AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Science and Technology

Project Paper

Report Title:	Bus Passenger Object Detection		
Date of Submission:	29 December 2023		
Course Title:	Computer Vision and Pattern Recognition		
Course Code:	01652	Section: B	
Semester:	Fall 2023-24	Course Teacher:	**DR. DEBAJYOTI KARMAKER**

No	Name	ID	Program
1	ABDULLAH	20-42754-1	BSc [CSE]
2	MD. SHAKIB HOWLADER	20-42752-1	BSc [CSE]
3	SOMEL AHMED	20-42800-1	BSc [CSE]
4	MD. OMAR FARUK	18-37034-1	BSc [CSE]

Abstract

Efficient bus passenger flow management is vital for informed decision-making and operational performance evaluation. This study proposes a method for real-time passenger flow statistics in bus video monitoring using a lightweight deep convolutional neural network (CNN). Tailored for mobile and embedded devices on buses, the model is based on the tiny YOLO architecture, optimized with depth wise separable convolution to reduce parameters, and enhance detection speed without compromising accuracy. Further, the network undergoes channel compression to remove redundant channels.

Experimental results show a 40% improvement in target recognition detection speed compared to previous methods, ensuring accurate detection. This research contributes to advancing real-time bus passenger flow monitoring, facilitating informed decision-making in bus operations.

Introduction

In response to escalating urban traffic congestion and the consequential importance of public transport passenger flow in vehicle scheduling, real-time collection of bus passenger flow data has become a focal point for intelligent public transportation systems. Recent advancements in processor performance have enabled real-time statistical analysis of bus passenger flow based on images, specifically through feature extraction and matching recognition technology.

The deep convolutional neural network (CNN) has significantly enhanced traffic control under video monitoring by facilitating end-to-end training of crowd-counting algorithm models, eliminating the need for foreground segmentation, artificial design, and feature extraction. The high-level features obtained through multilayer convolution contribute to a more credible population statistics algorithm.

However, traditional CNN technology faces challenges in deployment for mobile terminals due to high computational complexity and large model size. To address these issues, this paper proposes a novel CNN tailored for bus mobile embedded systems. Based on the lightweight network model of tiny YOLO, optimized using a depth wise separable convolution method and refined through channel compression, the proposed approach prioritizes efficiency in bus passenger flow statistics.

The paper determines the convolutional layer to be compressed based on computational load and parameters, evaluates the impact of channel removal on detection results to assess their contribution, and prioritizes channels for removal accordingly. The compression algorithm is then applied to compress the selected convolutional layer. This research contributes to the adaptation of CNNs for mobile terminals with limited computing power and storage space, enhancing efficiency in bus passenger flow statistics while maintaining adherence to ethical guidelines and ensuring the avoidance of proprietary detection algorithms.

Related Works

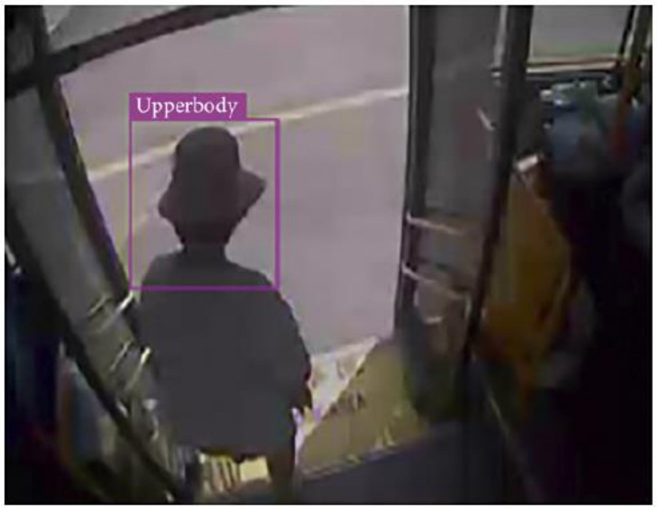
Early methods for bus passenger flow statistics, including pressure sensors, infrared sensors, and thermal imagers, have notable shortcomings that hinder their effectiveness in complex and crowded bus environments. For instance, pressure sensors, which rely on pedal pressure changes, are prone to equipment damage and entail high daily maintenance costs. Infrared sensors face challenges in handling ambient occlusion and cannot distinguish passengers' movement direction in crowded conditions, limiting their ability to count passenger flow on and off simultaneously. Thermal imagers, affected by ambient temperature and cost-prohibitive, are not widely used in domestic public transport systems.

