Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 3278235, 11 pages
https://doi.org/10.1155/2020/3278235

Research Article

Channel Compression Optimization Oriented Bus Passenger Object Detection

Shuo Zhang, Yanxia Wu, Chaoguang Men, Ning Ren, and Xiaosong Li

College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China

Correspondence should be addressed to Yanxia Wu; wuyanxia@hrbeu.edu.cn

Received 11 July 2019; Revised 6 January 2020; Accepted 28 January 2020; Published 11 March 2020

Academic Editor: Luis Payá

Bus passenger flow information can facilitate scientific dispatching plans, which is essential to decision making and operation performance evaluation. Real-time acquisition of bus passenger flow information is an indispensable part for bus intellectualization. The method of passenger flow statistics in bus video monitoring scene based on deep convolution neural network can provide rich information for passenger flow statistics. In order to adapt to the real scenario of mobile and embedded devices on buses, and to consider the bandwidth limitation, this paper uses a lightweight network model M_7, which is suitable for the vehicle system. Based on the classic network model tiny YOLO, the model is optimized by a depthwise separable convolution method. The optimized network model M_7 reduces the number of parameters and improves the detection speed, while maintaining a low loss in detection accuracy. As such, the network model M_7 is compressed and further optimized by removing redundant channels. The experimental results show that the detection speed of the network model target recognition after channel compression is 40%, which is faster than the precious channel compression on the premise of ensuring detection.

1. Introduction

With the development of economy and society, urban traffic is increasingly subject to overcrowdedness. Public transport passenger flow has become the most important information about vehicle scheduling in the transport sector. Real-time collection of bus passengers flow data has become the key issue of the intelligent public transportation system. By carrying out statistical analysis on the passenger flow of each bus and each station, accurate reference data can be provided to schedule vehicles, to optimize the allocation of bus resources, to improve the overall running speed of the bus system, and to improve the travel experience of passengers.

Early counting methods are mostly manual, which is not only time-consuming and inefficient but also manpower-wasting and material-consuming, aggravating the operating cost of the transportation system. In recent years, with the continuous upgrading of processor performance, real-time statistical analysis of bus passenger flow based on images has been realized. The technology of feature extraction and matching recognition based on the two-dimensional image has been widely used in passenger flow statistics at home and abroad. The development of the deep convolutional neural network in feature extraction has brought amazing convenience to traffic control under video monitoring. It can train the crowd counting algorithm model under video monitoring scene of bus end-to-end, eliminating the steps of foreground segmentation, artificial design, and feature extraction. The high-level features obtained by multilayer convolution of the convolutional neural network can improve the crowd counting algorithm with higher credibility of population statistics.

Traditional convolutional neural network technology is difficult to deploy for mobile terminals due to its high computational complexity and large model size. In order to solve the above problems, researchers proposed the convolutional neural network should adapt to mobile terminals, with limited computing power and storage space through efficient convolution.

Aiming at the passenger flow statistics of bus mobile embedded system, this paper proposes a novel CNN based on compression optimization of the M_7 network model,

which is based on the lightweight network model tiny YOLO optimized by a depthwise separable convolution method. The network model is studied and optimized from the perspective of channel compression.

Firstly, the amount of computation and parameters are used to determine the convolutional layer to be compressed. And then, the effect of removing channels on detection results is used to evaluate the contribution of channels to detection results. The priority removed channels are determined according to the contribution. Finally, the compression algorithm is used to compress the convolutional layer.

2. Related Works

Early methods of bus passenger flow statistics include technologies such as pressure sensor, infrared sensor, and thermal imager. However, these passenger flow statistics methods have many shortcomings, which cannot effectively adapt to complex and crowded bus occasions. For example, the pressure sensor counts the number of passengers by the size and change of the pedal pressure, which has the problems of easily damaged equipment and high daily maintenance cost. The infrared product cannot solve the ambient occlusion problem and distinguish passengers' movement direction under crowded conditions. That is, it cannot count the passenger flow on and off at the same time. The thermal imager is easily affected by ambient temperature and its price is too high to be widely used in the domestic public transport system. Due to its simplistic description of passenger characteristics, the bus passenger flow statistics system of the above technologies cannot achieve satisfactory results in terms of the crowded and chaotic situation in dealing with the crowded and chaotic situation during the peak period of bus passenger flow, with only an accuracy rate of 60–70%.

In the statistical method of bus passenger flow, some achievements have been made by using the traditional image processing method combined with the feature extracted by manual methods. However, due to the influence of the target's shape, background complexity, illumination, and shadow, it is difficult to design a robust feature artificially while adapting to the complex target detection environment. With the continuous development of computer vision technology, the use of video image processing method for target detection, recognition, and tracking is increasingly mature, with great research achievements made so far. In recent years, with the rapid development of deep learning, compared with traditional image processing and machine learning algorithms, deep learning shows great advantages in big datasets and gradually becomes a research hotspot in the field of intelligent transportation.

Deep learning with the help of convolutional neural network grasps the characteristics of objects autonomously from a large number of data samples, which makes continuous breakthroughs in the field of target detection and brings new solutions to target detection. In the condition of actual bus passenger target detection, the space is narrow and the problem of mutual occlusion between passengers during the peak time is very serious. Therefore, it is impossible to take the passenger as the whole detection object. Compared with the passenger body, the head area of the person is in the highest position, and the head area occupies the most important position when looking down from the top view. It can transform the problem of detecting the passenger target into the problem of detecting the head target. It is as shown in Figure 1.

A large number of passenger head target samples are extracted from bus video, and a model of passenger head target is obtained by using deep network training, and then the target detection algorithm is used to detect and identify the bus passenger head target. The convolutional neural network model is superior in many area experiments, but it is still subject to time and space constraints in practical applications. Deep and large convolutional neural networks models are computationally intensive but still not able to meet the needs of many application occasions in time. In addition, large-scale model parameters also take up a large amount of memory space, which is not applicable for mobile embedded terminals. Therefore, the compression network model is an important research issue on the premise of not affecting the accuracy of the convolutional neural network model.

