Deep Learning-Based Lightweight Target Detection

Zuyi Jin *

School of Electronic and Electrical Engineering, University of Newcastle, Australia, Singapore

Abstract. With the rapid development of computer vision technology, target detection techniques have also been widely used in various fields. However, traditional target detection methods usually require a large amount of computational resources and complex model structures, which pose challenges to the realization and deployment of target detection. Therefore, lightweight target detection methods arise. Lightweight target detection methods refer to reducing the computational and parametric quantities of the model as much as possible while keeping the detection accuracy unchanged. The emergence of this method can not only improve the speed and efficiency of target detection, but also make the target detection technology more adaptable to embedded devices and other resource-limited scenarios. In this paper, through the summary of the deep learning method, the impact on the accuracy and the reduction of computation is compared to give a reference to the subsequent embedded systems or small computational amount of system lightweighting.

Keywords: shufflenet, mobilenet, senet.

1. Introduction

In recent years, with the rapid development of computer technology, computer vision technology has become an indispensable part of human life. Among them, target detection, as a research hotspot in the field of deep learning and computer vision, is widely used in several fields, such as security prevention and control, enterprise business and automatic driving. Target detection algorithms analyze target objects in images or videos to determine whether they exist or not and provide location and category information. In the field of security prevention and control, target detection can help intelligent surveillance devices to perform human shape recognition and motion detection, and accurately locate the target position and category in real time. In the field of enterprise business, target detection technology can intelligently detect and identify documents such as ID cards and financial statements, improving work efficiency and reducing labor costs [1]. In addition, target detection technology can also analyze and identify human behavior, helping companies to monitor employee behavior. In autonomous driving technology, target detection technology can help cars intelligently recognize vehicles, pedestrians and other objects in front of them to achieve accurate and timely operation. In addition to the above application scenarios, target detection technology has a wide range of applications in various industries such as healthcare, education, and aviation. However, target detection algorithms face many challenges, such as the effects of a wide variety of objects, target morphology, movement and overlap, as well as difficulties in size, motion changes, light changes, complex backgrounds and object occlusion. In order to solve these problems, scholars have devoted themselves to researching target detection techniques, among which lightweight network design is an important research direction. Lightweight network refers to the network with smaller number of parameters and computation, and its training-generated model occupies less storage space, which greatly reduces the computational burden [2].

Lightweight network design in target detection primarily strives to maintain an optimal trade-off between detection accuracy and computational efficiency. This is particularly critical in real-time applications such as autonomous driving and video surveillance, where the delay caused by heavy computations can be detrimental.

Further, the development and optimization of lightweight networks have opened the door to deploy target detection algorithms on edge devices. These devices, such as drones, mobile phones, and IoT devices, often have limited computational resources and storage. With lightweight networks, they can perform complex target detection tasks locally without the need to transfer data to the cloud, thereby reducing latency and improving user privacy.

On the other hand, designing an effective lightweight network for target detection is a challenging task. It requires a fine balance in reducing the complexity of the network while retaining sufficient expressive power for accurate object detection. Moreover, the diversity and complexity of real-world scenarios add to the difficulty of the task. Current research efforts are focused on developing more efficient network architectures, as well as better training and optimization techniques, to enhance the performance of lightweight networks in target detection.

2. Lightweighting Method Based on Network Structure

2.1. ShuffleNet

A lightweight convolutional neural network that is characterized by a channel disruption before each grouped convolutional layer, allowing the output channels to interact with all the channels of the input, thus improving the efficiency of information transfer and enhancing the expressive power of the network. As shown in the Figure 1, figure a is the result of ordinary grouped convolution, which has the disadvantage that the output channels are only associated with some of the input channels, resulting in insufficient information flow and poor network performance. Figure b is the structure of ShuffleNet, i.e., the feature maps after grouped convolution are mixed by channel by random arrangement, so that the global information can be better utilized. Figure c shows the equivalent schematic after the operation [3]. Since the number of channels acting on each convolution kernel is reduced when grouped convolution is performed, the amount of computation can be drastically reduced.

