




# MCCP: Multi-Collaboration Channel Pruning for Model Compression

Yuanchao Yan<sup>1</sup> · Bo Liu<sup>1</sup> · Weiwei Lin<sup>2</sup>  · Yuping Chen<sup>1</sup> · Keqin Li<sup>3</sup> · Jiangtao Ou<sup>4</sup> · Chengyuan Fan<sup>4</sup>

Accepted: 21 July 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

## Abstract

It is difficult to load large deep neural networks on resource-constrained devices. Channel pruning can compress the model and effectively reduce the resource demand to solve this problem. However, most channel pruning methods evaluate channels one-sidedly and sometimes remove important channels incorrectly. Thus, we propose a new multi-collaboration channel pruning (MCCP) method by analyzing the input and output of the batch normalization (BN) layer. The importance of the channel is evaluated by combining the weights of the convolution layer and the two learnable parameters of the BN layer to achieve more reasonable pruning. Besides, we impose polarization regularization on the scaling factors of neurons to make them easier to distinguish between important and unimportant channels to minimize the performance loss of the model after pruning. We confirm the effect of our method. MCCP reduces the number of parameters of the YOLOv3 model by 95.9%, improves the inference

---

✉ Weiwei Lin  
linww@scut.edu.cn

Yuanchao Yan  
yyc9580@qq.com

Bo Liu  
liugubin530@126.com

Yuping Chen  
1127952401@qq.com

Keqin Li  
lik@newpaltz.edu

Jiangtao Ou  
jiangtaoou@ieee.org

Chengyuan Fan  
chengyuanfan@ieee.org

<sup>1</sup> School of Computer Science, South China Normal University, GuangZhou 510631, People's Republic of China

<sup>2</sup> School of Computer Science and Engineering, South China University of Technology, GuangZhou 510006, People's Republic of China

<sup>3</sup> Department of Computer Science, State University of New York., New Paltz New York 12561, USA

<sup>4</sup> AI Sensing Technology, Chancheng District, Foshan 528000, People's Republic of China

speed by 3.8 times, compresses the model volume to 9.6MB, and has comparable recognition accuracy.

**Keywords** Channel pruning · Model compression · Object detection

## 1 Introduction

Convolutional neural networks (CNNs) are developing rapidly and have made great achievements in the fields of computer vision, such as object detection [1–4], image classification [5–7], semantic segmentation [8], and human pose estimation [9, 10]. However, with the increasing complexity of the model, the demand for computing and storage resources is getting higher and higher, and the requirements cannot be met on the resource-constrained devices (drones, self-driving cars, etc.), so model deployment is difficult to complete. In particular, edge computing [11] and autonomous driving [12] are developing rapidly and becoming popular. These fields are characterized by lightweight and low latency.

To solve the above contradiction, many researchers use model compression to reduce redundant network parameters, so as to save storage and computing resources and enable them to deploy on resource-constrained devices. Typically, the methods of model compression include low-rank decomposition [13], weight quantization [14–17], knowledge distillation [18] and pruning [19–23]. Low-rank decomposition is based on the information redundancy of the convolution kernel matrix to sparse the convolution kernel matrix by merging dimensions and imposing low-rank constraints. Weight quantization refers to the use of lower bit width to represent typical 32-bit floating-point network parameters to achieve compression. But the application scenarios of both are very limited. Knowledge distillation is to train a compact neural network by extracting the knowledge of a trained large model, but generally requires a professional artificial design of the student network. In contrast, pruning is simpler to compress models and can greatly save storage space and computing costs.

Pruning includes unstructured pruning [19, 24] and structured pruning [20–23]. The granularity of unstructured pruning is finer. The subdivision of the network structure can be accurate to any redundant parameters between neurons, such as a weight. Yet, such fine-grained pruning usually leads to irregular network structure and discontinuous storage space, and the acceleration effect is not obvious. Moreover, the application scope of unstructured pruning is relatively small and usually requires the support of some specific software and hardware. By contrast, the granularity of structured pruning is coarse, and the unit of pruning is usually a channel. By deleting unimportant channels, the network structure has not been changed, so we can more effectively use the existing framework to compress and accelerate the model. The channel-level pruning methods can not only maintain the original model structure but also reduce the number of model parameters, so as to achieve the effect of the inference acceleration.

Most channel pruning methods choose one or two influencing factors that can determine the performance of the model as the channel evaluation factors and then prune the corresponding channel through the analysis of the evaluation factors. [25] proposed to calculate the sum of the absolute values of all weights in the filter and then deleted the filter with a smaller value. He believed that the smaller values of the filters, the weaker important they are. [26] and [27] used L1 regularization to punish the scaling factors of the BN layer [28] and deleted the corresponding channels with smaller scaling factors. [29] and [30] also considered another parameter of the BN layer. They believed that not only the scale factors but

also the shift factors should be considered. [31] evaluated the importance of channels by combining the weights of the convolution layer and the scaling factors of the BN layer.

The above-mentioned methods only consider a single or a combination of two influencing factors to evaluate the importance of the channels, which may lead to the loss of some important channels. When the compression ratio becomes high, it is difficult to restore the lost performance even after fine-tuning. In this article, by analyzing the input and output of the BN layer, we propose a new method to delete unimportant channels. This method combines the weights of the convolution layer and the two learnable parameters of the BN layer.

The main contributions of our work are as follows:

1. We comprehensively analyze the input and output of the BN layer, focusing on the influencing factors before the activation function, and verify the influence of the weights of the convolutional layer and the two learnable parameters of the BN layer on the output.
2. To weigh the effects of the three influencing factors, we propose a multi-collaboration channel pruning method. We perform regularization penalties on the convolutional layer and the BN layer. And we delete the channels by the weights and scaling factors, and then retain the channels by the shift factors.
3. To achieve a better sparsity effect and make it easier for the parameters to distinguish channels with lower importance, we use polarization regularization to punish the scaling factors.

## 2 Related Work

According to whether it is hardware-friendly or the size of pruning granularity, we can divide pruning into unstructured pruning and structured pruning. Unstructured pruning is mainly deleting the connection weights of neurons. For example, [19] deleted connections with a smaller connection weight norm between neurons. In order to avoid deleting important connections, [32] and [33] restored some of the deleted connections after pruning. [34] used quantization and Huffman coding to further compress after deleting redundant connections. [35] used the concept of biological neural synapse and took the product of the scaling factors and the Frobenius norms of the filter as the synaptic strength to indicate the importance of connections between neurons. However, unstructured pruning will lead to network structure being irregular and storage structure being discontinuous, which makes it difficult to optimize storage space and runtime memory. And it needs specific hardware support, which adds additional resource requirements.

The common method of structured pruning is channel pruning. Because it does not require specific hardware to support, it is more practical than unstructured pruning. According to the design of the filter evaluation function, there are two ideas for pruning filters, one is based on the inherent properties of CNN, and the other is to customize an evaluation factor.

The methods based on the inherent properties of CNN use the inherent properties to prune, such as activation value and weight. [36] believed that neurons with an activation value of 0 are redundant and deleted channels with an activation value of 0. [20] proposed to delete the filter with a small L2 norm. [21] proposed to iteratively remove filters based on the average activation value of all training set samples. [22] selected unimportant channels by the first-order gradient.