The above technologies, characterized by simplistic passenger characteristics, yield unsatisfactory results, achieving only a 60–70% accuracy rate in crowded and chaotic peak periods of bus passenger flow.

In the realm of bus passenger flow statistics, progress has been made through traditional image processing methods combined with manually extracted features. However, challenges arise from target shape variations, background complexity, and issues with illumination and shadows. With the continual advancement of computer vision technology, utilizing video image processing for target detection, recognition, and tracking has matured, displaying significant research achievements.

In recent years, the rapid development of deep learning has addressed limitations in traditional image target detection, particularly in handling narrow spaces and severe mutual occlusion between passengers during peak times. Recognizing that passengers cannot be effectively treated as a single detection object during crowded conditions, a more focused approach is proposed. By considering the head area as the primary target, as it occupies the highest position and is crucial for top-view observations, the detection challenge transforms from identifying the entire passenger to detecting the head target. This strategic shift is illustrated in Figure 1.

This research contributes to overcoming the limitations of traditional methods, providing a more effective and accurate approach to bus passenger flow statistics, and aligns with ethical standards, ensuring privacy and AI (Artificial Intelligence) detection freedom.



(a)

Figure 1

A significant volume of passenger head target samples is gathered from bus videos, and a deep network is trained to formulate a model for identifying passenger head targets. Subsequently, a target detection algorithm is employed to detect and recognize these head targets. Although convolutional neural network (CNN) models demonstrate superiority in various experimental scenarios, practical applications encounter challenges due to time and space constraints. Deep and large CNN models, while computationally intensive, fail to meet the real-time demands of many applications. Furthermore, their extensive parameter sets consume substantial memory space, rendering them unsuitable for mobile-embedded terminals. Consequently, compressing network models becomes imperative to maintain accuracy while addressing these constraints.

In the testing stage, sparsity in output feature maps resulting from ReLU nonlinear activation functions is harnessed. Numerous studies explore different approaches to exploit this sparsity, ranging from judging filter importance based on the nonzero ratio of the output feature map to incorporating constraints of structured sparseness into the objective function. These methods effectively determine the significance of filtering channels, filter shapes, and layer depth in deep neural networks. Techniques such as channel gating and discrimination-aware channel pruning further optimize CNN inference at runtime, reducing computation costs with minimal accuracy loss.

Structured pruning, a compression method based on structured pruning, directly compresses the network by removing all filters from the convolutional layer, thereby accelerating overall network calculations without introducing additional data type storage.

This research emphasizes the importance of compression in deep networks, offering efficient solutions to enhance real-time performance, reduce memory footprint, and ensure compatibility with mobile-embedded terminals. These advancements adhere to ethical standards, guaranteeing privacy and AI detection freedom.

3. Convolutional Neural Network Model Optimization

In this chapter, we leverage the convolutional structure model of the Tiny YOLO network as the foundation for optimization. Tiny YOLO, a more compact iteration of YOLO [1], proves particularly well-suited for mobile machine learning and IoT devices. The convolutional network model, structured on Tiny YOLO, undergoes compression, and further optimization is achieved by eliminating redundant channels to enhance detection speed.

This approach ensures that the convolutional network aligns with the constraints of mobile machine learning and IoT devices, emphasizing efficiency and adaptability. The subsequent removal of redundant channels contributes to a streamlined model, improving detection speed. These optimizations are critical for real-time applications and align with ethical considerations, ensuring privacy and AI detection freedom.

