Mobile-Yolo, A Lightweight Neural Network Based On YOLOv4-tiny

Minhao Gu

Abstract— **In recent years, with the continuous development of deep learning, the depth of network models becomes deeper and deeper, which brings the number of parameters and computation of network models becomes more and more huge. Unlike cloud devices, edge devices or embedded devices often have power and computational limitations, which pose a significant challenge to deploy neural network models on edge devices. How to deploy neural network models to edge devices with little loss of accuracy? The problem of how to smoothly deploy the network models to edge devices and perform fast inference with little loss of accuracy becomes a non-negligible problem. In this paper, based on the YOLOv4-tiny network, we replace the backbone feature extraction network of YOLOv4-tiny with mobilenet to build a new lightweight network mobile-yolo, and the experimental results show that our mobile-yolo has good results.**

Index Terms—**Deep learning, edge devices , YOLOv4-tiny, mobile-yolo.**

I. Introduction

Currently, artificial intelligence (AI) algorithms have been successful in many fields such as computer vision, natural language processing, speech recognition, and semantic segmentation[1], while computer vision is mainly applied to target detection[2,3], and image classification. With the continuous development of technology in the field of deep learning, the layers of deep neural network models are becoming deeper and deeper. For example, the LeNet[4] network started with only 2 convolutional layers, to VGG16[5] with 16 convolutional layers, ResNet50[6] with 49 convolutional layers, and later YOLOv4[7] with 107 convolutional layers. Although increasing the depth of the neural network can improve the detection effect of the network model to a certain extent, it also brings a problem. The number of neural network parameters and the computational effort become larger as the number of layers increases.

Currently, the development of AI technology faces a major problem in that the algorithms it uses are too complex and rely heavily on the high computing power of the cloud and data centers. Compared to cloud devices, edge devices like ARMs, FPGAs, and ASICs have extremely tight memory and compute resource constraints, making it less easy to deploy AI applications on edge devices[8]. In recent years, AI applications are gradually moving to the edge, such as smart robots[9], smart homes[10], self-driving cars[11], and wearables[12], which can have a huge impact on our lives. The implementation of these applications usually relies on algorithms. As the core of the algorithm, how to effectively reduce the amount of computation, number of parameters, and number of visits to the network model without affecting the performance of the network, and enhance the neural network on the edge-end devices It is still a great challenge to improve the operational efficiency of neural networks on edge devices.

Reducing the parameters and computation of network models while ensuring the accuracy of the models is gradually becoming an important direction in the field of computer vision. The design of lightweight networks can be divided into two categories: network structure design and model compression, which can be subdivided into four subcategories: knowledge distillation, pruning[13], quantization[14] and low-rank decomposition. When performing network structure We generally adopt a lightweight mindset when designing the network structure, for example, using lightweight convolution methods such as depth-separable convolution, grouped convolution We generally adopt a lightweight mindset when designing network structures, such as using deep separable convolution, grouped convolution, and other lightweight convolution methods. In addition, we can use global pooling instead of fully-connected layers, 1 × 1 convolution to achieve a more efficient and more accurate decomposition of features. 1 × 1 convolution can reduce the computational effort of the network model. These methods can effectively reduce the computational effort of the network model.

II. Related Work

Since the birth of AlexNet[15], convolutional neural networks have been widely used in image classification, image segmentation, target detection, etc. The number of layers of neural networks has been increasing due to the demand for network performance, which has improved the performance of the network but also brought about efficiency problems.

The earliest lightweight network model can be traced back to SqueezeNet[16] proposed by Berkeley and Stanford researchers, which proposed a fire module consisting of two parts, the first part is the squeeze layer and the second part is the expand layer. squeeze layer is 1 × 1 convolution, the number of convolution kernels is smaller than the number of feature maps in the previous layer, while expand layer uses 1 × 1 and 3 × 3 convolution respectively, and then concat operation. SqueezeNet also adopts a similar idea to VGGNet in the design of network structure., using a stacking of fire

Manuscript received Dec 17, 2022
Minhao Gu, School of Computer Science and Technology, Tiangong University, Tianjin, China

modules. One of its core points is to use depth-wise separable convolution to replace the traditional convolutional model in order to reduce the network weight parameters. The depth-wise separable convolution can be divided into depth-wise convolution and??point-wise convolution.

When the number of weights is the same, the depth-separable convolution can reduce the computational effort of the network exponentially compared with the traditional convolution operation, which can effectively improve the operation efficiency of the network. However, if only depth-wise convolution is used, there is a problem of poor information flow, i.e., the output feature map contains only part of the input feature map instead of all the information. Therefore, MobileNet[17] uses point-wise convolution to propose a good solution to this problem of poor information flow, and in ShuffleNet[18], the point-wise convolution is replaced by channel shuffle to improve the network model.