The methods of custom evaluation factors mine attribute features and take some influencing factors as the evaluation factors of the channels through transformation. [37] proposed to

remove filters based on the geometric median value in the layer. [35] obtained the importance score of each feature through the feature selector and then obtained the importance score of the channels through backpropagation to remove the low-scoring channels. [38] proved that the corresponding feature maps generated by a single filter have the same average rank. They believed that the filters with a lower-rank feature map have a small amount of information and can be safely deleted. [39] introduced a new scaling factor parameter to learn the sparse structure and removed the filter corresponding to the scaling factor of zero. [30] multiplied the two learnable parameters of the BN layer and then deleted the redundant channels according to the distribution of the scaling factors. But because the structure was changed and the performance of the model was affected, the very deep neural network with many residual structures [5] was not suitable.

Many of the above methods lack a comprehensive consideration of the channels, and it is easy to lose more performance after pruning. The method of introducing additional evaluation factors to select the redundant channel may change the original structure, thereby affecting the performance of the model. The M CCP method we proposed considers both the BN layer and the continuous convolutional layer. It combines three influencing factors to determine whether the channels are important and does not affect the original structure, so as to achieve a more reasonable pruning.

In addition to pruning, weight quantization, low-rank decomposition, and knowledge distillation also can effectively compress the model, and there is no conflict between the methods. These methods can be combined to achieve further compression. For example, [34] use quantisation to achieve more after pruning. [24] used knowledge distillation to fine-tune after pruning to restore the performance of the model. Our method can also be used in combination with other compression methods.

### 3 Multi-Collaboration Channel Pruning

The key to channel pruning is how to find unimportant channels and delete them, so as to effectively compress the model. Most channel pruning methods only consider a single influencing factor or a combination of two influencing factors to determine whether the channels are important. We believe that the above methods lack partial rationality and do not fully consider the determinants of channel importance. In this section, we consider the convolutional layer and the BN layer of the continuous structure of the network and then analyze the input and output of the BN layer. Finally, we get three parameters that need to be considered and then propose corresponding method.

#### 3.1 Input and Output Analysis of the BN Layer

Most deep neural networks for image recognition are composed of continuous convolution layers. The output of the convolution layer can be represented by Eq. (1):

$$Z_l = X_l W_l + b_l \quad (1)$$

where index  $l$  denotes the  $l$ -th layer,  $X_l$  is the input ( $X_l \in \mathbb{R}^{M_l \times N_l \times n_l}$ , where  $M_l$  and  $N_l$  denote the height and width of the input feature map,  $n_l$  is the number of input channels),  $W_l$  is the set of weights ( $W_l \in \mathbb{R}^{n_{l+1} \times n_l \times K_l \times G_l}$ , where  $K_l$ ,  $G_l$  and  $n_l$  are the width, height and the number of input channels of the filter, and  $n_{l+1}$  is the number of output channels),  $b_l$  is the bias ( $b_l \in \mathbb{R}^{n_{l+1}}$ ), and  $Z_l$  is the output feature map ( $Z_l \in \mathbb{R}^{M_{l+1} \times N_{l+1} \times n_{l+1}}$ ). It can be seen

as  $n_l$  feature maps and  $n_{l+1}$  filters to do convolution calculate, and then  $n_{l+1}$  output feature maps can be obtained.

[25] believed that those filters with a smaller sum of absolute values are not important and can be deleted. Even when the BN layer is not added to normalize the input value, the output of the  $l$ -th layer is determined by  $W_l$  and  $b_l$  in Eq. (1). [25] can effectively compress the model, but it is not reasonable enough. To select important channels from all channels, the weight  $W_l$  and the bias  $b_l$  of the convolutional layer need to be considered at the same time. At present, most large convolutional networks choose to add the BN layer to alleviate model over-fitting. Therefore, we should also consider another structure the BN layer.

Generally, after adding the BN layer, we will discard the bias  $b_l$ , and then the output of the convolution layer is represented by Eq. (2):

$$Z_l = X_l W_l \tag{2}$$

The function of the BN layer [28] is to standardize the input values and reduce the difference of values to the same range. This makes each layer of the network more independent, thereby alleviating the problem of disappearance gradients during the training process. The conversion of the BN layer can be expressed by Eq. (3):

$$Y_l = \gamma_l \times \frac{Z_l - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta_l = \gamma_l \times \frac{X_l W_l - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta_l \tag{3}$$

where  $\mu_B$  and  $\sigma_B^2$  are the expectation and the variance of the input  $X_l$  respectively.  $\epsilon$  is a constant slightly greater than zero to ensure that the denominator is not zero.  $\mu_B$ ,  $\sigma_B^2$  and  $\epsilon$  are real numbers related to input  $X_l$ .  $\beta_l$  is the shift factor of the BN layer ( $\beta_l = [\beta_l^1, \beta_l^2, \dots, \beta_l^{n_{l+1}}]$ , it denotes the  $n_{l+1}$  channels of the  $l$ -th layer),  $\gamma_l$  is the scale factor of the BN layer ( $\gamma_l = [\gamma_l^1, \gamma_l^2, \dots, \gamma_l^{n_{l+1}}]$ ),  $\beta$  and  $\gamma$  provide the possibility of linearly transforming normalized activations back to any scales, and  $Y_l$  is the output of the  $l$ -th layer ( $Y_l \in \mathbb{R}^{M_{l+1} \times N_{l+1} \times n_{l+1}}$ ).

From Eq. (3), we can see that the output of the  $l$ -th layer is related to the convolutional layer  $W_l$ , the BN layer  $\gamma_l$ , and the BN layer  $\beta_l$ .

### 3.2 Channel Selection

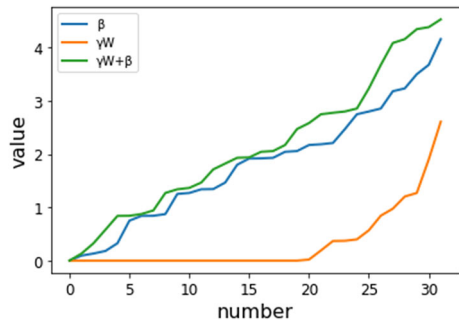
In Eq. (3),  $\gamma_l$ ,  $W_l$ , and  $\beta_l$  are learnable parameters, and the performance of a trained model is determined by these three learnable parameters. A set of  $\gamma$ ,  $W$ , and  $\beta$  corresponds to a channel, so we can use any one parameter in the set as an evaluation factor. When considering the influence of the BN layer, most researchers use  $\gamma_l$  as the evaluation factor, such as [27] and [26]. There is also a combination of  $W_l$  and  $\gamma_l$  as the evaluation factor of the channel, such as [31]. These methods can effectively perform channel pruning, but they do not fully consider what factors affect the output of the BN layer, which will cause important channels to be deleted. The output of the BN layer is related to three learning parameters.

In the past, only the  $\gamma$  of the BN layer was used as the evaluation factor of the channel, but when  $\gamma_l$  is very small, that is, when  $\gamma_l \approx 0$ , Eq. (3) can be transformed into Eq. (4):

$$Y_l \approx \beta_l \tag{4}$$

At this time, the output of layer is determined by the shift factor  $\beta_l$ . So, when  $\gamma_l$  is very small but  $\beta_l$  is very large, the influence of  $\beta_l$  cannot be ignored. [29] analyzed the normalized input data that obeyed the Gaussian distribution and found that different linear transformations of  $\gamma$  and  $\beta$  would severely affect the Gaussian distribution. Therefore, through the

**Fig. 1** Line chart of size distribution of each parameter, the horizontal axis is the order of channels, the vertical axis is the size of the value



comprehensive consideration of the three learnable parameters, the importance of the channel can be evaluated more reasonably.