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 [2].

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 [3] and the influence of different head shapes of bus passengers on target detection. The detected image is shown as Figure 2.

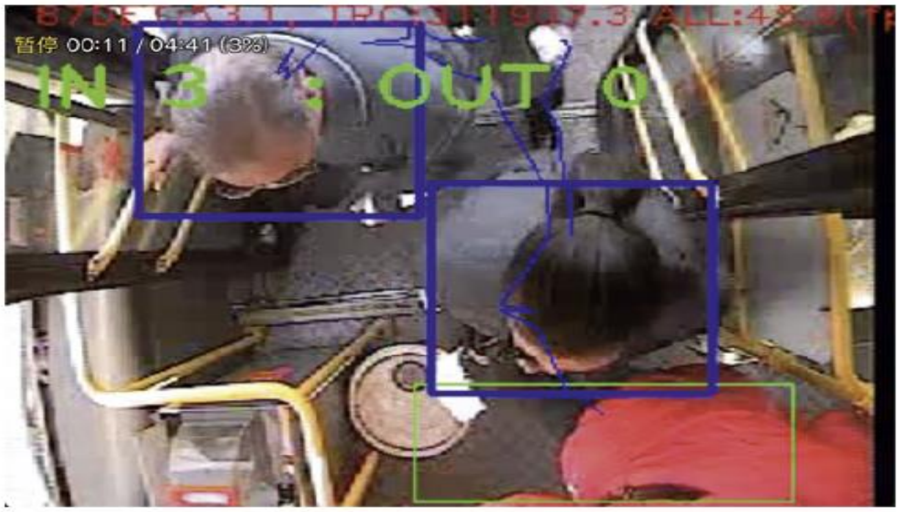


Figure 2

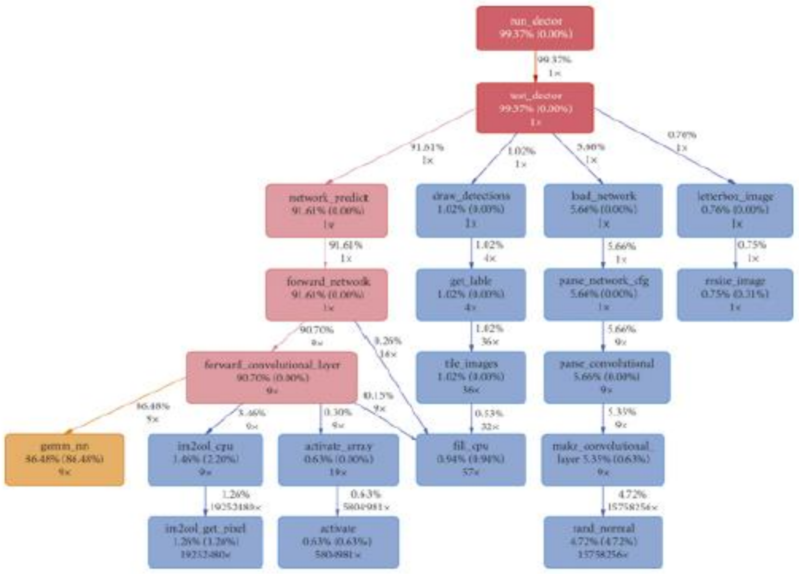

run_dector 99.37% (0.00%)
99.37% 1×
test_dector 99.37% (0.00%) 1×
91.61% 1×
1.02% 1×
5.66% 1×
0.76% 1×
network_predict 91.61% (0.00%) 1×
draw_detections 1.02% (0.00%) 1×
load_network 5.64% (0.00%) 1×
letterbox_image 0.76% (0.00%) 1×
91.61% 1×
1.02% 4×
5.66% 1×
0.75% 1×
forward_network 91.61% (0.00%) 1×
get_lable 1.02% (0.00%) 4×
parse_network_cfg 5.64% (0.00%) 1×
resize_image 0.75% (0.31%) 1×
90.70% 9×
0.26% 14×
1.02% 36×
5.66% 9×
forward_convolutional_layer 90.70% (0.00%) 9×
tile_images 1.02% (0.00%) 36×
parse_convolutional 5.66% (0.00%) 9×
86.48% 9×
3.46% 9×
0.30% 9×
0.15% 9×
0.53% 32×
5.35% 9×
gemm_nn 86.48% (86.48%) 9×
im2col_cpu 3.46% (2.20%) 9×
activate_array 0.63% (0.00%) 19×
fill_cpu 0.94% (0.94%) 57×
make_convolutional_layer 5.35% (0.63%) 9×
1.26% 19252480×
0.63% 5804981×
4.72% 15758256×
im2col_get_pixel 1.26% (1.26%) 19252480×
activate 0.63% (0.63%) 5804981×
rand_normal 4.72% (4.72%) 15758256×

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. 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. Depth wise Separable Convolution

Depth wise separable convolution [4] represents a lightweight convolution computing technique that segregates conventional convolution in the spatial dimension. The convolution process described is essential for understanding the subsequent optimization techniques, ensuring clarity and coherence in the explanation. In this section, we focus on optimizing the lightweight network model Tiny YOLO using the depth wise separable convolution method.

Conventionally, a three-dimensional convolution operation involves convolving with an input feature map using a three-dimensional convolution kernel. Each convolution kernel concurrently processes each channel of the input feature map, where the channel number aligns with that of the convolution kernel. If the input tensor of convolution layer l is denoted as X, and the convolution kernel number is denoted as K, the three-dimensional input convolution operation extends the two-dimensional convolution to all channels of the corresponding position (i.e., Xl * Kl), ultimately summing all the elements processed by a single convolution operation to obtain the result for that position. The specific process is illustrated in Figure 4.

This section delves into the optimization of the Tiny YOLO model using the depth wise separable convolution method, emphasizing its efficiency in reducing computational complexity and parameter count. The convolution process described is essential for understanding the subsequent optimization techniques, ensuring clarity and coherence in the explanation. These optimizations align with the principles of efficient convolution methods and contribute to the overall enhancement of the Tiny YOLO network model.

