2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems

Structure Characteristic-aware Pruning Strategy For Convolutional Neural Networks

1st Peixuan Zuo
BeiHang University
Beijing, China
zuopeixuanzryhd@163.com

2nd Rui Wang*
BeiHang University
Beijing, China
wangrui@buaa.edu.cn

3rd Xianya Fu
BeiHang University
Beijing, China
fuxianya@buaa.edu.cn

4th Hailong Yang
BeiHang University
Beijing, China
hailong.yang@buaa.edu.cn

5th Yi Liu
BeiHang University
Beijing, China
yi.liu@buaa.edu.cn

6th Lianyi Zhang
Secience and Technology on Special System Simulation Laboratory
Beijing, China
lyzhang117@163.com

7th Han Zhang
Beijing Simulation Center
Beijing, China
xia_mei2000@163.com

8th Depei Qian
BeiHang University
Beijing, China
depeiq@buaa.edu.cn

***Abstract*—Convolutional Neural Networks have received considerable attention over the past few years, and they are widely used in various fields. However, the computational complexity and excessive storage space caused by over-parameterized of these networks are also being serious. This will limit further development of deep learning in resource-constrained devices such as the mobile terminal. It has been proposed that pruning is a useful method to optimize networks, whereas there is no contribution focus on utilizing the traits of the network to guide pruning. The primary purpose of this paper is to research the different effects of layers with varying depth and the different number of parameters during the pruning process. By taking advantage of the characteristics of network structure itself, we propose a new pruning strategy with three variants to effectively prune CNNs which can automatically determine the most appropriate sparsity for each conv-layer. Using iterative pruning and retraining, we evaluated our strategy on InceptionV3 and ResNet50. We analyzed our three variants and made a comparison between each other.**

***Index Terms*—Convolutional Neural Network, Pruning, Structure characteristic, Image classification, Object detection.**

I. Introduction

Convolutional Neural Networks(CNNs) [1] have attracted considerable attention in the last few years and achieved state-of-the-art performance in various application domains, especially computer vision, natural language processing, and speech recognition, etc. Because of the rapid development of powerful GPUs [2], researchers have available experimental conditions including plenty of storage resources and strong computer power of exploring the learning capabilities of deep neural networks. Nowadays, the accuracy of the CNNs' image classification has already surpassed that of humans.

In spite of its great success, the structure of neural network has become deeper and more complex to obtain better performance, and the improvement of precision is always accompanied by the increased of parameters, which leads to some problems such as large storage, excessive computation and high energy consumption required by CNNs. For instance, AlexNet [3], which proposed in 2012, has only eight layers, while Renset152 [4], proposed three years later, contains 152 layers which take up more than 200M storage space and over 22 GFLOPs to classify a single image.

Because of the above disadvantages, CNNs' applications are difficult to deploy on resource-constrained devices, e.g., mobile phones. Thus, many approaches have been proposed to compress CNNs models and reduce resource consumption from various aspects to meet the requirements of practical applications. One of the most widely used approaches is pruning. As early as the 1990s, LeCun et al. [5] have found that removing part of redundant parameters from a pre-trained network model only bring a negligible loss in accuracy. Based on the assumption that several parameters have little effect on the final result, pruning methods are proposed to reduce the redundant parameters from a CNN model which has already been trained. As shown in Fig. 1, this method is known figuratively as pruning. There are two main advantages to pruning CNNs: On the one hand, the trimmed parameters don't participate in calculation during the prediction process so that the total computational complexity can be reduced. On the other hand, the reduction of parameters leads to a decrease in model size, thus saving storage space.

Although pruning has been proposed for a long time, with the increase of sparsity, the loss of precision is more difficult to control. In this paper, the proportion of the parameters removed to the total parameters is called sparsity, and the ratio of the parameters of each layer removed is called layer sparsity. Studies always rely on long-term retraining to restore accuracy and focus on proposing various pruning criteria, but they ignored to explore the characteristics of the network

This work was supported in part by National Key R&D Program of China under grant No.2017YFB0203201.This work is also supported by NSFC under grant No.61732002. Rui Wang is the corresponding author.