In recent years, compression methods of convolutional neural network based on structured pruning have been proposed successively, overcoming the unaccelerated problem caused by unstructured sparse connections [1–3]. Its core idea relies on the filter significance criterion (i.e., identify the least important filtering criteria), thereby directly removes the significant filter, and accelerates the network calculation. In 2016, Ref. [4] proposed to add an architectural sparse loss function to the loss function of the traditional depth model and assign 0 to the filter less than a given threshold, to directly remove the entire convolution filter with a value of 0 in the test stage.

As a large number of ReLU nonlinear activation functions exist in the mainstream deep network, the output feature map is highly sparse. Reference [5] takes full advantage of this feature to calculate the nonzero ratio of the output feature map corresponding to each filter, as a criterion to judge whether the filter is important. Reference [6] added the constraints of structured sparseness to the objective function and used forward-backward splitting method to solve the optimization problem of sparse structural restriction and directly decided the number of network nodes and redundant nodes in the training process. Reference [7] added regularization restrictions on filtering channels, filter shapes, and layer depth of deep neural networks into the loss function and learned structured convolution filtering by using structured sparse learning. Besides, the norm value of direct measurement filtering is used to judge the significance of filtering [8]. For example, the least L1 norm of the current layer is removed directly, that is, the feature map of the response is removed, and then the channel number of the convolutional filter of the next layer is reduced responsively. Finally, the recognition accuracy of the removed model is improved through retraining. Taking advantage of this property, Ref. [9]

(a) (b)

Figure 1: The passenger object detection results through bus video.

explores the importance of the input channel of the next layer convolution kernels, instead of considering the current layer filter directly, and establishes an effective channel selection optimization function to remove redundant channels and the corresponding current layer filter. Reference [10] dynamically prunes computation on subset of input channels to reduce the computation cost for CNN. Channel gating identifies regions in the features that contribute less to the classification result and skips the computation on a subset of the input channels for these ineffective regions. Unlike static network pruning, channel gating optimizes CNN inference at runtime by exploiting input-specific characteristics, which allows substantially reducing the computation cost with almost no accuracy loss. Existing pruning methods either train from scratch with sparsity constraints on channels, or minimize the reconstruction error between the pretrained feature maps and the compressed ones. Both strategies suffer from some limitations: the former kind is computationally expensive and difficult to converge, whilst the latter kind optimizes the reconstruction error but ignores the discriminative power of channels. Reference [11] investigates a simple-yet-effective method called discrimination-aware channel pruning (DCP) to choose those channels that really contribute to discriminative power. Reference [12] proposes a new filter pruning strategy based on the geometric median, named FPGM, to accelerate the deep CNNs. This article deems the filter is also a point in Euclidean space and could be calculated according to geometric median to get the "center" of the filter, which is their common properties. If a filter is close to this geometric median, the information of the filter can be considered to overlap with (or even redundant with) other filters, so that the filter can be removed without much impact on the network. After removing it, its function can be replaced by other filters. Thus, a filter evaluation method FPGM independent of norm is obtained, which breaks the limitation of norm evaluation index.

The compression method of deep network based on structured pruning can directly compress the network and accelerate the calculation of the whole network by removing the whole filtering of convolutional layer without introducing additional data type storage.

3. Convolutional Neural Network Model Optimization

In this chapter, the tiny YOLO network convolution structure model is employed as the basis for optimization. Tiny YOLO is a smaller version of YOLO [13], which is more suitable for mobile machine learning and IoT devices. Although the convolutional network model structure based on tiny YOLO can achieve good detection results in target detection of bus passengers, it is still too complex to be used in practical applications for embedded devices with limited resource performance.

A depthwise separable convolution method is used to optimize the network structure at first in this chapter. On the premise of guaranteeing the detection accuracy, parameters of the network model are reduced and the detection speed increased. And then, based on this optimized network model, channels are compressed, and the network model is optimized again by removing redundant channels to further improve the detection speed.

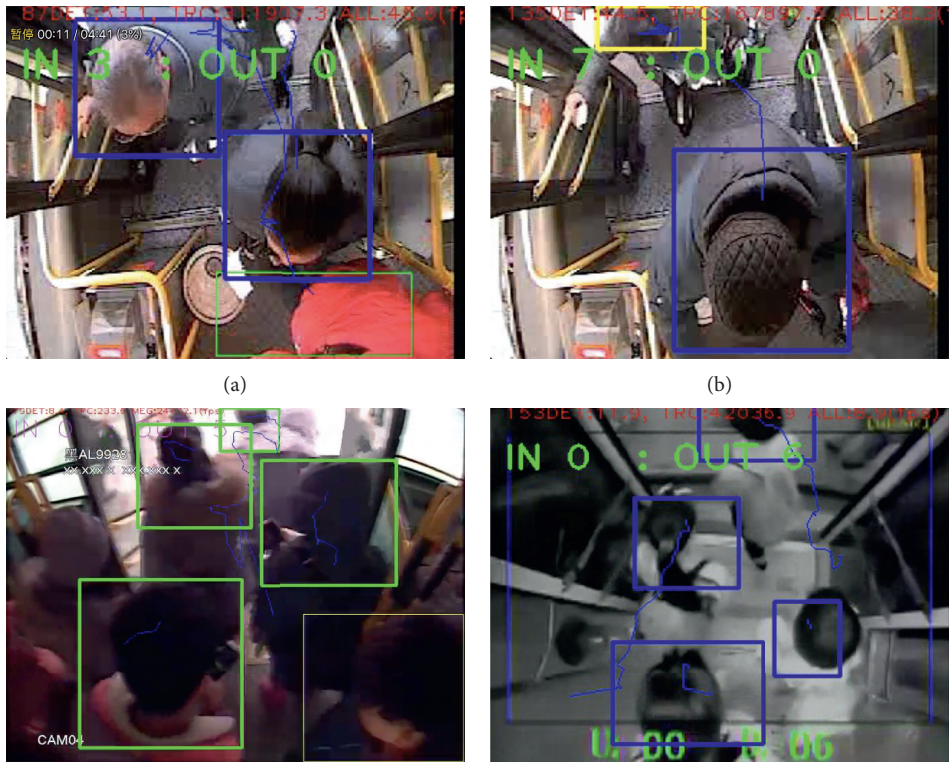
3.1. The Basis of Convolutional Network Model Optimization: Tiny YOLO. Tiny YOLO is a lightweight network model based on the Darknet reference network. The whole network consists of nine convolutional layers, six maximum pooling layers, and one detection layer. The network convolution structure is shown in Table 1 [14].