In Eq. (3), the output  $Y_l$  of the BN layer is proportional to the product of  $\gamma_l$  and  $W_l$ , and  $Y_l$  is also proportional to  $\beta_l$ . So, we set up the following equation and use the result of the equation as the evaluation factor of the channel, and in order to match  $\gamma_l$ , we sum over  $W_l$  to reduce its dimension, as shown in Eq. (5):

$$F_l^i = |\gamma_l^i | \sum_{j=1}^{n_l} \sum_{k=1}^{K_l} \sum_{g=1}^{G_l} |W_l^j | + \beta_l^i \text{ for } i = 1, 2, \dots, n_{l+1} \tag{5}$$

We use their definitely worth to calculate the result, because positive numbers are always larger than negative numbers, but negative numbers also play a role. And we sum it up to turn  $W$  into a one-dimensional list so that  $W$  can be multiplied by  $\gamma$ . In Eq. (5),  $F_l = [\gamma_l^1 W_l^1 + \beta_l^1, \gamma_l^2 W_l^2 + \beta_l^2, \dots, \gamma_l^{n_{l+1}} W_l^{n_{l+1}} + \beta_l^{n_{l+1}}]$ ,  $F_l$  is a list that represents all the evaluation factors for the  $l$ -th layer. But through experiments, we find the value of  $\beta$  is much larger than the values of  $\gamma$  or  $W$ . For this reason, it is not advisable to directly add its value to the equation. One solution is to impose a regularization penalty on  $\beta$ . But if  $\beta$  is also subject to a regularization penalty, more performance will be lost. We randomly select a layer and show its  $\beta$ , the product of  $\gamma$  and  $W$ , and the sum according to Eq. (5) in the Fig. 1. As can be seen from the figure, if the evaluation factor is obtained according to Eq. (5), the selection right of the channel is basically determined by the  $\beta$ . That is, the green line basically fits the blue line, while the yellow line is difficult to work.

Therefore, after analysis, we believe that when most or all the channels in a layer need to be deleted, we can approximately regard the  $\gamma$  and  $W$  of this layer as very small, which is approximately zero. At this time, the output of this layer is mainly determined by the shift factor  $\beta$ . Therefore, in Eq. (5), we first let  $\beta_l=0$ , select the redundant channels by the product of  $\gamma$  and  $W$ , and then use the layer pruning protection ratio to retain the channels with larger  $\beta$ . The new evaluation factor equation is expressed by Eq. (6):

$$F_l^i = |\gamma_l^i | \sum_{j=1}^{n_l} \sum_{k=1}^{K_l} \sum_{g=1}^{G_l} |W_l^j | \text{ for } i = 1, 2, \dots, n_{l+1} \tag{6}$$

### 3.3 Polarization Regularization

The analysis shows that the influencing factors of the output of the BN layer are the three learnable parameters  $\gamma$ ,  $W$ , and  $\beta$ . In order to reduce the performance loss after pruning, we perform sparse training to make most of the learnable parameters of the model equal to or close to zero. The general sparse training method is to use the L1-norm to penalize the scaling factors of the BN layer to obtain sparse parameters, such as [27]. But L1 regularization

lacks the ability to distinguish. After penalizing all parameters, the obtained parameters are reduced continuously. So, we use the polarization regularization method proposed by [40] to divide the scaling factors into two parts, one is close to zero, and the other is close to a certain number greater than zero. Only delete the corresponding channels whose scaling factors tends to zero, and the loss of the model is less. But the influence of the weights of the convolutional layer is not considered in [40], and sparse training is not performed on it. [31] considered the weights of convolution layer, but they used L1 regularization for sparse training, which lacked the distinction of parameters. We consider two consecutive layers and use polarization regularization to train the network. Our complete loss function is as follows:

$$Loss = L(f(X, W), Y) + \lambda_1 R(W) + \lambda_2 R_s(\gamma) \tag{7}$$

where,

$$R(W) = \sum_{l=1}^L \|W_l\|_1 \tag{8}$$

$$R_s(\gamma) = \sum_{l=1}^L \sum_{i=1}^{n_{l+1}} t |\gamma_l^i| - |\gamma_l^i - \bar{\gamma}_l|, (t \in \mathbb{R}, \bar{\gamma}_l = \frac{1}{n_{l+1}} \sum_{i=1}^{n_{l+1}} \gamma_l^i) \tag{9}$$

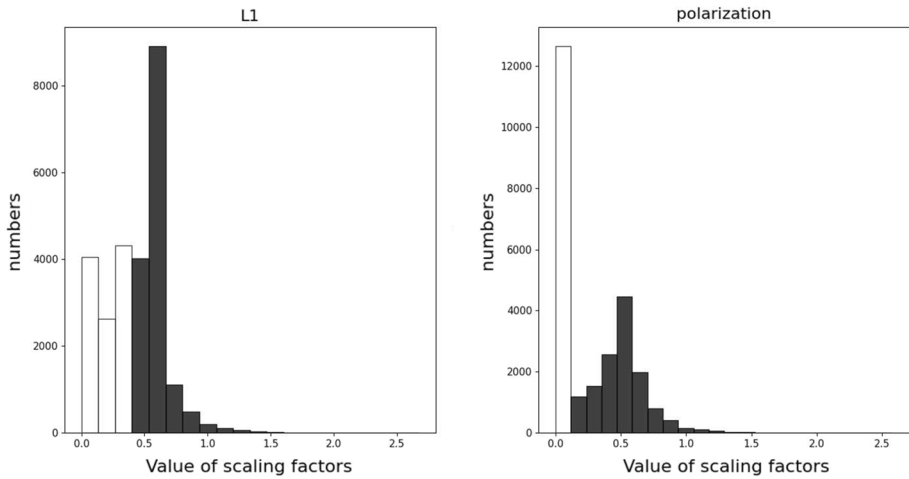
$L(f(X, W), Y)$  denotes the normal training loss of the data set  $X, Y$  is the label of sample  $X, W$  is the weight set of all convolutional layers,  $\gamma$  is the scaling factors set of all BN layers, the subscript  $l$  indicates the  $l$ -th layer.  $R(W)$  denotes the L1 regularization for the weights of convolutional layers ( $W_l \in \mathbb{R}^{n_{l+1} \times n_l \times K_l \times G_l}$ , where  $K_l, G_l$  and  $n_l$  are the width, height and the number of input channels of the filter, and  $n_{l+1}$  is the number of output channels), and  $R_s(\gamma)$  denotes the polarization regularization is applied to the BN layer ( $\gamma_l \in \mathbb{R}^{n_{l+1}}$ ).  $\lambda_1$  and  $\lambda_2$  are penalty factors.  $\bar{\gamma}_l$  is the average value of the scaling factor of the  $l$ -th layer.  $t$  is a hyperparameter of polarization regularization. It is used to adjust the size distribution of  $\gamma$ .  $t$  is approximately proportional to the number of parts that tend to zero. The larger  $t$  is, the more parameters tend to zero, but  $t$  is set too large will affect the accuracy [40].

The distribution of the scaling factor of the BN layer obtained after sparse training with two different regularization methods is shown in Fig. 2. When the pruning rate is 50%, the white is the parameter distribution that is pruned, and the black is the parameter distribution that is retained. It can be seen that the scaling factors obtained by L1 regularization has a continuous distribution and lacks distinction. It is difficult to find a reasonable pruning threshold. The performance loss of the model is great after pruning. The polarization regularization method can delete the part that tends to zero, and the performance loss is less. The specific results of the experiment are shown in Sect. 4.