978-1-7281-2058-4/19/$31.00 ©2019 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2019.00211

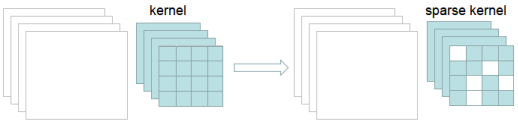


Fig. 1. Left: The original convolutional kernel without pruning. Right: The convolutional kernel after the pruning operation, the white part is the removed part and such pruning of an individual weight is called fine-grained pruning. The removed weight does not participate in the calculation in inference, nor does it need to be stored.

structure itself in the pruning process. Notwithstanding some works [9] detected differences between layers, they analyzed each layer separately and manually specified the appropriate threshold for each layer, which needs extensive experiments and took a lot of time. Different from the works list above, our work focuses on automatically determining the most appropriate sparsity for each layer. In our work, we consider the location and number of parameters of each convolutional layer in the CNN, and we propose a more targeted sparse method for network models with different structures.

The main work of this paper is to pruning convolutional layers. In our work, we propose a strategy with three variants which are named Prune_fixed, Prune_param and Prune_depth according to the characteristics of structures of CNN for pruning CNN, and we compare the effects of these three variants on accuracy in a fixed number of training steps. Our key contributions can be summarized as follows:

- We Implemented a particular pruning strategy with three variants based on the distinctions in the importance of different layers. 1). Prune_fixed: It is considered that each layer takes the same responsibility for the accuracy, and all layers should be pruned without discrimination. 2). Prune_param: For CNNs with large differences in the number of parameters in each layer, it is considered that the influence of pruning on the layer with smaller parameters is greater, and the networks should be pruned according to the number of parameters in the layer. 3). Prune_depth: For CNNs with large quantities of layers, it is considered that the layers close to the original image have a more significant influence on the overall accuracy, and should be pruned discriminatory according to the depth of the layer.
- We analyze the characteristics of weights at each layer in the network, and evaluated whether the weight of each layer has the same effect on overall accuracy from two aspects involving the depth of the layer and the number of layer parameters.
- We conducted experiments on InceptionV3 [8] and ResNet50 and compared the advantages and disadvantages of the three variants in terms of sparsity and precision. The effect of Prune_fixed is consistent with the result in [7], but the final accuracy of two hierarchical pruning variants is more excellent. And we chose the most versatile variant, Prune_fixed, which is the indiscriminate pruning, to perform pruning with the sparsity of 0.5 on several typical CNNs.

The rest of this paper is organized as follows. In Section 2, we present some background on network compression and acceleration. In Section 3, we describe our strategy with three variants to implement the pruning on CNNs. In Section 4, we give related experimental data, explain experimental phenomena, and give some suggestion during pruning. In section 5, we discuss some future directions in this filed and concludes the paper.

II. RELATED WORK

Some existing works have been applied with the purpose to optimize neural networks to make them more suitable for source-limited devices [11]. Pruning optimization was proposed long before the rapid development of deep learning and has been thoroughly studied in recent years [5], [27], [9]. Pruning means discovering which parameters are relatively nonsignificant in the network and discard them, and only critical parameters are left to participate in the inference. Several pruning strategies were carried out around the judgment on the significance of connectivity. Han [6] took the absolute value of the weight as the pruning standard, and this method can achieve better compression under the condition of controlling the loss of precision. Started from the perspective of minimizing energy consumption of CNNs with marginal accuracy degradation, Yang [10] proposed a pruning algorithm based on energy cost estimation involving data movement and computation on the actual hardware. This method guides the pruning process based on the ability that is identifying the part of the weights that consumes the most energy.

More recently, [7] compare the advantage and disadvantage between large-sparse and small-dense models and proved that large-sparse models are outperformed small-dense in the case of trade-off accuracy and model size. It realized the pruning experiment on InceptionV3 and when the sparsity reaches 0.75, the accuracy of top1 drops by 2%. However, this method didn't carry out a unified analysis of different layers. So the accuracy sacrifice is more significant. Our work compensates for this defect.

The pruning method of individual weights has studied intensely. However, the irregular network structure brought by those fine-grained approaches limits their universality and flexibility [18]. To avoid these weaknesses, studies concerns on coarse-grained pruning [12], [13], [14], [15] to pursue more regular network results. Since coarse-grained sparse networks require fewer indexes which point to unstructured parameters, Vector-level pruning requires less storage space than fine-grained pruning. There is also kernel-level pruning which directly prunes two-dimensional matrix in the 3D kernel. The most extreme method like channel-level [16] reduction and filter-level [17] reduction, change the overall structure of the neural network by reducing the input and output feature maps and the weights of the corresponding convolutional kernel.