The performance of tiny YOLO convolution neural network in target detection is tested by using a bus passenger test set containing 12,749 pictures. The selected test samples contain various objective natural factors, such as the brightness of light [15] and the influence of different head shapes of bus passengers on target detection. The detected image is shown as Figure 2.

In this test set, the detection accuracy of tiny YOLO is 0.945 during rush hour. The GNU profiler tool is used to get the function call relationship of the program at the time of

Table 1: Tiny YOLO network structure table. Reproduced from [14].

Layer	Name	Filters	Size	Input	Output
0	conv	16	3 × 3/1	416 × 416 × 3	416 × 416 × 16
1	max		2 × 2/2	416 × 416 × 16	208 × 208 × 16
2	conv	32	3 × 3/1	208 × 208 × 16	208 × 208 × 32
3	max		2 × 2/2	208 × 208 × 32	104 × 104 × 32
4	conv	64	3 × 3/1	104 × 104 × 32	104 × 104 × 64
5	max		2 × 2/2	104 × 104 × 64	52 × 52 × 64
6	conv	128	3 × 3/1	52 × 52 × 64	52 × 52 × 128
7	max		2 × 2/2	52 × 52 × 128	26 × 26 × 128
8	conv	256	3 × 3/1	26 × 26 × 128	26 × 26 × 256
9	max		2 × 2/2	26 × 26 × 256	13 × 13 × 256
10	conv	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
11	max		2 × 2/1	13 × 13 × 512	13 × 13 × 512
12	conv	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024
13	conv	1024	3 × 3/1	13 × 13 × 1024	13 × 13 × 1024
14	conv	30	1 × 1/1	13 × 13 × 1024	13 × 13 × 30
15	Detection				



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

Figure 2: Test set image with various objective factors. Reproduced from [14].

detection. The function call relationship diagram is shown as Figure 3.

It can be seen from Figure 3 that the gemm_nn function responsible for matrix multiplication takes up 86% of the entire program runtime. This function is called nine times, corresponding to each convolution operation of 9 convolutional layers in the network structure. It can be seen that the excessive number of parameters leads to excessive calculation, which is the main reason for time consumption for the detection model of tiny YOLO.

3.2. Depthwise Separable Convolution. Depthwise separable convolution [16] is a lightweight convolution computing technology, which separates conventional convolution in the spatial dimension. This technique not only reduces the computational complexity of convolution operation but also reduces the number of parameters. Therefore, it is a highly efficient convolution method. In this section, the lightweight network model tiny YOLO is taken as the research object to optimize with depth separable convolution method.