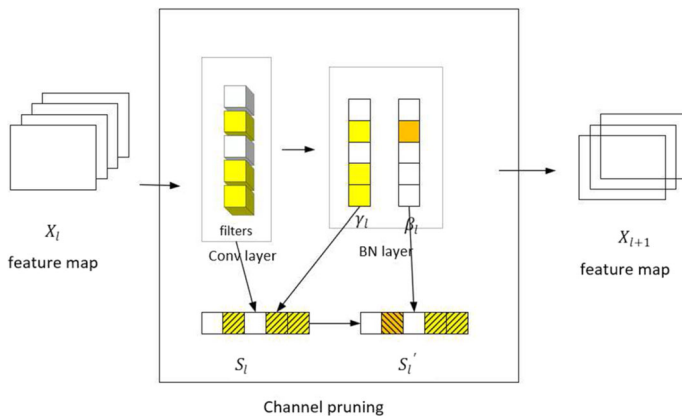
### 3.4 Implementation of Channel Pruning

#### 3.4.1 The Illustration Process

The channel pruning process of the MCCP method can be shown in Fig. 3. This is a process of channel reduction. By deleting useless channels, the number of parameters and volume of the model are reduced. The first three-dimensional image  $X_l$  denotes the input feature map of the  $l$ -th layer. In the convolution layer, we use the cube to represent the filter, and each filter corresponds to a channel. In the BN layer, we use a rectangle to represent the parameter  $\gamma$  or  $\beta$ . A rectangle is a value of  $\gamma$  or a value of  $\beta$ , and a rectangle represents a channel.  $S_l$  in the figure is all channels of the current layer. A rectangle represents a channel, yellow indicates that the channel is unimportant and can be deleted, white indicates that the channel does not need to be deleted, and orange indicates the newly determined reserved channel. Firstly,



**Fig. 2** The left image shows the scaling factors distribution under L1 regularization, and the right image shows the scaling factors distribution under polarization regularization. The horizontal axis is the value of the scaling factor and the vertical axis is the number of scaling factors



**Fig. 3** The process of reducing the number of channels after channel pruning

calculate the weight of the convolution layer and the  $\gamma$  of the BN layer according to Eq. (6) to obtain the list of evaluation factors for each channel. Select the deleted channel according to the size of the evaluation factor. Then, select the important channels to be reserved through  $\beta$  of BN layer, and finally get a list  $S'_l$  with an identity, this list indicates which channel needs to be deleted. The final three-dimensional image  $X_{l+1}$  denotes the output feature map of the  $l$ -th layer.

In Fig. 3, the input feature map of the four channels will output feature map with the number of channels five after the convolution calculation of the five filters, but the feature map with the number of channels three will be output through the channel pruning.



### 3.4.2 Algorithm Implementation

Based on the above work, we propose an algorithm as shown in Algorithm 1. Algorithm 1 is used to get a list of mask, this list indicates which channels in each layer need to be removed. We use  $global\_percent$  as the global pruning ratio and  $layer\_keep$  as the layer-level protection ratio to control the size of the pruning. In Algorithm 1,  $sort(x)$  is a sorting algorithm, which sorts the numbers in  $x$  from small to large.

---

#### Algorithm 1 MCCP algorithm

**Input:** Filter weight list of convolution layer, scale factor list  $\gamma$  of BN layer, shift factor list  $\beta$  of BN layer,  $global\_percent, layer\_keep$

**Output:** A list  $mask$  indicates whether the channels need to be deleted.

- 1: Obtain the evaluation factors of all channels  $F$  according to Eq. (6)
  - 2: Calculate the global pruning threshold, sort  $F$  from small to large, and intercept the threshold  $thresh$  at the corresponding position according to global percent.
  - 3: Loop through each channel to determine whether it needs to be deleted
  - 4: **for**  $l = 0$  to  $L$  **do**
  - 5:      $remain\_channels = 0$
  - 6:     **for**  $i = 0$  to  $I$  **do**
  - 7:         **if**  $S_l^i \leq thresh$  **then**
  - 8:              $S_l^i = 0$
  - 9:         **else**
  - 10:              $S_l^i = 1$
  - 11:              $remain\_channels ++$
  - 12:         **end if**
  - 13:     **end for**
  - 14:     **if**  $remain\_channels < layer\_keep \times I$  **then**
  - 15:          $\beta_l \leftarrow |\beta|$  of current layer
  - 16:          $sorted\_beta_l = sort(\beta_l)$
  - 17:          $\hat{b} = sorted\_beta_l[(1 - layer\_keep) \times I]$
  - 18:         **for**  $i = 0$  to  $I$  **do**
  - 19:             **if**  $\beta_l^i \geq \hat{b}$  **then**
  - 20:                  $S_l^i = 1$
  - 21:             **end if**
  - 22:         **end for**
  - 23:     **end if**
  - 24: **end for**
- 

Algorithm 1 first gets the evaluation factor list  $F(F \in \mathbb{R}^n, n$  is the number of all channels in the network) of all channels according to Eq. (6), then sorts  $F$  from small to large, and intercepts the threshold  $thresh$  at the corresponding position according to  $global\_percent$ . Then it loops through the evaluation factors corresponding to each channel of each layer, and compares it with the  $thresh$ . The channels that are smaller than the pruning threshold have their  $S_l^i$  set to 0.  $S_l^i$  is the channel mask,  $S_l^i = 0$  denotes that the  $i$ -th channel of the  $l$ -th layer needs to be deleted, and  $S_l^i = 1$  denotes that the channel is reserved. After traversing all channels in a layer, calculate the number of reserved channels. If it is less than the maximum number of pruned at the layer, set the  $S_l^i$  of those channels whose shift factor is greater than the threshold  $\hat{b}$  to 1. The threshold  $\hat{b}$  is obtained by sorting all the shift factors of the current layer from small to large and then intercepting it with  $layer\_keep$ , which is similar to obtaining  $thresh$ . Then continue to loop through the next layer until all layers have been traversed.

After executing Algorithm 1, the list *mask* can be obtained. This list contains the  $S_l^i$  of each channel at each layer, which denotes whether each channel in a layer needs to be deleted.

## 4 Experiments and Analysis

We use a Linux server with Intel(R) Xeon(R) CPU E5-2678 v3 @ 2.50GHz (48 CPUs), 32 GB RAM and four NVIDIA GTX2080Ti GPU cards for training models. The system version is Ubuntu 18.04.5 LTS. All experiments are implemented on the Ubuntu 18.04.5 operating system using the Pytorch 1.8.2 deep learning framework.

### 4.1 Dataset

To verify the effectiveness of the method, we conducted experiments on two datasets.

#### 4.1.1 Pedestrian Detection Dataset

The dataset includes 3008 surveillance images taken by surveillance cameras at different periods. The dataset includes 2402 training set images and 606 validation set images. The images are manually marked into four classes (i.e., pedestrians, heads, motorcycles, and bicycles).

#### 4.1.2 Oxfordhand Dataset

The dataset includes 5628 images collected from various different public image data set sources as listed in [41]. In each image, all the hands that can be perceived clearly by humans are annotated. The dataset includes 4069 training set images, 821 test set images and 738 validation set images. The images are manually marked into one class (hand).