At the same time, those drastically pruning sacrifices more accuracy than fine-grained sparsity.

Besides, there are other potentially useful approaches to optimize CNNs including quantization [19], low-rank decomposition [20] and Network distillation [21], etc.

III. OUR STRATEGY

We selected the fine-grained pruning method based on the absolute value of weight to explore the influence of the characteristics of the convolutional neural network itself on the pruning effect. From the mathematical point of view, the absolute value of weight is the most straightforward influencing factor of the prediction results. So we choose the absolute value of weights to be the pruning criterion. We prefer fine-grained pruning method which can better control the sparsity of different layers.

The primary scheme of pruning the connection of a network is to establish which weights are significant and to remove the insignificant ones. The deleted connections will not participate in inference and storage, which can save lots of computing resources and compress the network model.

As shown in Fig. 2, to actualize pruning operation, we replace the original weight layer with a new masked weight layer, which is the result of multiplying the weight layer and the mask layer. The mask layer is added as the same shape and size as the original weight layer, and its contents consist of two numbers: one and zero. A value of 0 in the mask means regardless of the weight of the same position in the weight matrix, the value multiplied by the mask layer is 0 and will not participate in the calculation of forwarding and backward propagation. At the beginning of the pruning, none of the connections should be removed, so masks are matrices of all 1. During the pruning process, the value of the same position on the mask corresponding to a parameter is set to 0 when this parameter is judged to need to be removed. Otherwise, the value is still 1. After pruning, other parameters that have not been removed should be retrained to compensate for the decreased precision of pruning. Based on Han, pruning and retraining iterative can achieve better compression ratio compared with retraining after the directly pruning reaches final sparsity. Therefore, we adopt the method of gradually increasing the total sparsity during the pruning process and constantly fine-tuning overall layers under current sparsity, so as to restore the accuracy in time.

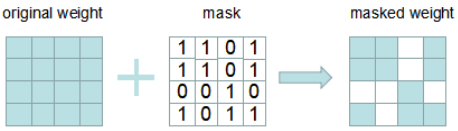


Fig. 2. After multiplying the original weight with the mask layer, the removed weights in the masked weight matrix became 0, and finally, the masked weights were calculated instead of the initial weights.

The essential step in pruning is to determine which connections should be removed. The simplest method is based on the absolute value of the parameters. We can think that the smaller the absolute value of the weight, the lower the influence on the generated feature map in the reasoning procedure. So we can find out a threshold, and all weights below this threshold are considered unimportant so that can be removed. However, from the perspective of the entire network, more issues need to be considered. There are some crucial questions, such as whether the layers of the network should be considered separately or as a whole? Are the parameters of each layer in the network play the same role in the whole network? Where is the distinction if the importance of each layer is different? What is the effect of layers with different characteristics on pruning operations and final precision? We start our work based on these issues. In the following context, the threshold of a layer means that the weight of this layer whose absolute value is less than this threshold needs to be removed.

A. PRUNE_FIXED

If each layer in the network is viewed isolated, all layers in the network can be considered as equally significant to the ultimate precision. Then each layer is pruned individually, and the pruning ratio of every layer is according to the overall sparsity, that is the layer sparsity is equal to the overall sparsity, and each layer maintains its layer threshold to achieve the purpose of pruned separately. We named this non-differential pruning variant as Prune_fixed, its general flow is as follows, and the other two variants have the same necessary steps as non-differential pruning:

Algorithm 1: Pruning with PRUNE_FIXED

Input: L contains each layer, target sparsity S, retraining step τ.
Output: threshold $\{\delta^l\}$ of each layer.
while *sparsity* s != S **do**
  $s = s + \Delta s$;
  $update\ \{\delta^l\},\ l \in L$;
  $prune\ connections\ using\ \{\delta^l\},\ l \in L$;
  for *each* $i \in [1, \tau]$ **do**
    $retrainint\ to\ restore\ accuracy$;
  end
end

B. PRUNE_PARAM

Different from the previous Prune_fixed, the number of parameters become one of the criterions for judging the importance of different layers in a network. 1)Layers with more parameters account for a more substantial proportion of the overall parameters and produce a more significant influence on the compression ratio after pruning, while layers with fewer parameters pruning with the same layer sparsity, the impact for the overall compression ratio is still small. 2) A layer with fewer parameters prunes a large percentage of weights, which has a more significant effect on accuracy.

