, p 399–416
37. Liu W, Anguelov D, Erhan D et al (2016) Ssd: Single shot multibox detector. European conference on computer vision, Springer, Cham, p 21–37
38. Bochkovskiy A, Wang CY, Liao HYM (2020) Yolov4: Optimal speed and accuracy of object detection. arXiv preprint: arXiv:2004.10934
39. Duan K, Bai S, Xie L et al (2019) Centernet: Keypoint triplets for object detection. Proceedings of the IEEE/CVF International Conference on Computer Vision, p 6569–6578
40. Chen Q, Wang Y, Yang T et al (2021) You only look one-level feature. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, p 13039–13048
41. Wang CY, Liao HYM, Wu YH et al (2020) CSPNet: A new backbone that can enhance learning capability of CNN. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, p 390–391
42. Liu S, Qi L, Qin H et al (2018) Path aggregation network for instance segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, p 8759–8768
43. Wang RJ, Li X, Ling CX (2018) Pelee: A real-time object detection system on mobile devices. arXiv preprint: arXiv:1804.06882
44. Liu Z, Li J, Shen Z et al (2017) Learning efficient convolutional networks through network slimming. Proceedings of the IEEE international conference on computer vision, p 2736–2744
45. Han S, Pool J, Tran J et al (2015) Learning both weights and connections for efficient neural networks. arXiv preprint: arXiv:1506.02626
46. Srinivas S, Subramanya A, Venkatesh Babu R (2017) Training sparse neural networks. Proceedings of the IEEE conference on computer vision and pattern recognition workshops, p 138–145
47. Li H, Kadav A, Durdanovic I et al (2016) Pruning filters for efficient convnets. arXiv preprint: arXiv:1608.08710
48. Changpinyo S, Sandler M, Zhmoginov A (2017) The power of sparsity in convolutional neural network. arXiv preprint: arXiv:1702.06257
49. Wang CY, Bochkovskiy A, Liao HYM (2021) Scaled-yolov4: Scaling cross stage partial network. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, p 13029–13038
50. Ge Z, Liu S, Wang F et al (2021) Yolox: Exceeding yolo series in 2021. arXiv preprint: arXiv:2107.08430



Xiao Ke received the Ph.D. degree in Artificial Intelligence from Xiamen University, Xiamen, China, in 2011. He is currently an associate professor with the College of Mathematics and Computer Science at Fuzhou University. His research interests include computer vision, pattern recognition, machine learning, and artificial intelligence. He has published 60+ papers in journal and conferences like: *IEEE Transaction on Multimedia, IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Circuits and Systems for Vedio Technology, Pattern Recognition, Neurocomputing*, et al.

Wenyao Chen is currently pursuing the M.S. degree with the college of mathematics and computer science, Fuzhou University. His research interests relate to computer vision, deep learning, model compression in edge computing and their relations with innovative technology.

Wenzhong Guo received the B.S. and M.S. degrees in Computer Science from Fuzhou University, Fuzhou, China, in 2000 and 2003, respectively, and the Ph.D degree in Communication and Informations System from Fuzhou University in 2010. He is currently a full professor with the College of Mathematics and Computer Science at Fuzhou University. His research interests include intelligent information processing, sensor networks, network computing, and network performance evaluation. Currently, he leads the Network Computing & Intelligent Information Processing Lab, which is a key Lab of Fujian Province, China. He is a member of ACM, a senior member of China Computer Federation(CCF).