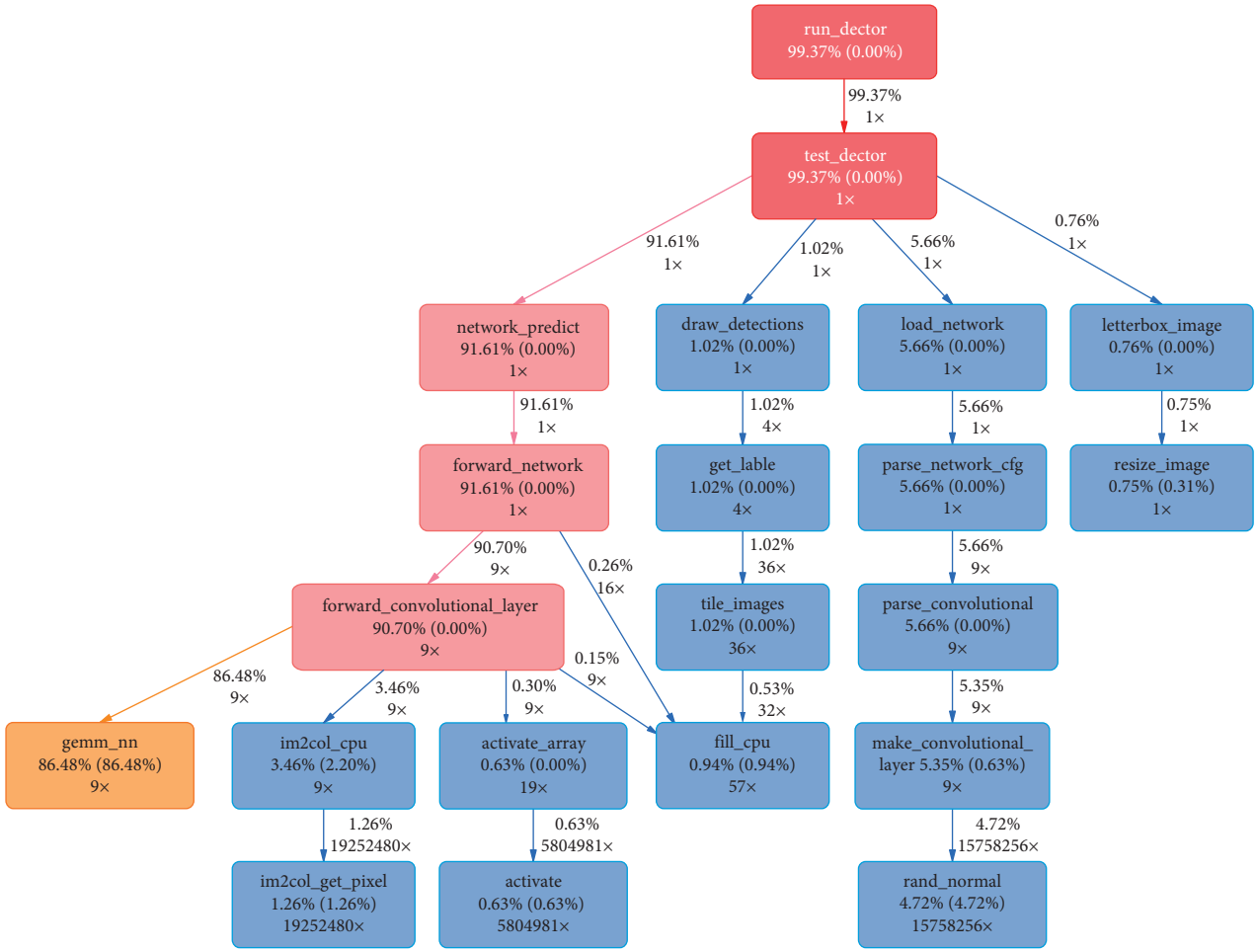


Figure 3: The function call relationship diagram of tiny YOLO in the test set. Reproduced from [14].

Usually, a conventional convolution operation is a three-dimensional convolution kernel that convolves with an input feature map. Each convolution kernel simultaneously operates each channel of the input feature map, and the channel number of the input feature map is consistent with that of the convolution kernel. If the input tensor of the convolution layer l is $x^l \in R^{H^l \times W^l \times D^l}$, the convolution kernel number of the layer is $f^l \in R^{H \times W \times D^l}$. The three-dimensional input convolution operation simply extends the two-dimensional convolution to all the channels of the corresponding position, (i.e., D^l) and finally sums all the HWD^l elements processed by once convolution operation as the result of convolution operation in this position. The specific process is shown in Figure 4.

Depthwise separable convolution is a factorized convolutions operation, which can be decomposed into two smaller operations: depthwise convolution and pointwise convolution. The specific process is shown in Figure 5.

The depthwise separable convolution method divides the original convolutional layer into two convolutional layers. In the depthwise convolution operation, each convolution kernel is convolved with each channel of the input only. The pointwise convolution operation is responsible for feature

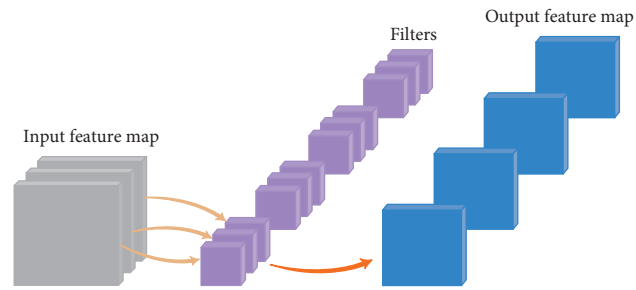


Figure 4: Conventional convolution operation. Reproduced from [14].

fusion that combines the results of the previous convolution. The original three-dimensional convolution kernel is a convolution operation between multiple input channels and convolution kernels to get an output channel. After the improvement, only one channel is required to obtain an output channel, and then a 1×1 convolution is used for channel feature fusion. The advantage of this method is the reduction of parameters and their calculation. The number of parameters of depthwise separable convolution is about 1/3 of the conventional convolution, which can reduce the calculation by about 8 to 9 times.

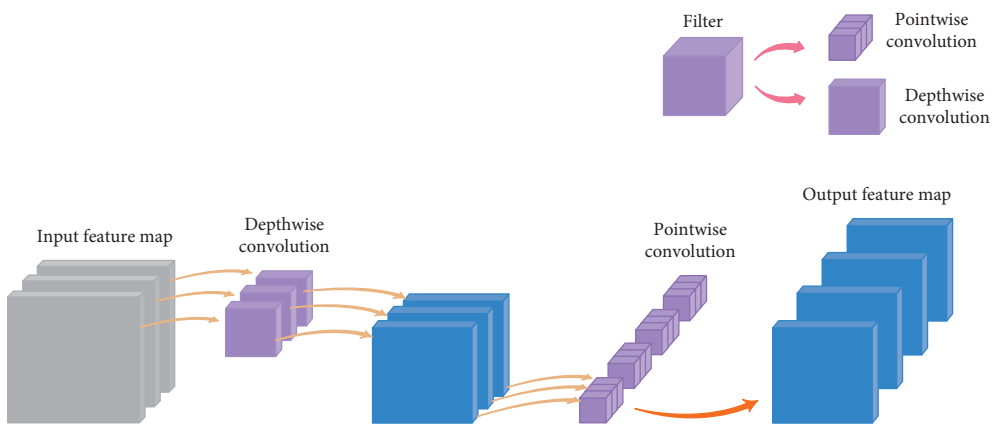


Figure 5: The depthwise separable convolution.

The M_7 network model studied in this paper replaces the original tiny YOLO network structure with a depthwise convolution and pointwise convolution network structure, except for the first and last layer convolution. The network model structure is shown in Table 2.

3.3. Convolutional Neural Network Model Channel Compression

3.3.1. Channel Compression Principle. In the convolutional neural network, channels (also known as feature graphs) are the output results obtained by the convolution operation of the input layer. Figure 6 is a schematic diagram for channel compression in a single convolutional layer. In the figure, A, B, and C contain multiple channels, and W_1 and W_2 contain multiple standard convolution kernels. After convolution operation of A and W_1, the output result B is obtained. According to the characteristics of the convolutional neural network, it can be known that the number of channels in B is consistent with the number of convolutional kernels in W_1 and is also related to the size of all convolutional kernels in W_2. If channels in B are removed (the gap blank part in B), part of the convolution kernel in the previous convolutional layer (the convolution kernel at the dotted line in W_1) shall be removed at the same time, and the size of all the convolution kernel in the latter convolutional layer (the gap blank part of the convolution kernel in W_2) shall be reduced accordingly.