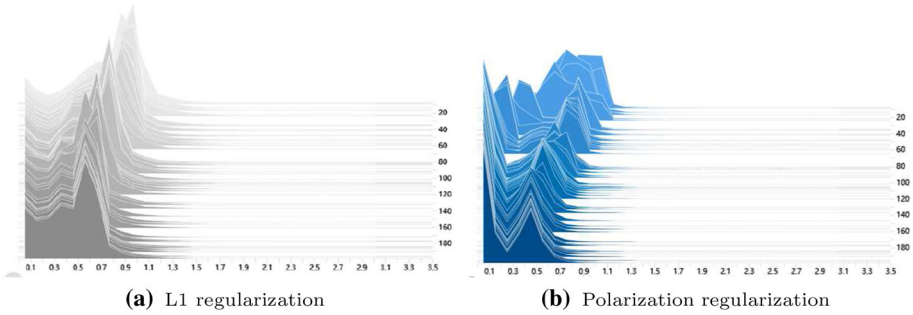
## 4.2 Benchmark Model

### 4.2.1 Object Detection Algorithm

Object detection algorithms include two types: two-stage and one-stage object detection algorithms. The two-stage methods first generate candidate boxes, and then classify them through convolutional neural networks, including R-CNN [2] and R-FCN [3] and so on. The one-stage methods do not need to generate candidate boxes, and directly convert the problem of target boxes positioning into regression problems, mainly the YOLO series, such as the representative works YOLOv3 [1] and YOLOv4 [42]. The two algorithms have their advantages. The two-stage methods have advantages in detection accuracy and positioning accuracy, and the one-stage methods have advantages in detection speed. Our detection task has higher requirements for time delay, so we choose a one-stage target detection network.

### 4.2.2 Benchmark

We choose a one-stage algorithm, its typical representative YOLOv3, as the basic network model, and three models of different sizes as the benchmark experiment. The sizes of the



**Fig. 4** The histogram of the scaling factor. The width, height, and depth of the histogram correspond to the value of the scaling factor, the number of scaling factors, and the number of training times

three models are obtained by using the SlimYOLOv3 [26] to prune the YOLOv3 model when the global pruning rates are 50%, 90% and 95%.

## 4.3 Training

### 4.3.1 Normal Training

We train the YOLOv3 network on the two datasets following the default parameter settings of Darknet. The input image size of our YOLOv3 network is  $416 \times 416$ . We use the SGD optimizer to train and set the momentum to 0.9, the weight decay to 0.0005, max batches set to 10000, and the initial value of the learning rate to 0.001.

### 4.3.2 Sparsity Training

When training sparsely on two datasets, our network input size is  $416 \times 416$ , SGD is used to train all networks, momentum is set to 0.97, weight decay is set to 0.0004569, max epochs is set to 200, and the initial value of the learning rate is set to 0.002324.

The effects of L1 regularization and polarization regularization can be visually represented by the histogram of the evolution of the scaling factors distribution in the training process, as shown in Fig. 4. The influence of the learning rate should be greater than that of the hyperparameter, so the value of the hyperparameter should be set smaller than the learning rate. In the experiment shown in the Fig. 4, the test dataset is pedestrian detection dataset. The hyperparameters of the polarization regularization method we use are  $t = 2.0$  and  $\lambda_2 = 5e^{-4}$ . The penalty factor used by L1 regularization method is  $5e^{-4}$ . It can be seen from Fig. 4 that the polarization regularization gradually distributes the scaling factors into two crests, one crest pole is zero, and the other pole is a certain number greater than zero. This shows that polarization regularization can push the scale factors to different extremes so that there is a distinction between the scale factors, and so that it is easier to distinguish between reserved channels and deleted channels.

## 4.4 Fine-tuning

If the channels are pruned too much, the performance of the model will be severely degraded, and the distribution structure of the parameters of the model will also change. Fine-tuning

can restore the performance of the model. The hyperparameters we use for fine-tuning are the same as sparse training, but no regularization penalty is performed.

#### 4.5 Evaluation Metrics

We use the following seven evaluation metrics (Precision, Recall, mAP, F1, Volume, Parameters, Time) to evaluate the performance of the model. In object detection field, Precision and Recall are related to Intersection over Union (IoU) and confidence threshold. IoU is the ratio of the intersection and union of candidate bound and ground truth bound. The higher the value, the higher the correlation. Confidence threshold is used to judge whether the object in the boundary box is a positive sample or a negative sample. If it is greater than the confidence threshold, it is a positive sample, and if it is less than the confidence threshold, it is a negative sample. In our experiment, we set the IoU threshold to 0.5 and the confidence threshold to 0.001.

- Precision : Precision is the proportion of the number of correctly predicted samples to the number of samples predicted to be true.
- Recall : Recall is the proportion of all positive samples that have been successfully predicted.
- mAP : Average Precision (AP) is the detection accuracy of a single class. mAP is the detection accuracy of all classes. AP is the quality of the model in single class and mAP measures the quality of the learned model in all classes.
- F1 : F1 is obtained through precision and recall, which is the harmonic average of precision and recall. The higher the F1 value, the better the performance of the model.
- Volume : Volume size of the model. The smaller the volume size of the model, the more suitable for deployment on resource constrained devices.
- Parameters: Number of parameters of the model. The less the parameters of the model, the less the calculation times and the faster the recognition speed.
- Time : Time is the inference time of identifying classes on one picture. The faster the speed, the less waiting time for detection. Because the time difference between before and after pruning is very small when tested on 2080Ti, we test the inference time on the more basic hardware Tesla k80.

#### 4.6 Result and Analysis

We use the method of SlimYOLOv3 to pruned on the YOLOv3 model and obtain three models (SlimYOLOv3-50, SlimYOLOv3-90, and SlimYOLOv3-95) as our benchmark. Then we use the MCCP method we proposed to prune the network, get models with the same model volume and compare their evaluation metrics. Using the slimyolov3 method, after 50%, 90% and 95% pruning, the compressed model volumes on the pedestrian detection dataset are 79MB, 27MB and 13MB respectively, and on the oxfordhand dataset, the compressed model volumes are 75MB, 23MB and 12MB respectively. We compare the evaluation metrics of the two methods under the condition of pruning the same volume. In addition, we slim the network model as much as possible, so that the model can be deployed on resource-constrained devices. The overall results of the experiment are shown in Tables 1 and 2.

**Table 1** Results on the pedestrian detection dataset

Model	Precision(%)	Recall(%)	mAP(%)	F1(%)	Parameters(Million)	Time(ms)	Volume(MB)
Yolov3	50.5	78.6	73.6	61.2	61.54	63.00	235.0
Yolov4	32.2	85.2	75.4	44.9	63.95	74.10	244.2
SlimYOLOv3-50	15.5	82.0	72.9	25.0	20.53	39.91	78.4
Ours-50	15.6	83.6	73.7	25.1	20.48	35.47	78.3
SlimYOLOv3-90	14.2	82.6	72.2	23.0	6.90	24.53	26.4
Ours-90	14.8	82.4	72.2	23.7	6.94	25.15	26.6
SlimYOLOv3-95	13.3	80.4	69.5	21.6	3.15	18.20	12.3
Ours-95	14.1	82.9	71.3	22.8	3.14	18.00	12.2
Ours-best	13.7	82.3	70.3	22.1	2.50	16.62	9.6