Algorithm 2: Pruning with PRUNE_PARAM

Input: L contains each layer, target sparsity S, Δs, threshold η, ζ of the loss value, retraining step τ.
Output: threshold $\{\delta^l\}$ of each layer.
while *sparsity* s *!=* S **do**
  $PreLossRecord = lossRecord$;
  $lossRecord = 0$;
  $s = s + \Delta s$;
  $update\ \{\delta^l\},\ l \in L$;
  $prune\ connections\ using\ \{\delta^l\},\ l \in L$;
  for *each* $i \in [1, \tau]$ **do**
    $retraining\ and\ compute\ loss$;
    $lossRecord+ = loss$;
  end
  $lossRecord = lossRecord/\tau$;
  $\theta = PreLossRecord - lossRecord$;
  if *lossRecord* $> \eta$ *or* $\theta > \zeta$ **then**
    $L = L - l_{min}$;
  end
end

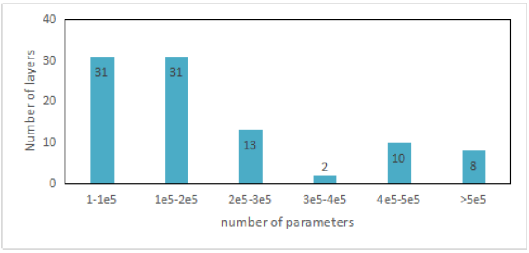


Fig. 3. It shows frequency statistics of the parameters of each layer in InceptionV3. There are 31 layers with the number of parameters less than 100,000, and the total number of these layers accounts for less than 10% of all parameters.

Different layers are treated differently depending on the parameters of each layer. As can be seen from the Fig. 3, the number of parameters of each layer of InceptionV3 varies greatly. We designed a pruning method to control the sparsity of different layers dynamically. We assume that the layer with the least parameters in the currently trimmed layers should be stopped pruning if the classification accuracy has been significantly reduced during the pruning process. In the implementation, this is equivalent to adding a hyper-parameter to control sparsity dynamically.

We define an easily accessible value loss to substitute the accuracy, and we establish two thresholds η and ζ of the loss value to monitor the pruning procedure. We calculate the average of the loss $lossRecord$ of each τ steps and then compute the difference θ between $lossRecord$ the average loss value $PreLossRecord$ of the previous τ steps. It is found that θ in the training of the pre-trained model is small, but θ in the pruning process fluctuates greatly. We assume that when θ varies greater than ζ, it indicates that the newly added sparsity affects the overall accuracy, and the pruning of the current minimum layer should be stopped to compensate for the loss of precision. In order to control the slow change of the loss value, we not only constrain the difference of loss between two stages, we also require that the loss value itself cannot exceed a fixed threshold η which varies according to the specific network.

C. PRUNE_DEPTH

Algorithm 3: Pruning with PRUNE_DEPTH

Input: W^{abs}, constrain ε.
Output: threshold δ.
while *sparsity* s *!=* S **do**
  $s = s + \Delta s$;
  $min = 0; max = max(W^{abs})$;
  while *min* $<$ *max* **do**
    $\delta = \frac{min+x}{2}$;
    $compute\ sparsity\ threshold\_s\ using\ \delta$;
    if $|threshold\_s - s| < \varepsilon$ **then**
      $return\ \delta$;
    end
    if $threshold\_s > s$ **then**
      $max = \delta$;
    else
      $min = \delta$;
    end
  end
end

Prune_depth establish the sparsity of each layer from the perspective of the entire network and achieves hierarchical pruning of the model. Analyze from the depth of the network we assume that because of the shallower layers are closer to the original image, the features they extract are more important than the deeper ones. If the error is significant at the beginning of the extraction features, the subsequent feature extraction would amplify the effect of the error and thus affect the final accuracy. Therefore, it is necessary to realize hierarchical pruning under the condition that the overall sparse ratio remains unchanged. The shallower layers should have as less pruning as possible compared to the deeper layers. Our work suggests that, for the ease of application, the sparsity of each layer should be determined according to the overall weights characteristics, rather than artificially specified.