ShuffleNet was proposed by the Face++ team in 2017. The network is characterized by using group convolution and channel shuffle operations to reduce the number of parameters of the network model. The group convolution has been implemented since AlexNet, but the group convolution is used due to the hardware limitation, while the channel shuffle operation is used to improve the problem of poor information flow between groups. In terms of network topology, ShuffleNet uses the classical residual structure. YOLOv4-tiny[19], proposed by AlexeyAB, is a lightweight version of YOLOv4 with only one-tenth of the number of parameters. In terms of network structure, it borrows three residual modules from ResNet, uses LeakyRelu as the activation function, and uses a feature pyramid structure when merging the effective feature layers, which gives YOLOv4-tiny a significant performance advantage over other lightweight network models.

III. Framework

In this chapter, a new network model mobile-yolo is constructed based on the design idea of lightweight network, which can improve the inference speed with little loss of accuracy. The following table shows the parameters of each layer of mobile-yolo

Table1.Parameters of mobile-yolo layers

layers	In_c	Out_c	S	P	Group
CBH_3×3	3	32	2	1	-
CBH_3×3	32	32	1	1	32
CBH_1×1	32	64	1	1	-
CBH_3×3	64	64	2	1	64
CBH_1×1	64	128	1	0	-
CBH_3×3	128	128	1	1	128
CBH_1×1	128	128	1	0	-
CBH_3×3	128	128	2	1	128
CBH_1×1	128	256	1	0	-
CBH_3×3	256	256	1	1	256
CBH_1×1	256	256	1	0	-
CBH_3×3	256	256	2	1	256
CBH_1×1	256	512	1	0	-
CBH_3×3	512	512	1	1	512
CBH_1×1	512	512	1	0	-
CBH_3×3	512	512	1	1	512
CBH_1×1	512	512	1	0	-
CBH_3×3	512	512	1	1	512
CBH_1×1	512	512	1	0	-
CBH_3×3	512	512	1	1	512
CBH_1×1	512	512	1	0	-
CBH_3×3	512	512	1	1	512
CBH_1×1	512	512	1	0	-
CBH_3×3	512	512	2	1	512
CBH_1×1	512	1024	1	0	-
CBH_3×3	1024	1024	1	1	1024
CBH_1×1	1024	1024	1	0	-
CBH_1×1	1024	256	1	0	-
CBH_3×3	256	512	1	1	-
Conv2D_3×3	512	75	1	0	-
CBH_3×3	256	128	1	0	-
Upsample	128	128	-	-	-
CBH_3×3	640	256	1	1	-
Conv2D_3×3	256	75	1	0	-

Where CBH indicates our base convolution block (Conv2D+BatchNormal+Hard-Swish), CBH_3 × 3 and CBH_1×1 indicate the size of the convolution kernel as 3×3 and 1×1 convolution blocks, respectively. In_c and Out_c correspond to the number of input channels and output channels of each layer, respectively; S indicates the step length of the convolution kernel sliding on the feature map during convolution. When S equals to 1, it will not change the length and width of the feature map, and when S equals to 2, the length and width of the output feature map will be compressed to one-half of the input; Padding indicates whether to padding the feature map, 0 means no padding, 1 means a circle of zeros around the input feature map before doing the operation; Group indicates whether to group convolution, mostly used in depth separable Convolution, '-' means no grouping, and the number of group is consistent with the number of input channels.

IV. Experiment

A. *Experiment preparation*

The training device we use is Tesla V100, the CUDA version we use is 11.4, the torch version is 1.7.0, the torchvision version is 0.8.0, before we start training, we will fix the size of the input image at 416×416, the number of rounds for each training is 100, because no pre-training weights are added, so 100 The initial learning rate is set to 1e-3, the weight decay is 5e-4, the batch_size is 32, and the optimizer we use is the Adam optimization algorithm. Adam is a first-order optimization algorithm that can replace the traditional stochastic gradient descent process and can iteratively update the weights in the neural network based on the training data, with the characteristics of fast convergence.

B. *Dataset*

The experimental dataset used in this paper is the VOC dataset, which includes the data of VOC2007 and VOC2012 versions. As one of the most commonly used standard datasets in the field of target detection and image segmentation, it contains 4 major categories and 20 minor categories of objects, with a total of 18,000 images.

Before the experiment starts, we divide the 18,000 images into a training validation set and a test set in a random ratio of 9:1, i.e., the training validation set has 16,200 images and the test set has 1,800 images. After dividing the 16,200 images in the training validation set, we will again divide them into training and validation sets according to the ratio of 9:1. That is, there are 14580 images for training, 1620 images for validation, and 1800 images for testing.

Table2.VOC data set segmentation

	Total	Train_set	Valid_set	Test_set
ratio	1	0.81	0.09	0.1
number	18000	14580	1620	1800

C. *Results and Analysis*

In order to verify the performance of our designed lightweight network, we compared mobile-yolo with the original YOLOv4-tiny network in terms of the number of parameters, computation, map, and inference speed of the network, respectively, and the results are shown in the following table.