**Table 2** Results on oxfordhand dataset

Model	Precision(%)	Recall(%)	mAP(%)	F1(%)	Parameters(Million)	Time(ms)	Volume(MB)
Yolov3	15.8	89.7	80.4	26.9	61.5	62.3	234.9
SlimYOLOv3-50	12.3	90.8	77.2	21.6	19.4	36.0	74.1
Ours-50	12.5	90.8	78.4	22.0	19.6	32.1	74.7
SlimYOLOv3-90	11.7	90.3	77.1	20.7	5.9	21.8	22.5
Ours-90	11.4	90.8	78.8	20.3	5.7	21.6	21.8
SlimYOLOv3-95	12.3	89.0	76.4	21.6	3.0	18.1	11.8
Ours-95	11.7	90.4	77.0	20.6	3.1	17.6	12.0

#### 4.6.1 Performance Evaluation and Analysis of Different Pruning Ratios

After channel pruning, the original model is fully compressed. Compared with YOLOv3, after using the SlimYOLOv3 for pruning, on the pedestrian detection dataset the parameters of the three models are reduced by 66.6%, 88.8%, and 94.9% respectively, the model volume are reduced by 66.6%, 88.7%, and 94.8% respectively, and the inference speed increased by 1.6, 2.6, and 3.5 times respectively. And on the oxfordhand dataset, the parameters of the three models are reduced by 68.5%, 90.4%, and 95.1% respectively, the model volume are reduced by 68.5%, 90.4%, and 95.0% respectively, and the inference speed increased by 1.7, 2.9, and 3.4 times respectively.

We use our method (MCCP) for pruning and obtain the same model volume. On the pedestrian detection dataset, the parameters of the three pruned models are reduced by 66.7%, 88.7%, and 94.9% respectively. The model volume is reduced by 66.7%, 88.7%, and 94.9% respectively. The inference speed is increased by 1.8, 2.5, and 3.5 times respectively. And on the oxfordhand dataset, the parameters of the three pruned models are reduced by 68.1%, 90.7%, and 95.0% respectively. The model volume is reduced by 68.2%, 90.7%, and 94.9% respectively, and the inference speed increased by 1.9, 2.9, and 3.5 times respectively.

It can be seen from Table 1 that at any pruning rate, the mAP, precision, Recall and F1 obtained by the MCCP are higher than or equal to SlimYOLOv3. MCCP and SlimYOLOv3 have our own advantages in the recall and inference time. When the global pruning percent is 50% and 95%, the mAP we get is significantly higher than SlimYOLOv3. Compared with YOLOv4, our mAP is only slightly reduced, but the detection speed is 4.5 times faster than YOLOv4. Finally, we tested the best pruning of the MCCP, and finally get Ours-best that can get better evaluation metrics than SlimYOLOv3-95. Compared with the YOLOv3, Ours-best has a volume reduction of 95.9%, a parameter reduction of 95.9%, and the inference speed is increased by 3.8 times. The compression effect is obvious.

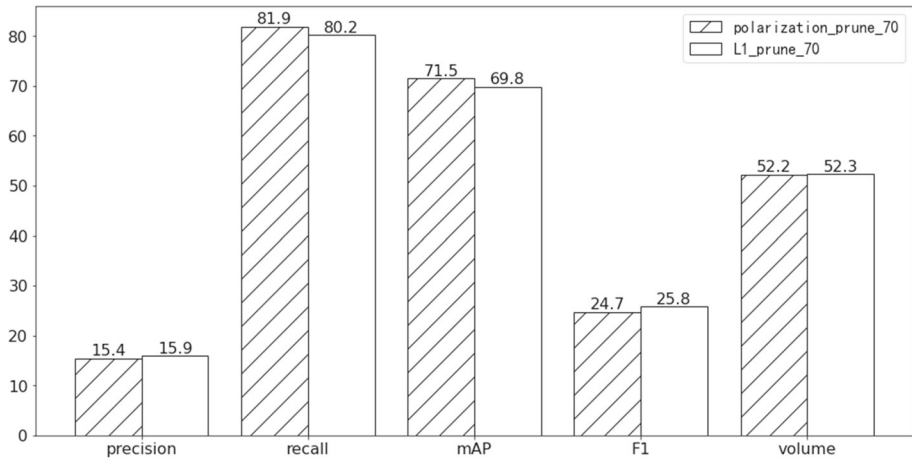
And it can be seen from Table 2 that at any pruning rate, the mAP and Recall obtained by the MCCP are higher than SlimYOLOv3. Other evaluation metrics are basically equivalent. The performance of a model is mainly determined by mAP and inference time. In high compression ratio, We can compress the model to 12.00MB, improve the inference speed by 3.5 times, and the value of mAP decreases only slightly.

Combining the analysis above, the MCCP method proposed in this paper has a better pruning effect than SlimYOLOv3.

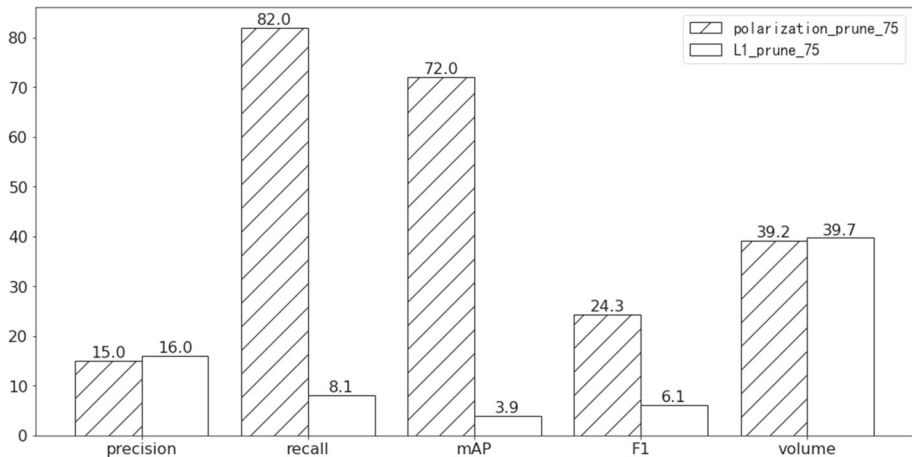
#### 4.6.2 Analysis of the Effect of Polarization Regularization

To prove the effect of polarization regularization (on pedestrian detection data), we use two different regularization methods to punish the scaling factors of the BN layer. In this experiment, the convolution layer weights are not regularized. After training 200 epochs, we get the same model volume and compare each evaluation metrics after pruning. The penalty factor used for L1 regularization is  $1e^{-3}$ , and the two hyperparameters used for polarization regularization are  $t = 2.0$ ,  $\lambda_2 = 1e^{-3}$ . The experimental results are in Figs. 5 and 6.

From Figs. 5 and 6, we can see that the model trained using L1 regularization, the evaluation metrics are lower than the corresponding evaluation metrics of the model obtained using the polarization regularization method after pruning. And when the pruning rate is 75%, the performance loss of the pruning model after L1 regularization training is great (the unfilled bar graph). But the polarization regularization method can retain higher model performance (the '/' filled bar graph), and some performance metrics are significantly higher. This shows



**Fig. 5** When pruning ratio is 70%, the value of each evaluation metrics. The horizontal axis is evaluation metrics, and the vertical axis is the value of each evaluation metrics



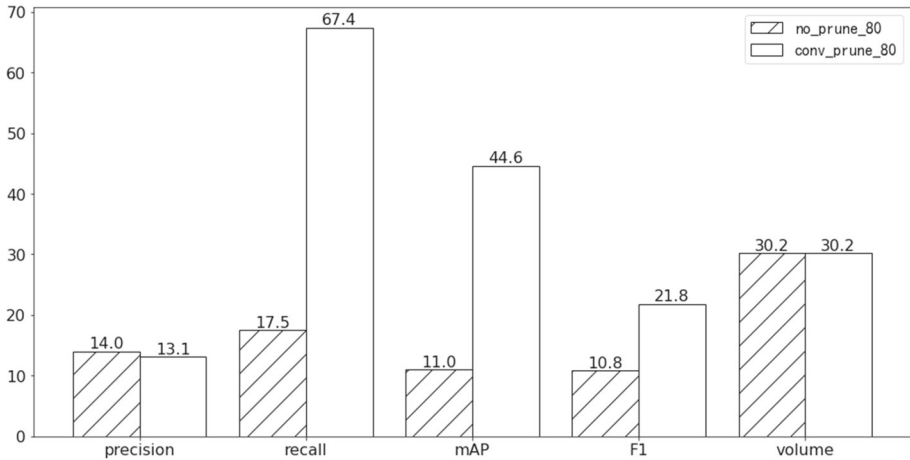
**Fig. 6** When pruning ratio is 75%, the value of each evaluation metrics. The horizontal axis is evaluation metrics, and the vertical axis is the value of each evaluation metrics

that polarization regularization has a better effect than L1 regularization, and it is easy to separate the retention channels and the redundant channels.