Fig. 4 shows the distribution of the pre-trained InceptionV3 parameters, The X-axis represents the depth of the layer, and the Y-axis represents the average of the absolute values of the parameters of each layer. What can be seen in Fig. 4 is that as the layer is deepened, the magnitude of the parameter's average value gradually decreases. Based on this phenomenon, We try to use the characteristics of the original weight to consider the weight of all layers uniformly. Based on a total sparsity we compute a threshold, and each layer uses the same

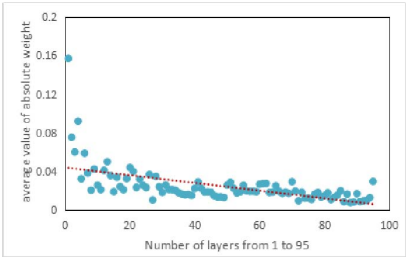


Fig. 4. The average value of the absolute weight of each layer of InceptionV3 shows that the absolute value of the weight of the shallow layers is large, whereas the absolute value of the weight of the deep layers is diminutive.

threshold to guide the pruning. In this way, the sparsity of each layer varies with depth.

The constant sorting of millions of parameters is a massive burden on memory and computing power which slowing down the pruning process. We designed a faster algorithm to achieve the same pruning ambition about less memory occupy and computing power.

IV. EXPERIMENT

In this section, we present experimental data to demonstrate the rationality of our method. We first verify the validity of our proposed method using layers of different traits. For better understanding, we use our strategy with three variants tested on InceptionV3 and ResNet50 to compare the differences. And then we choose the pruning results of InceptionV3 to carry out detailed experimental analysis of the three variants. The reason that we select InceptionV3 and ResNet50 is that they have a sufficient number of convolutional layers, and the number of parameters between the layers varies greatly. They are widely used in image classification and object detection, etc.

We employ the most common non-discriminating pruning method in image classification and object detection field and give the experimental results when the sparsity is 0.5. The accuracy is measured on the ImageNet ILSVRC 2012 dataset [22], and our hardware platform is GeForce GTX 1080 GPU.

A. VALIDITY EXPERIMENT

To verify the rationality of Prune_param and Prune_depth, we perform some validity experiments on InceptionV3 and VGG16. Performance is measured by the decrease of top1 accuracy after a particular convolutional layer is pruned with fine-tuning, shown in TABLE. I, and TABLE. II. As expected, error increase as the layer is shallow and the difference in the number of parameters in the layer also brings the variation of accuracy.

The main idea of Prune_param is that the layers with fewer parameters should have as few pruning as possible so that the image information passing through those layers will not be seriously lost. As shown in TABLE. I, we selected two adjacent layers with the different number of parameters from InceptionV3. The two layers are Conv2d_4a_3x3 and Mixed_5b/Branch_0/Conv2d_0a_1x1, which are located in

TABLE I
INCEPTIONV3:ACCURACY LOSS WHEN LAYER SPARSITY IS 0.95

Variant	Parameters	Layer	Layer's Name	Top1 Loss
Prune_param	138240	-	Conv2d_4a_3x3	4.4%
	12288	-	Mixed_5b/Branch_0 /Conv2d_0a_1x1	9%
Prune_depth	-	31^{st} layer	Mixed_6b/Branch_0 /Conv2d_0a_1x1	4%
	-	61^{st} layer	Mixed_6e/Branch_0 /Conv2d_0a_1x1	1.2%

TABLE II
VGG16:ACCURACY LOSS WHEN LAYER SPARSITY IS 0.95

Variant	Parameters	Layer	Layer's Name	Top1 Loss
Prune_param	2359296	-	Conv2d_4_2	7.2%
	1179648	-	Conv2d_4_1	7.7%
Prune_depth	-	9^{st} layer	Conv2d_4_2	7.2%
	-	13^{st} layer	Conv2d_5_3	7.7%

layers fifth and sixth of InceptionV3 and the number of parameters is 138240 and 12288, respectively. The two layers were separately pruned and the target sparsity was set to 0.95. The layer with fewer parameters is greatly reduced in accuracy after being sparse, and the accuracy loss of the layer with more parameters is only half of that of the sixth layer. Considered that the depth of the selected layer may affect the experimental results, the two selected layers are intimately connected, and the layers with fewer parameters are slightly deeper than the layers with more. The experimental results further verify the influence of the number of parameters in the layers on the final accuracy.