Briefly, in the standard convolutional neural network model, compression channels will affect the number of convolution kernels in the former convolutional layer and the size of the convolution kernels in the latter convolutional layer.

In the lightweight network model M_7, which is based on the tiny YOLO model, the channel compression process is slightly different. Figure 7 is a schematic diagram for channel compression in a single convolutional layer. In the figure, A, B, C, and D contain multiple channels. W_1 is the standard convolution core, W_2 is the depthwise convolution kernel, and W_3 is the pointwise convolution kernel. The number of channels in B is consistent with the number of convolution kernels in W_1 and is also related to the size of convolution

Table 2: M_7 network structure table.

Layer	Name	Filters	Size	Input	Output
0	conv	16	3 × 3/1	416 × 416 × 3	416 × 416 × 16
1	max		2 × 2/2	416 × 416 × 16	208 × 208 × 16
2	conv dw	16	3 × 3/1	208 × 208 × 16	208 × 208 × 16
3	conv	32	1 × 1/1	208 × 208 × 16	208 × 208 × 32
4	max		2 × 2/2	208 × 208 × 32	104 × 104 × 32
5	conv dw	32	3 × 3/1	104 × 104 × 32	104 × 104 × 32
6	conv	64	1 × 1/1	104 × 104 × 32	104 × 104 × 64
7	max		2 × 2/2	104 × 104 × 64	52 × 52 × 64
8	conv dw	64	3 × 3/1	52 × 52 × 64	52 × 52 × 64
9	conv	128	1 × 1/1	52 × 52 × 64	52 × 52 × 128
10	max		2 × 2/2	52 × 52 × 128	26 × 26 × 128
11	conv dw	128	3 × 3/1	26 × 26 × 128	26 × 26 × 128
12	conv	256	1 × 1/1	26 × 26 × 128	26 × 26 × 256
13	max		2 × 2/2	26 × 26 × 256	13 × 13 × 256
14	conv dw	256	3 × 3/1	13 × 13 × 256	13 × 13 × 256
15	conv	512	1 × 1/1	13 × 13 × 256	13 × 13 × 512
16	max		2 × 2/1	13 × 13 × 512	13 × 13 × 512
17	conv dw	512	3 × 3/1	13 × 13 × 512	13 × 13 × 512
18	conv	1024	1 × 1/1	13 × 13 × 512	13 × 13 × 1024
19	conv dw	1024	3 × 3/1	13 × 13 × 1024	13 × 13 × 1024
20	conv	1024	1 × 1/1	13 × 13 × 1024	13 × 13 × 1024
21	conv	30	1 × 1/1	13 × 13 × 1024	13 × 13 × 30
22	Detection				

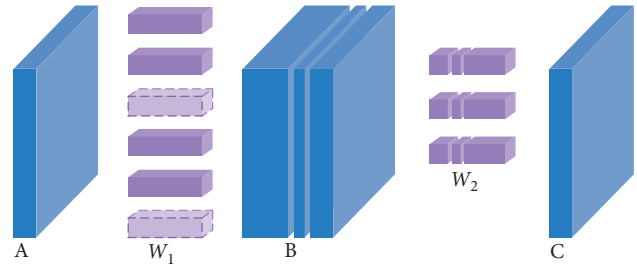


Figure 6: Channels removed diagram.

kernels in W_2 and W_3. Therefore, when the channels in B are compressed (the blank part of the gap in B), part of the convolution kernels in W_1 are removed due to the influence of channel compression (the convolution kernels of the dotted line in W_1), and the size of the convolution kernels in

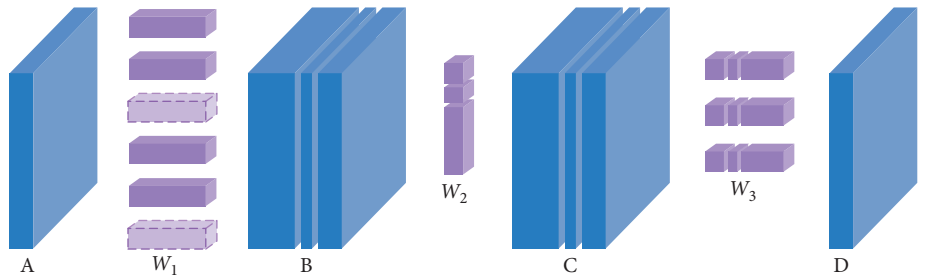


Figure 7: Improved network model channels removed diagram.

W_2 and W_3 is reduced due to the influence of channel compression (the blank part of the convolution kernel in W_2 and W_3).

Briefly, in M_7 network model, compression channel will affect the number of convolution kernels in the former convolutional layer and the size of depthwise convolution kernels and pointwise convolution kernels in the latter convolutional layer.

3.3.2. Channel Compression Process. The process of channel compression in a convolutional neural network is divided into the following five steps to calculate each convolutional layer compressible space, to assess importance of each channel in the convolutional layer, to remove some unimportant channels, to train the compressed channel, and to judge whether to continue the compression.

The channel compression flow chart is shown in Figure 8.

(1) Calculating Each Convolutional Layer Compressible Space. Each convolutional layer is selected with more parameters and floating-point calculations. In the network model M_7, the amount of floating-point computation is related to the number of convolution kernels and the width and height of input channels. Because in the basic operations, the operation time of addition and the number of addition operations are much less than those of multiplication. Only approximate floating-point multiplication computations that need to be performed at this level are considered when computing floating-point computations.