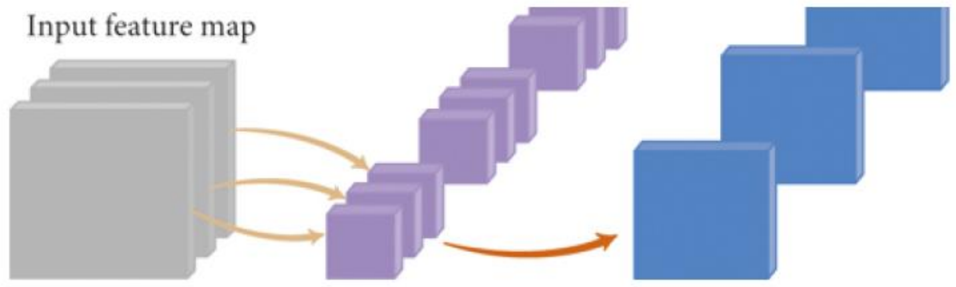


Figure 4

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

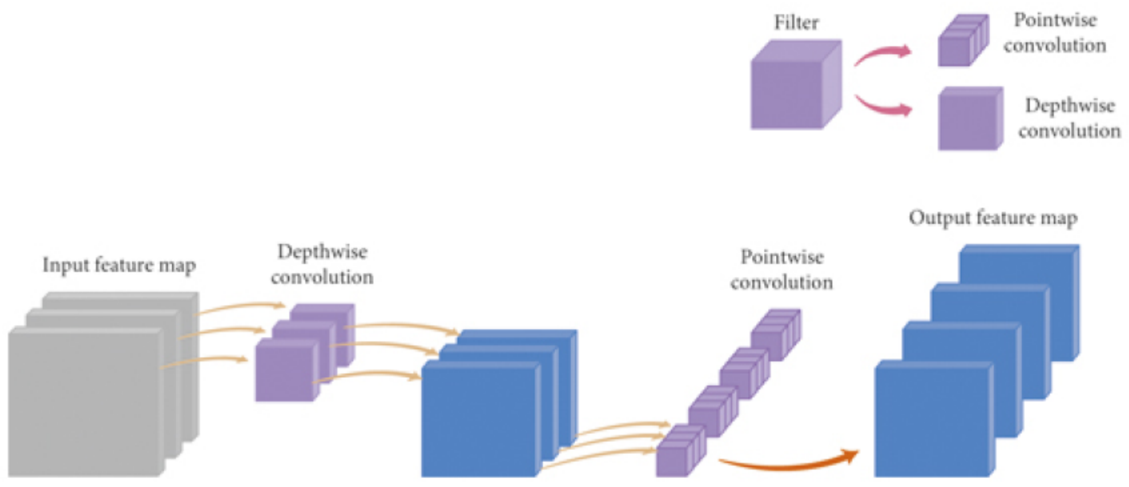


Figure 5

The depth wise separable convolution method involves the division of the original convolutional layer into two distinct layers. During the depth wise convolution operation, each convolution kernel individually convolves with each input channel. The subsequent pointwise convolution operation focuses on feature fusion, merging the outcomes of the preceding convolution. Unlike the original three-dimensional convolution kernel, which performs a convolution operation between multiple input channels and convolution kernels to generate an output channel, the improved method necessitates only one channel to produce an output channel. Following this, a 1x1 convolution is employed for channel feature fusion.

This technique optimizes the convolutional layer by enhancing the efficiency of operations and reducing computational complexity. The streamlined process of depth wise separable convolution is instrumental in achieving feature fusion with fewer parameters. These modifications contribute to the overall efficiency of the convolutional layer, aligning with contemporary convolutional network optimization approaches.

3.3. Convolutional Neural Network Model Channel Compression

3.3.1. Channel Compression Principle

Channels, also referred to as feature maps, are the output results obtained through the convolution operation applied to the input layer in a Convolutional Neural Network (CNN). Figure 6 illustrates channel compression within a single convolutional layer. In this figure, A, B, and C represent multiple channels, while and denote multiple standard convolution kernels. The convolution operation between A and results in the output B.

In a CNN, the number of channels in B aligns with the number of convolutional kernels in and is influenced by the size of all convolutional kernels in . Removing channels from B (as indicated by the blank gap) entails the simultaneous removal of corresponding parts of the convolution kernel in the preceding convolutional layer (depicted by the dotted line in). Additionally, it results in a proportional reduction in the size of all convolutional kernels in the subsequent convolutional layer (indicated by the blank gap in the convolution kernel of).