Table3.comparison between mobile-yolo and YOLOv4-tiny

Model	Acti	Map(%)	Infer_time
YOLOv4-tiny	LeakyRelu	56.58	0.1068s
YOLOv4-tiny	Hard-Swish	57.20	0.1047s
mobile-yolo(our)	Relu	61.95	0.0827s
mobile-yolo(our)	Hard-Swish	62.98	0.0788s

Where Acti represents the different activation functions, it can be found that no matter which activation function is used, the accuracy and inference speed of our mobile-yolo on the VOC dataset is also slightly better than the original YOLOv4-tiny network.

D. *Results and Analysis*

The following figure shows the prediction effect of mobile-yolo network on VOC dataset.

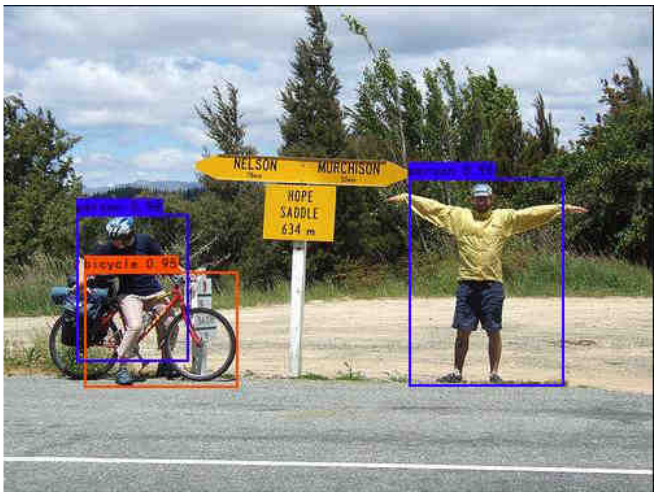

Figure 1. Prediction effect on VOC dataset

V. Conclusion

In this paper, we design a lightweight network mobile-yolo based on YOLOv-tiny from the problem that deep neural networks are not easy to deploy in memory and arithmetic-limited edge-end devices, aiming to reduce the computation and number of parameters of deep neural networks while accelerating the inference speed of the model on hardware. Experimental results show that our lightweight network model mobile-yolo has a 1.3 times improvement in inference speed with higher accuracy than YOLOv4-tiny, so that we can prove that our lightweight network mobile-yolo has good practical application effect.

References

[1] Kurnikov P A, Sholomov D L. DNNs for multi-map semantic segmentation[C]//Thirteenth International Conference on Machine Vision. SPIE, 2021, 11605: 342-349.

[2] Teplyakov L, Kaymakov K, Shvets E, et al. Line detection via a lightweight CNN with a Hough layer[C]//Thirteenth International Conference on Machine Vision. SPIE, 2021, 11605: 376-385.

[3] Ye S P, Chen C X, Nedzved A, et al. Building detection by local region features in SAR images[J]., 2020, 44(6): 944-950.

[4] El-Sawy A, El-Bakry H, Loey M. CNN for handwritten arabic digits recognition based on LeNet-5[C]//International conference on advanced intelligent systems and informatics. Springer, Cham, 2016: 566-575.

[5] Guan Q, Wang Y, Ping B, et al. Deep convolutional neural network VGG-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study[J]. Journal of Cancer, 2019, 10(20): 4876.

[6] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[7] Gai R, Chen N, Yuan H. A detection algorithm for cherry fruits based on the improved YOLO-v4 model[J]. Neural Computing and Applications, 2021: 1-12. yolov4.

[8] Liu R, Li Y, Tao L, et al. Are we ready for a new paradigm shift? a survey on visual deep mlp[J]. Patterns, 2022, 3(7): 100520.

[9] Johnson S D, Blythe J M, Manning M, et al. The impact of IoT security labelling on consumer product choice and willingness to pay[J]. PloS one, 2020, 15(1): e0227800.

[10] Evans G. Solving home automation problems using artificial intelligence techniques[J]. IEEE transactions on consumer electronics, 1991, 37(3): 395-400.

[11] Gill T. Blame it on the self-driving car: how autonomous vehicles can alter consumer morality[J]. Journal of Consumer Research, 2020, 47(2): 272-291.

[12] Katzschmann R K, Araki B, Rus D. Safe local navigation for visually impaired users with a time-of-flight and haptic feedback device[J]. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2018, 26(3): 583-593.

[13] Reed R. Pruning algorithms-a survey[J]. IEEE transactions on Neural Networks, 1993, 4(5): 740-747.
[14] Gray R M, Neuhoff D L. Quantization[J]. IEEE transactions on information theory, 1998, 44(6): 2325-2383.
[????] Aliyu H A, Razak M A A, Sudirman R, et al. A deep learning AlexNet model for classification of red blood cells in sickle cell anemia[J]. Int J Artif Intell, 2020, 9(2): 221-228.
[16] Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.
[17] Sinha D, El-Sharkawy M. Thin mobilenet: An enhanced mobilenet architecture[C]//2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON). IEEE, 2019: 0280-0285.
[????] Zhang X, Zhou X, Lin M, 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.
[19] Jiang Z, Zhao L, Li S, et al. Real-time object detection method based on improved YOLOv4-tiny[J]. arXiv preprint arXiv:2011.04244, 2020.