Equation (1) shown as follows is the floating-point computations equation.

$$N_{\text{multi}} = W_{\text{out}} \times H_{\text{out}} \times N_{\text{filters}} \times s \times s \times C_{\text{in}}. \quad (1)$$

In the equation, N_{multi} is the floating-point multiplication computation, W_{out} is the width of the output channel, H_{out} is the height of the output channel, N_{filters} is the number of the convolution kernel, s is the width (height) of the convolution kernel, and C_{in} is number of input channels.

Equation (2) shown as follows is the convolution kernel parameter computations equation.

$$N = N_{\text{filters}} \times s \times s \times C_{\text{in}}. \quad (2)$$

In the equation, N is the amount of convolution kernel parameters, $N_{\text{filters}}$is the floating-point multiplication computation, s is the width (height) of the convolution kernel, and C_{in} is the number of input channels (in the case of 3×3 depth convolution, $C_{\text{in}} = 1$).

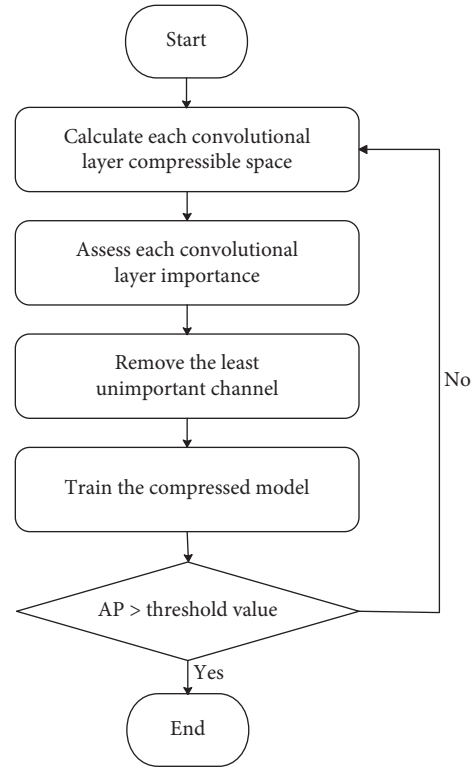


Figure 8: Channels compressed flow chart.

For example, calculate floating-point computations and convolution kernel parameters of 20th layer in M_7 network model. The floating-point computing can be calculated by equation (1): $N_{\text{multi}} = 13 \times 13 \times 1024 \times 1 \times 1 \times 1024 = 177209344$. According to equation (2), the number of convolution kernel parameters is $N = 1024 \times 1 \times 1 \times 1024 = 1048576$.

The number of convolution kernel parameters and floating-point multiplication computation of each layer in network model M_7 are shown in Figure 9.

From Figure 9, it can be seen that, in the M_7, floating-point computation and the number of convolution parameters in convolutional layers 2nd, 5th, 8th, 11th, 14th, 17th, 19th, and 21st account for a very low proportion in their respective total. All of these convolutional layers belong to depthwise convolution. Relatively, floating-point computations and the number of convolution parameters in the remaining convolutional

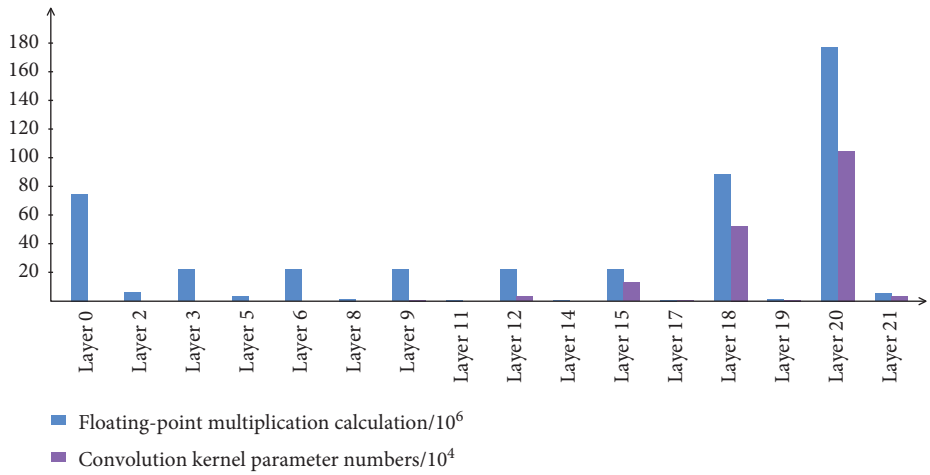


Figure 9: The number of convolution kernel parameters and floating. Point multiplication calculation of each layer in M_7.

layers 0th, 3rd, 6th, 9th, 12th, 15th, 18th, and 20th account for large shares of their total. Most of these convolutional layers fall into the type of pointwise convolutional layer.

Since the operation of the 0th convolutional layer is to input color images of three channels, channel compression cannot be carried out on this layer. Therefore, the pointwise convolutional layer of the 3rd, 6th, 9th, 12th, 15th, 18th, and 20th layer is selected for channel compression analysis, so as to achieve a more obvious optimization effect. Through comprehensive consideration of convolution kernel parameters, floating-point multiplication calculation, together with other factors, channel compression is carried out to the pointwise convolutional layer 20th.

(2) Assessing the Importance of Each Channel in the Convolutional Layer. After the selection of the convolutional layer that must be compressed, the importance of each channel in the layer is evaluated. The detection accuracy difference is used to measure the contribution of the channel to the network model and is the main basis to evaluate the importance of the channel. The difference of detection accuracy refers to the discrepancy between the accurate value of detection output when the input channel number of a convolutional layer is compressed and that of detection when the channel number is not compressed. The difference equation of detection accuracy is shown in equation (3):

$$AP = P - P'. \tag{3}$$

In the equation, AP is the number of convolution kernel parameters, P is the detection accuracy before channel compression, and P' is the detection accuracy after channel compression.

Then, the channels with a small difference of detection accuracy, that is, those with a small contribution to the network model, are sorted in an ascending order.