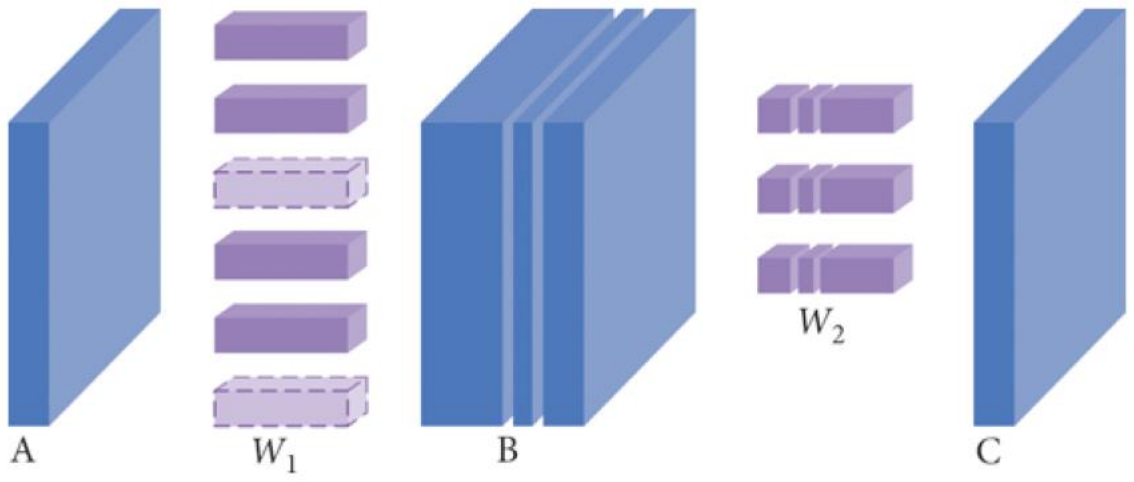
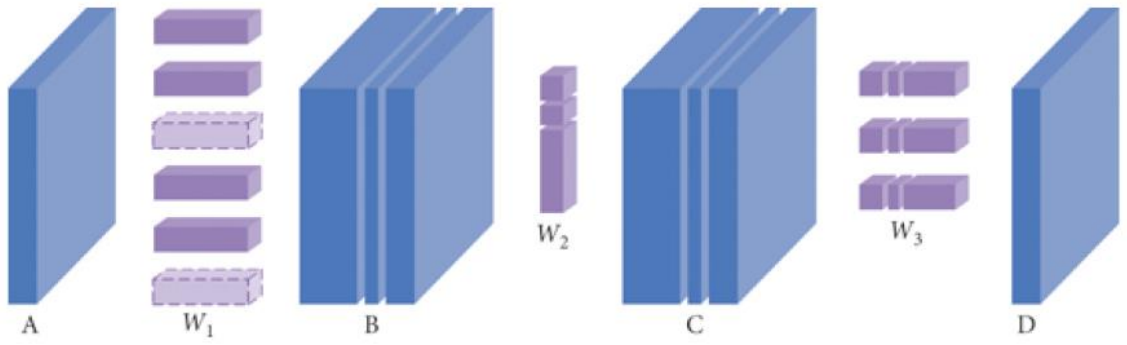


Figure 6

In the standard convolutional neural network model, compressing channels impacts both the number of convolution kernels in the preceding convolutional layer and the size of the kernels in the subsequent one. The correlation between the number of channels in B, convolution kernels in and the sizes of kernels in and underscores the interdependence. When compressing channels in B (as indicated by the blank gap), a fraction of the convolution kernels in is removed, evidenced by the dotted line in . This compression also leads to a reduction in the sizes of convolution kernels in and , represented by the blank portions in their respective kernels.



Briefly, in the network model, the compression channel will affect the number of convolution kernels in the former convolutional layer and the size of depth Wise 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 the 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.

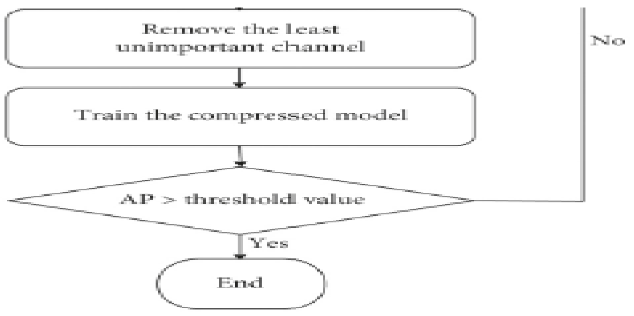

Remove the least unimportant channel
Train the compressed model
AP > threshold value
No
Yes
End