#### 4.6.3 Analysis of the Effect of Regularizing the Weights of the Convolutional Layer

To prove the effect of regularizing the weights of the convolutional layer (on pedestrian detection data), we perform sparse training twice. In one experiment, the weights of the convolutional layer are regularized, and the other experiment is not regularized. And then we compress model to the same volume. The hyperparameters used in the former experiment are  $t = 2.0$ ,  $\lambda_1 = 1e^{-4}$ ,  $\lambda_2 = 1e^{-3}$ . The hyperparameters used in the latter are  $t = 2.0$ ,  $\lambda_2 = 1e^{-3}$ , the experimental results are shown in Fig. 7.





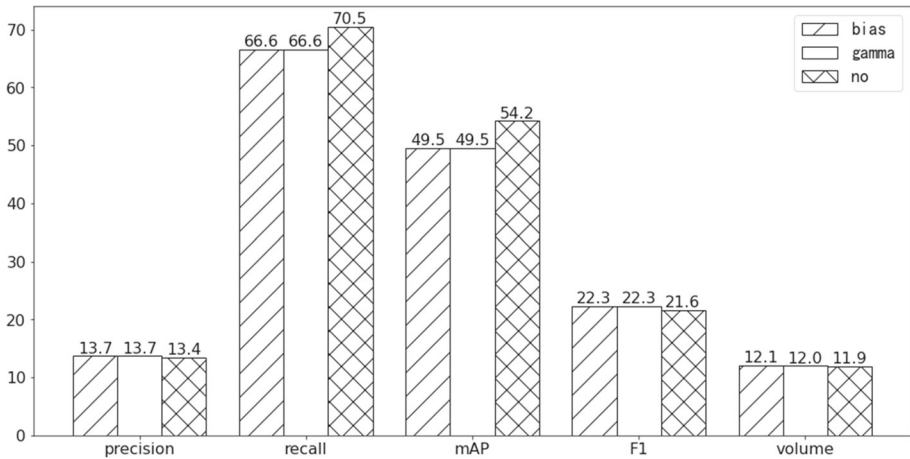
**Fig. 7** Regularization penalty and no penalty on the convolutional layer, the value of each evaluation metrics after pruning. The horizontal axis is evaluation metrics, and the vertical axis is the value of each evaluation metrics

In Fig. 7, when the pruning rate is 80%, the evaluation metrics of the model whose convolutional layer weights are not regularized drop sharply after pruning (the ‘/’ filled bar graph). The model whose convolutional layer weights are regularized have less performance loss after pruning, and the evaluation metrics are relatively higher (the unfilled bar graph). It shows that considering the continuous structure of the convolutional layer can be more reasonable pruning.

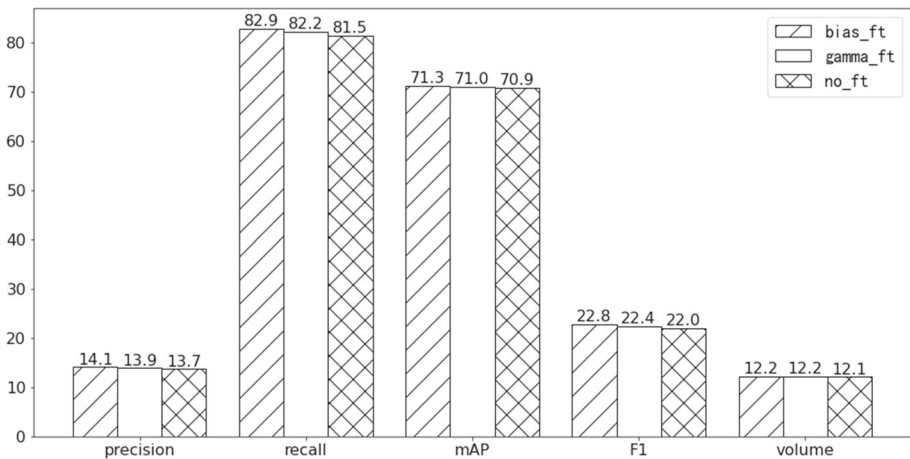
#### 4.6.4 Analysis of the Effect of the Shift Factors of the BN Layer

To prove the role of the shift factor (on pedestrian detection data), we conduct three experiments: (1) The pruning ratio is 95% while retaining 1% of the corresponding channels with the larger shift factors  $\beta$ . After pruning, fine-tune 50 epochs, as shown in Fig. 8 and In Fig. 9 (the ‘/’ filled bar graph); (2) The pruning ratio is 95% while retaining 1% of the corresponding channels with the larger scaling factors  $\gamma$ . After the pruning, fine-tune 50 epochs, as shown in Fig. 8 and In Fig. 9 (the unfilled bar graph); (3) The pruning ratio is 96% but no retention measure. If all the layers need to be pruned, only one corresponding channel with the largest  $\gamma$  is retained. In order to maintain the same pruning ratio as the previous two experiments, we appropriately increased the pruning rate to ensure that the final pruning volume is the same as the previous two pruning. After pruning, we also fine-tuned 50 epochs, as shown in Figs. 8 and 9 (the ‘X’ filled bar graph).

In high-ratio pruning, the amount of model parameters is drastically reduced, and the performance of the model is severely degraded. It must be fine-tuned to restore performance. From Fig. 8, the evaluation metrics of the model without retention measures are slightly higher than the models with the layer retention measure after pruning. But it can be seen from Fig. 9, after fine-tuning, the model that retains the corresponding channels with the larger shift factors  $\beta$  whose evaluation metrics are higher than others. It shows that retaining the corresponding channels with the larger shift factors  $\beta$  enables the model to have a better recovery ability, and it can be restored to a higher performance after fine-tuning.



**Fig. 8** After pruning, the value of each evaluation metrics using different retention measures. The horizontal axis is evaluation metrics, and the vertical axis is the value of each evaluation metrics



**Fig. 9** After fine-tune, the value of each evaluation metrics using different retention measures. The horizontal axis is evaluation metrics, and the vertical axis is the value of each evaluation metrics

## 5 Conclusion

In this paper, we propose a new channel pruning method Multi-Collaboration Channel Pruning (MCCP) to compress the network model. The originality of the MCCP is to consider the weights of the convolutional layer and the two learnable parameters of the BN layer at the same time. A more comprehensive judgment of the importance of channels. Through the comparison of experimental results, we verify the necessity of considering these three influencing factors. Also, compared with other state-of-the-art solutions, we can get higher detection accuracy under the same pruning rate. What's more, we use polarization regularization instead of L1 regularization, which makes it easier to distinguish the importance of model parameters. The experimental results also prove its effect. In future work, we will further compress the model in combination with other compression methods.