(3) Channel Compression Algorithm. After obtaining the influence of each channel compression on the detection accuracy, the channel compression is carried out. According to the contribution of each channel to the network model, it is removed in order from small to large. At the same time, it monitors whether the current detection accuracy is lower than the set threshold. When the detection accuracy is less than the set value, the compression will stop immediately and the channel compression algorithm ends. The channel compression algorithm is shown in Algorithm 1.

(1) Initialize: θ_p, s
(2) Compute value P
(3) **while** $\Delta P < \theta_p$ **do**
(4) Updata D_p
(5) $\{s, D_p\} \longrightarrow (I_{\text{channel}}, N)$
(6) $N \longrightarrow C$
(7) Train set $\{I_{\text{channel}}, W_{\text{original}}\} \longrightarrow W_{\text{compressed}}$
(8) Train set $\{C, W_{\text{compressed}}\} \longrightarrow W'$
(9) $\{C, W'\} \longrightarrow P'$
(10) $\Delta P = P - P'$
(11) **end while**

Algorithm 1: Channel compression algorithm.

Table 3: Experimental hardware environment parameter. Reproduced from [14].

Name	Type and configuration parameters
Operation system	Ubuntu 16.0.4
CPU	Intel(R) Core(TM) i7-7700K CPU @ 4.20 GHz
RAM	DDR4 4 ∗ 8G

θ_p represents detection accuracy decreasing threshold. s represents the percentage of removed channels in the total number of channels in the convolutional layer. D_p represents the difference value array of detection accuracy.

In Algorithm 1, the compressed channel index I_{channel} and the number of channels N are computed through compression ratio s and detection accuracy array D_p. Then, the new network configuration file C is generated from N, and the new weight file $W_{\text{compressed}}$ is obtained by using the original network weight W_{original} and I_{channel}. Since channel compression changes the original network structure, the compressed network model is then trained to convergence

Table 4: Hyperparametric configuration table. Reproduced from [14].

Interactive times	Learning rate	Input size	Momentum	Weight decay	Training algorithm	Batch
42000	0.001	416 ∗ 416 ∗ 3	0.9	0.0005	Gradient descent	64

Table 5: Experimental results of network compression iteration.

Compression ratio (%)	Remove channels	AP	Time-consuming (s)	FPS
0	0	0.936	0.225	0.449
5	52	0.937	0.202	4.958
10	102	0.934	0.197	5.084
15	154	0.936	0.194	5.164
20	204	0.932	0.190	5.270
25	256	0.933	0.188	5.321
30	308	0.934	0.185	5.414
35	358	0.930	0.180	5.549
40	410	0.935	0.176	5.667
45	460	0.932	0.174	5.740
50	512	0.934	0.168	5.941
55	564	0.932	0.166	6.024
60	614	0.931	0.162	6.173
65	666	0.849	0.159	6.289

on the training set, and the new weight file W' is obtained. And then the detection accuracy P' of the new weight is computed. Finally, according to the difference of accuracy ΔP it is determined whether to continue compression. When the difference value reaches the set threshold of detection accuracy decreasing, the compression stops. Channel compression algorithm ends.

4. Experiment

This section involves the experiments designed to investigate the optimization performance index of M_7 network model. On this basis, an iterative compression experiment is carried out for the proposed channel compression algorithm. By analyzing the relevant experimental data of each compression, the optimization effectiveness of the channel compression algorithm is verified by measuring the detection speed and accuracy.

4.1. Experimental Parameter Configuration. Table 3 shows the detailed information of the operating system, CPU, and RAM used in this experiment.

The passenger dataset in this paper consists of 12,749 images captured by bus videos from the camera photographing bus passengers vertically. The ratio of the training set and test set in these images is approximately 4 : 1. The datasets are about different weathers in our bus videos, including sunny day, rainy day, snowy day, and so on. As a result of driving the bus in day and night, light condition is a factor that cannot be ignored and often causes errors in the bus passenger object detection. The experiment has prepared enough datasets to take everything into account to ensure the reliability of our data.

The hyperparameters of the network structure (excluding the number of convolution kernels) were taken consistent default settings in the experiments. Some hyperparameters are shown in Table 4.

4.2. Analysis of Experimental Results. In the channel compression algorithm, the values of the θ_p and s are set to 2% and 5%, respectively. The number of channels calculated by the channel compression algorithm N is rounded downward. The network compression iterative experimental results are obtained by the channel compression algorithm. The experimental results are shown in Table 5.

As can be seen from Table 5, in the process of gradually increasing compression ratio, the detection accuracy is always floating around 0.93, while the detection time is gradually decreasing and the detection speed is constantly improving. When the compression ratio of the network model is 65%, the detection accuracy drops sharply and exceeds the threshold value of the detection accuracy difference set by 0.02, stopping the network compression.

The total execution time of the program is 0.225 s in the M_7 network model measured by the GNU profiler tool. Among them, the execution time of the gemm_nn function is 0.129 s, accounting for 57.14% of the total execution time of the program. After optimization by the channel compression algorithm, the deletion parameters account for 62% of the total parameters, and the total execution time of the program is 0.162 s. Among them, the execution time of the gemm_nn function is 0.061 s, accounting for 37.84% of the total execution time of the program. After channel compression, the convolution operation is no longer the bottleneck of time-consuming of the detection program. The number of parameters decreased from 7 MB in M_7 network model to 3.3 MB. Compared with the M_7 network model, the detection speed of the network optimized by the channel compression algorithm is increased by nearly 40% times, while the detection accuracy is only reduced by 5%.

Table 6: Performance comparison between tiny YOLO, M_7, and compressed M_7.