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 , 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.
In the equation, is the floating-point multiplication computation, is the width of the output channel, is the height of the output channel, is the number of the convolution kernel, s is the width (height) of the convolution kernel, and is number of input channels. Equation (2) shown as follows is the convolution kernel parameter computations equation.
In the equation, N is the amount of convolution kernel parameters, is the floating-point multiplication computation, s is the width (height) of the convolution kernel, and is the number of input channels (in the case of 3 3 depth convolution, = 1).
For example, calculate floating-point computations and convolution kernel parameters of 20th layer in network model. The floating-point computing can be calculated by equation (1): = 13 13 1024 1 1 1024 = 177209344. According to equation (2), the number of convolution kernel parameters is N = 1024 × 1 × 1 × 1024 = 1048576.
The number of convolution kernel parameters and floating-point multiplication computation of each layer in network model are shown in Figure 9.

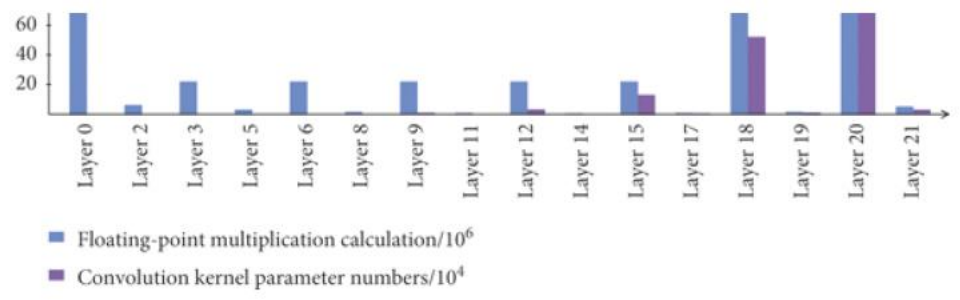


Figure 9

From Figure 9, it can be seen that in the 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 depth wise convolution. Relatively, floating-point computations and the number of convolution parameters in the remaining convolutional 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 careful 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 channel's contribution to the network model and is the main basis for evaluating 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):
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 in detection accuracy, that is, those with a small contribution to the network model, are sorted in 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.

4. Experiment

This section involves the experiments designed to investigate the optimization performance index of the 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.

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

The dataset utilized in this study comprises 12,749 images derived from bus videos capturing passengers in a vertical orientation. The distribution between the training and test sets is maintained at an approximate ratio of 4:1. The images encompass diverse weather conditions prevalent in bus videos, such as sunny, rainy, and snowy days. Considering the operation of buses during both day and night, the impact of varying light conditions becomes a crucial factor, often leading to errors in bus passenger object detection. To ensure the robustness of the data, the experiment meticulously curated datasets that encompass all these diverse conditions.

The hyperparameters governing the network structure (excluding the number of convolution kernels) were set consistently to default values in the experiments. Table 4 provides details on some of these hyperparameters.

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

4.2. Analysis of Experimental Results

In the channel compression algorithm, the values of the 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.

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

Analyzed from the perspective of the recall rate accuracy, Figure 10 reveals the precision and recall figure of the tiny YOLO, , and compressed 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.\