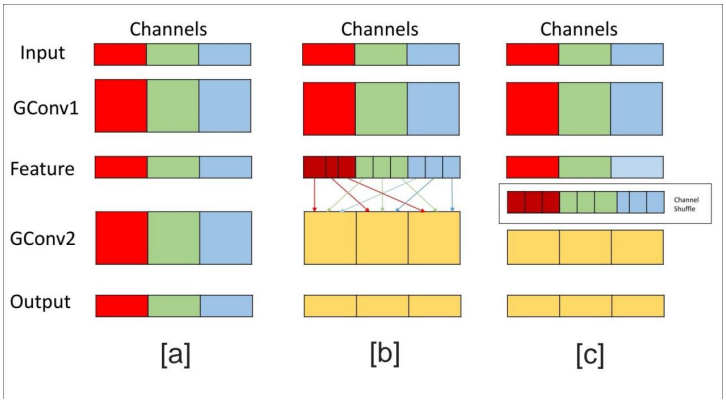


Fig. 1 Shufflenet structure [3]

2.2. MobileNet

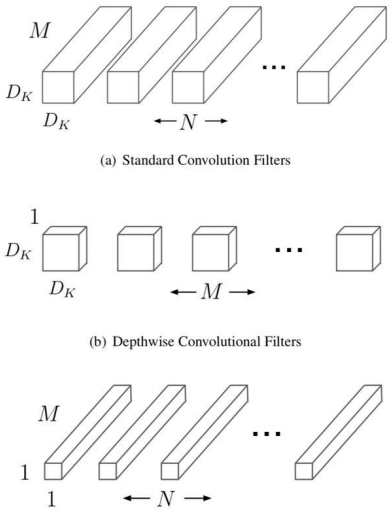


Fig. 2 Mobile net structure [4]

As shown Figure 2 above, Mobilenet is a lightweight convolutional neural network that is characterized by the use of the idea of depth-separable convolution, which decomposes the convolution operation into a combination of depth convolution (or channel convolution) and 1x1 point convolution [4]. Where depth separable convolution is the convolution of each channel individually, equivalent to a grouped convolution with only one channel in each group, and then using the lower overhead 1×1 convolution for channel merging, which can greatly reduce the amount of computation.

2.3. Senet

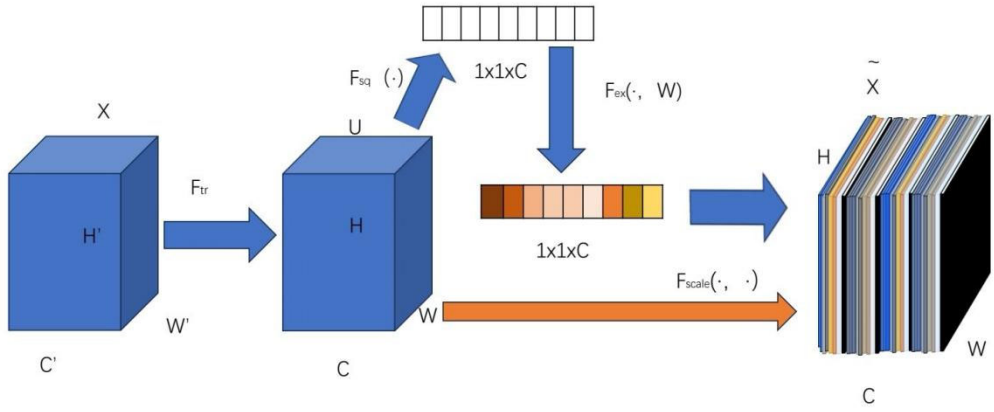


Fig. 3 Senet structure [5]

As shown Figure 3 above. SEnet is a convolutional neural network that is characterized by the introduction of a mechanism to assign weights to each channel. A set of features is output in the previous layer, at this point there are two paths, the first passes through directly, and the second first performs a Squeeze operation (global average pooling), which compresses each channel's two-dimensional features into a one-dimensional one, thus obtaining a vector of feature channels (each number represents a feature of the corresponding channel) [5]. Then Excitation operation is performed to input this column of feature channel vectors into two fully connected layers and sigmoid, model the dependency between feature channels, and the output obtained is actually the corresponding weights of each channel, which are weighted to the original features by Scale multiplication channel (the first way), which completes the weight assignment of feature channels.

3. Based on the Pruning Lightweighting Method

3.1. Network Pruning: unstructured pruning and structured pruning

Unstructured pruning and structured pruning are two different approaches to neural network pruning. Unstructured pruning refers to pruning individual weights; structured pruning refers to pruning filters/channels/layers. One problem with unstructured pruning methods (direct pruning of weights) is that the resulting weight matrix is sparse and cannot be compressed and accelerated without specialized hardware/libraries. Instead, structured pruning methods prune at the channel or layer level. No specialized hardware/libraries are required to run them since the original convolutional structure is still present [6]. Among structured pruning methods, channel pruning is the most commonly used because it prunes at the finest-grained level while still being compatible with traditional deep learning frameworks [7].