Network	AP	Time-consuming (s)	FPS	Weight size (MB)
Tiny YOLO	0.945	0.915	1.093	60.5
M_7	0.936	0.225	4.449	7
Compressed M_7	0.887	0.162	6.217	3.3

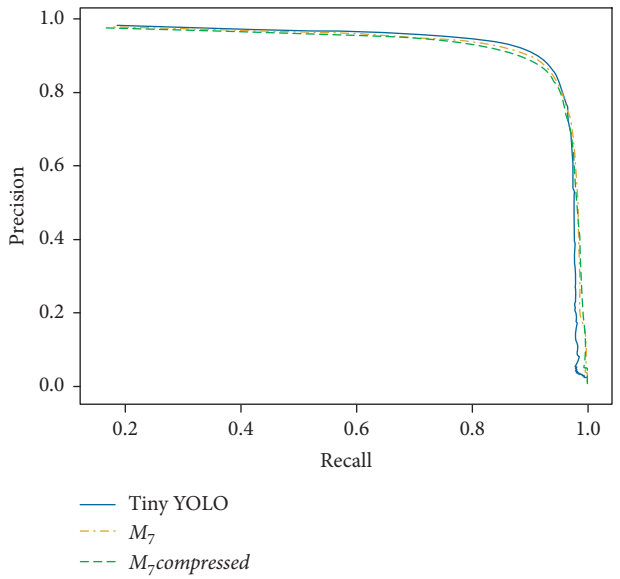


Figure 10: Precision-recall curve of each network model.

In three sets of contrast experiments, this section measures the original network model tiny YOLO, improved network model M_7, and compressed M_7 from the aspects of average precision (AP), time-consuming, frames per second (FPS), and model weight size, as shown in Table 6.

In terms of AP, the values of each network model differ slightly, and the value that varies greatly is just declining by about one percent. In terms of time-consuming detection, the time for detecting an image in the improved network model is reduced from 0.915 s to 0.162 s, compared with the original network model. In terms of FPS, the detection speed increases from 1.093 frames to 6.217 frames, with an obvious improvement in the detection speed. In terms of weight file size, it is reduced from the original 60.5 MB to 3.3 MB.

Analyzed from the perspective of the recall rate accuracy, Figure 10 reveals the precision and recall figure of the tiny YOLO, M_7, and compressed M_7 network structures. In Figure 10, the abscissa is the recall rate and the ordinate is the detection accuracy. From the figure, the detection accuracy of each network model drops significantly when the recall rate exceeds 0.93.

Based on the indicators above, the compressed M_7 model significantly improves the detection speed while reducing the detection time as well as weight file size. In particular, there is little sacrifice in detection accuracy.

5. Conclusion

In this paper, the convolution network model optimization of bus passenger target detection is studied by means of video observation on the basis of a lightweight network model M_7 optimized by depthwise separable convolution method. Firstly, the influence of channel compression on network model is analyzed, and then the channel compression process is described. Secondly, the network parameters and theoretical calculation quantity are analyzed to determine the convolutional layer which needs channel compression. And then the importance of each channel to the whole network model is evaluated according to the difference of its influence on detection accuracy. Finally, combined with the channel compression algorithm, the lightweight network model M_7 is compressed. The experimental results show that when 62% of parameters of the convolutional network are removed by channel compression, the time consumption of convolution operation is reduced by one time, the size of weight file is reduced by nearly one time, and the detection speed is increased by 40%. To sum up, channel compression achieves better compression effect.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D Program of China (2016YFB1000402), the Natural Science Foundation of Heilongjiang Province (F2018008), the Foundation for Distinguished Young Scholars of Harbin (2017RAYXJ016), and the Fundamental Research Funds for the Central Universities (3072019CFT0602).

References

[1] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," 2015, http://arxiv.org/abs/1507.06149.

[2] H. Song, H. Z. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding," 2016, http://arxiv.org/abs/1510.00149v5.

[3] S. Han, J. Pool, J. Tran et al., "Learning both weights and connections for efficient neural network," *Advances in Neural Information Processing Systems*, vol. 1, pp. 1135–1143, 2015.

[4] V. Lebedev and V. Lempitsky, "Fast ConvNets using group-wise brain damage," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2554–2564, Las Vegas, NV, USA, 2016.

[5] H. Li, A. Kadav, I. Durdanovic et al., "Pruning filters for efficient ConvNets," 2017, http://arxiv.org/abs/1608.08710v3.
[6] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: towards compact CNNs," in *Proceedings of the European Conference on Computer Vision*, pp. 662–677, Amsterdam, The Netherlands, 2016.
[7] H. Y. Hu, R. Peng, Y. W. Tai et al., "Network trimming: a data-driven neuron pruning approach towards efficient deep architectures," 2016, http://arxiv.org/abs/1607.03250.
[8] W. Wei, C. P. Wu, Y. D. Wang et al., "Learning structured sparsity in deep neural networks," *Advances in Neural Information Processing Systems*, vol. 29, pp. 2074–2082, 2016.
[9] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: a filter level pruning method for deep neural network compression," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5058–5066, Venice, Italy, 2017.
[10] W. Hua, C. D. Sa, Z. Zhang, and G. E. Suh, "Channel gating neural networks," 2018, http://arxiv.org/abs/1805.12549.
[11] Z. Zhuang, "Discrimination-aware channel pruning for deep neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 875–886, 2018.
[12] Y. He, P. Liu, Z. Wang et al., "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, Long Beach, CA, USA, June 2019.
[13] J. Redmon, "YOLO: real-time object detection," 2016, https://pjreddie.com/darknet/YOLO/.
[14] S. Zhang, Y. Wu, C. Men, and X. Li, "Tiny YOLO optimization oriented bus passenger object detection," *Chinese Journal of Electronics*, vol. 29, no. 1, pp. 132–138, 2020.
[15] G. Chen, S. Krishnan, Y. Zhao, and W. Xie, "Illumination invariant face recognition," *Intelligent Computing Theories*, vol. 7995, pp. 385–391, 2013.
[16] L. Sifre and S. Mallat, *Rigid-motion scattering for image classification*, Ph.D. thesis, Ecole Polytechnique, CMAP, Palaiseau, France, 2014.