**Acknowledgements** This work is supported by Key-Area Research and Development Program of Guangdong Province(2021B0101420002), National Natural Science Foundation of China (62072187,61872084), Guangzhou Science and Technology Program key projects (202007040002), Guangdong Major Project of Basic and Applied Basic Research(2019B030302002), and Guangzhou Development Zone Science and Technology (2020GH10).

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

1. Farhadi A, Redmon J (2018) Yolov3: An incremental improvement. In: Computer Vision and Pattern Recognition. Springer, Berlin/Heidelberg, Germany, pp 1804–2767
2. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv Neural Inf Process Syst* 28:91–99
3. Dai J, Li Y, He K, Sun J (2016) R-fcn: Object detection via region-based fully convolutional networks. In: *Advances in Neural Information Processing Systems*, p 379–387
4. Fan D-P, Lin Z, Zhang Z, Zhu M, Cheng M-M (2020) Rethinking rgb-d salient object detection: Models, data sets, and large-scale benchmarks. *IEEE Transactions on neural networks and learning systems* 32(5):2075–2089
5. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p 770–778
6. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*
7. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p 2818–2826
8. Lu Y, Chen Y, Zhao D, Chen J (2019) Graph-fcn for image semantic segmentation. In: *International Symposium on Neural Networks*, p 97–105. Springer
9. Hong C, Yu J, Wan J, Tao D, Wang M (2015) Multimodal deep autoencoder for human pose recovery. *IEEE Trans Image Process* 24(12):5659–5670
10. Hong C, Yu J, Tao D, Wang M (2014) Image-based three-dimensional human pose recovery by multiview locality-sensitive sparse retrieval. *IEEE Trans Industr Electron* 62(6):3742–3751
11. Qiu T, Chi J, Zhou X, Ning Z, Atiquzzaman M, Wu DO (2020) Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Communications Surveys & Tutorials* 22(4):2462–2488
12. Kiran BR, Sobh I, Talpaert V, Mannion P, Al Sallab AA, Yogamani S, Pérez P (2021) Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*
13. Xu Y, Li Y, Zhang S, Wen W, Wang B, Qi Y, Chen Y, Lin W, Xiong H (2020) Trp: Trained rank pruning for efficient deep neural networks. *arXiv preprint arXiv:2004.14566*
14. Wu S, Li G, Chen F, Shi L (2018) Training and inference with integers in deep neural networks. *arXiv preprint arXiv:1802.04680*
15. Kryzhanovskiy V, Balitskiy G, Kozyrskiy N, Zuruev A (2021) Qpp: Real-time quantization parameter prediction for deep neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p 10684–10692
16. Putra RVW, Shafique M (2021) Q-spinn: A framework for quantizing spiking neural networks. In: *2021 International Joint Conference on Neural Networks (IJCNN)*, p 1–8. IEEE
17. Fang J, Shafiee A, Abdel-Aziz H, Thorsley D, Georgiadis G, Hassoun JH (2020) Post-training piecewise linear quantization for deep neural networks. In: *European Conference on Computer Vision*, p 69–86. Springer
18. Liu Y, Shu C, Wang J, Shen C (2020) Structured knowledge distillation for dense prediction. *IEEE transactions on pattern analysis and machine intelligence*

19. Han S, Pool J, Tran J, Dally WJ (2015) Learning both weights and connections for efficient neural networks. arXiv preprint [arXiv:1506.02626](https://arxiv.org/abs/1506.02626)
20. He Y, Kang G, Dong X, Fu Y, Yang Y (2018) Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint [arXiv:1808.06866](https://arxiv.org/abs/1808.06866)
21. Tan CMJ, Motani M (2020) Dropnet: Reducing neural network complexity via iterative pruning. In: International Conference on Machine Learning, p 9356–9366. PMLR
22. Molchanov P, Tyree S, Karras T, Aila T, Kautz J (2016) Pruning convolutional neural networks for resource efficient inference. arXiv preprint [arXiv:1611.06440](https://arxiv.org/abs/1611.06440)
23. Yu R, Li A, Chen C-F, Lai J-H, Morariu VI, Han X, Gao M, Lin C-Y, Davis LS (2018) Nisp: Pruning networks using neuron importance score propagation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, p 9194–9203
24. Chen L, Chen Y, Xi J, Le X (2021) Knowledge from the original network: restore a better pruned network with knowledge distillation. *Complex & Intelligent Systems*, 1–10
25. Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2016) Pruning filters for efficient convnets. arXiv preprint [arXiv:1608.08710](https://arxiv.org/abs/1608.08710)
26. Zhang P, Zhong Y, Li X (2019) Slimyolov3: Narrower, faster and better for real-time uav applications. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, p 0–0
27. Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C (2017) Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE International Conference on Computer Vision, p 2736–2744
28. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, p 448–456. PMLR
29. Chen Y, Li R, Li R (2021) Hrcp: High-ratio channel pruning for real-time object detection on resource-limited platform. *Neurocomputing* 463:155–167
30. Zhao C, Ni B, Zhang J, Zhao Q, Zhang W, Tian Q (2019) Variational convolutional neural network pruning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2780–2789
31. Chen Y, Wen X, Zhang Y, Shi W (2021) Ccprune: Collaborative channel pruning for learning compact convolutional networks. *Neurocomputing* 451:35–45
32. Guo Y, Yao A, Chen Y (2016) Dynamic network surgery for efficient dnns. arXiv preprint [arXiv:1608.04493](https://arxiv.org/abs/1608.04493)
33. Jia H, Xiang X, Fan D, Huang M, Sun C, He Y (2020) Stochastic model pruning via weight dropping away and back. In: 2020 2nd International Conference on Image Processing and Machine Vision, p 1–9
34. Han S, Mao H, Dally WJ (2015) Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint [arXiv:1510.00149](https://arxiv.org/abs/1510.00149)
35. Lin C, Zhong Z, Wu W, Yan J (2018) Synaptic strength for convolutional neural network. arXiv preprint [arXiv:1811.02454](https://arxiv.org/abs/1811.02454)
36. Hu H, Peng R, Tai Y-W, Tang C-K (2016) Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint [arXiv:1607.03250](https://arxiv.org/abs/1607.03250)
37. He Y, Liu P, Wang Z, Hu Z, Yang Y (2019) Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, p 4340–4349
38. Lin M, Ji R, Wang Y, Zhang Y, Zhang B, Tian Y, Shao L (2020) Hrank: Filter pruning using high-rank feature map. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, p 1529–1538
39. Lin S, Ji R, Yan C, Zhang B, Cao L, Ye Q, Huang F, Doermann D (2019) Towards optimal structured cnn pruning via generative adversarial learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, p 2790–2799
40. Zhuang T, Zhang Z, Huang Y, Zeng X, Shuang K, Li X (2020) Neuron-level structured pruning using polarization regularizer. In: *NeurIPS*
41. Mittal A, Zisserman A, Torr PH (2011) Hand detection using multiple proposals, vol 2. In: *Bmvc*. Citeseer, p 5
42. Bochkovskiy A, Wang C-Y, Liao H-YM (2020) Yolov4: Optimal speed and accuracy of object detection. arXiv preprint [arXiv:2004.10934](https://arxiv.org/abs/2004.10934)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.