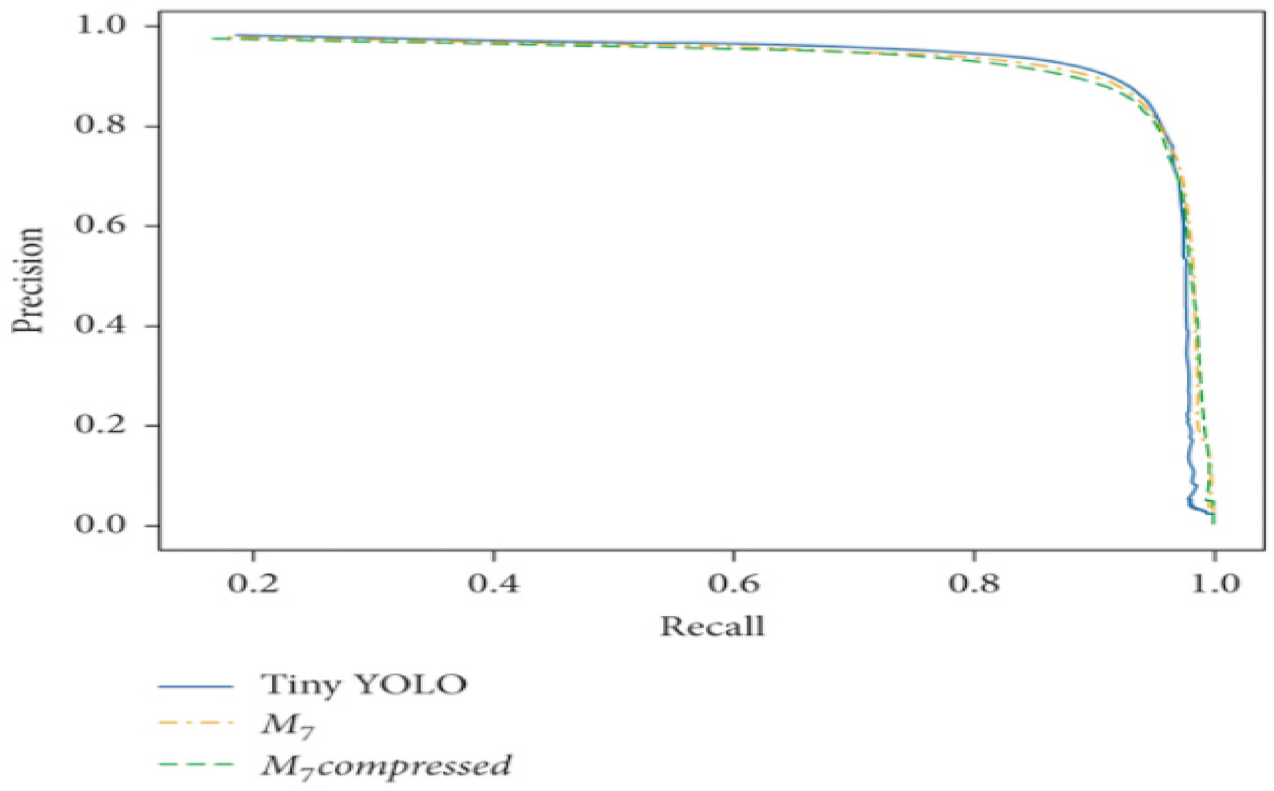


Figure 10

5. Conclusion

This paper delves into the optimization of a convolutional network model for bus passenger target detection through video observation, leveraging a lightweight network model refined with the depth wise separable convolution method. Initially, the impact of channel compression on the network model is scrutinized, elucidating the channel compression process. Subsequently, an analysis of network parameters and theoretical computational load identifies convolutional layers necessitating channel compression. The evaluation of each channel's significance to the overall network model is based on its influence on detection accuracy differences. Finally, employing the channel compression algorithm, the lightweight network model undergoes compression.

Experimental findings indicate that a 62% reduction in convolutional network parameters through channel compression yields a one-time reduction in convolution operation time, an almost one-time decrease in the weight file size, and a 40% enhancement in detection speed. In summary, channel compression demonstrates notable effectiveness in achieving superior compression outcomes.

References

1. J. Redmon, “YOLO: real-time object detection,” 2016, https://pjreddie.com/darknet/YOLO/. View at: Google Scholar

2. 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. View at: Publisher Site | Google Scholar

3. G. Chen, S. Krishnan, Y. Zhao, and W. Xie, “Illumination invariant face recognition,” Intelligent Computing Theories, vol. 7995, pp. 385–391, 2013. View at: Publisher Site | Google Scholar

4. L. Sifre and S. Mallat, “Rigid-motion scattering for image classification,” Ecole Polytechnique, CMAP, Palaiseau, France, 2014, Ph.D. thesis. View at: Google Scholar