3.2. A typical pruning algorithm is a three-stage process



Fig. 4 Classic pruning process [7]

As shown Figure 4 above.

1, Training: in order to achieve the best network performance, a large model with over-parameterization needs to be trained and used as a reference for optimization.

2, Pruning: in order to obtain a streamlined network structure, the trained large model can be pruned according to specific criteria, i.e., the number of channels or layers in the network structure can be readjusted, etc [8].

3, Fine-tuning: in order to regain the lost performance, it is common practice to fine-tune the pruned model. This means that we can use the retained (i.e., considered important) parameters from the pruned large network to initialize the pruned network. Subsequently, several rounds of fine-tuning are performed on the training set to further optimize network performance [9].

3.3. Currently applied pruning methods

convolution kernel pruning:

1. For each convolution kernel Fi, j, calculate the sum of the $Sj = \sum_{l=1}^{ni} \sum |Kl|$ absolute values of its weights [10].

2, Sorting by Sj, we can cut out the m convolutional kernels with the smallest sum of absolute values of weights and their corresponding feature maps. at the same time, the kernels associated with the cut out feature maps need to be removed in the next convolutional layer.

3, In order to create new weight matrices for layer i and layer i+1, we will copy the remaining weight parameters to the new model [11].

4. Summary

Deep Learning Lightweighting Network Structure Lightweighting and Pruning Lightweighting in Target Detection are methods for optimizing deep learning models for target detection tasks. Through these methods, the size of the model and the number of parameters can be reduced, the computational efficiency and inference speed can be improved, and it can be adapted to the environment with limited computational resources.

Network structure lightweighting methods are mainly used to reduce storage requirements and computation by optimizing and streamlining the structure of the target detection model. Common methods include feature graph dimension reduction, optimized feature extraction module and separation optimization. By reducing the size and number of channels of the feature map, simplifying the structure of the feature extraction module and separating different parts of the network, the complexity of the model can be reduced, and the computational efficiency can be improved.

The pruning lightweight approach, on the other hand, reduces the size and complexity of the model by removing redundant connections and parameters from the model. Parameter pruning reduces the number of parameters by setting to zero and removing parameters with smaller weight values that have less impact on model performance. Channel pruning reduces the width of the network by setting to zero and deleting channels with smaller weight values that have less impact on model performance. Structural pruning reduces the depth and complexity of the network by removing redundant layers, blocks, or modules.

These lightweighting methods can be used in combination with each other to further improve the lightweighting effect of the model. By network structure lightweighting and pruning lightweighting the computational resource consumption of the model can be reduced, and the operation efficiency

of the model can be improved while maintaining a high detection performance. In the process of lightweighting, it is necessary to balance the degree of lightweighting and the detection accuracy of the model to ensure that the lightweighted model still has a better target detection capability. In conclusion, network structure lightweighting and pruning lightweighting methods provide effective lightweighting solutions for deep learning target detection, which can be applied to environments with limited computational resources.

References

[1] Yang J, Li S, Wang Z, et al. Detecting manufacturing defects using deep learning: A comprehensive survey and current challenges [J]. Materials, 2020, 13(24): 5755.

[2] Yun Jie, Jiang De, Liu Y, et al. Real-time target detection method based on lightweight convolutional neural network [J]. Frontiers of Bioengineering and Biotechnology, 2022, 10: 861286.

[3] Zhang Xiao, Zhou Xiao, Lin Ming, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices [C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

[4] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861. 2017

[5] Hu J, Shen L, Sun G. Squeeze excitation network [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018:7132-7141.

[6] Molchanov P, Mallya A, Tyree S, et al. Importance estimation of neural network pruning [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019:11264-11272.

[7] Huang Z, Wang N. Data-driven sparse structure selection for deep neural networks [C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 304-320.

[8] Han S, Mao H, Dally W J. Deep compression: compressing deep neural networks using pruning, training quantization, and Huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.

[9] Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440. 2016

[10] Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710. 2016

[11] He Y, Zhang Xu, Sun Jie. Accelerating channel pruning of extremely deep neural networks [C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 1389-1397.