The main idea of Prune_depth is that the shallower the layer, the greater the impact on overall accuracy after pruning. As shown in TABLE. II, we select two layers with the same number of parameters from the InceptionV3, which are far apart from each other. The two layers are Mixed_6b/Branch_0/Conv2d_0a_1x1 and Mixed_6e/Branch_0/Conv2d_0a_1x1, which are located in the 31st and 61st layers, respectively. We separately pruned the two layers to 0.95 and evaluated the final accuracy loss. The precision of the shallow layer decreased by 4%, while the loss of the deeper precision was controlled within 2%.

We did similar work in VGG16, we pruned the ninth and thirteenth layers with the same number of parameters to verify Prune_depth, and we pruned the eighth and ninth layers that differ in the number of parameters by a factor of two to verify Prune_param. The experimental results proved that the depth of the pruning layer and the number of parameters of the pruning layer have different effects on the pruning effect.

B. TREAT LOSS AS METRIC OF ACCURACY

Traditional, researchers use a test dataset to evaluate the result of pruning. However, it is time-consuming to monitor the test accuracy during the pruning process continuously. Due

to the lack of testing, we use the value of loss as the matric of accuracy to judge whether the layer with the fewest parameters currently should stop trimmed or not. Because the relationship between loss value and accuracy is inversely proportional, the accuracy change during the pruning and retraining process can be tracked through the loss value. We can presume that the loss value decreases, indicating the accuracy increase. If the batch size is appropriate, loss value and accuracy value during training meet the following formula:

$$accuracy = e^{-\text{loss}} \tag{1}$$

Higher values of loss imply more inaccuracy in the pruning process. For instance, during the pruning process, if the average loss value of stage higher than that of the previous stage, it shows that pruning brings a decrease in accuracy. As shown in the Fig. 5, as the sparsity increase, there is an apparent change trend in the loss value, especially after the sparsity reaches 0.5.

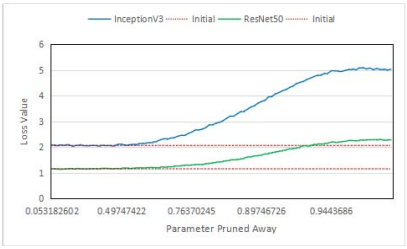


Fig. 5. When the sparsity below 0.5, the accuracy doesn't change, and the loss value is stable. But when the sparsity exceeds 0.5, the influence of pruning on accuracy increases gradually, and meanwhile, the loss value increases.

C. COMPARSION OF THE THREE VARIANTS

TABLE. III and TABLE. IV compared the effects of the three variants when the sparsity was 0.5 and 0.75, respectively. For clarity, we compare our strategy with pruning result showed in [7] and prove the superiority of our approach.

TABLE III
INCEPTIONV3:ACCURACY LOSS OF DIFFERENT PRUNING VARIANTS UNDER DIFFERENT SPARSITY

Variant	Sparsity	Top1 Increase Error
sparse_InceptionV3 [7]	0.5	<1.0%
	0.75	2.0%
Prune_fixed	0.5	<1.0%
	0.75	2.0%
Prune_param	0.5	<1.0%
	0.75	1.5%
Prune_depth	0.5	<1.0%
	0.75	1.2%

We don't retrain for a long time, and we discuss the effects of three variants when we fix a certain number of retraining steps. Results in TABLE. III and TABLE. IV are obtained with pruning steps of 40000 and a mini-batch of 32 on GPU.

We can observe that when the sparsity is 0.5, accuracy is limited influenced. But wait until the sparsity goes up to 0.75, our approach with Prune_fixed obtains the same effect as sparse-InceptionV3. Prune_param and Prune_depth successfully outperform from the result in [7].

TABLE IV
RESNET50:ACCURACY LOSS OF DIFFERENT PRUNING VARIANTS UNDER DIFFERENT SPARSITY

Variant	Sparsity	Top1 Increase Error
Prune_fixed	0.5	<1.0%
	0.75	1.6%
Prune_param	0.5	<1.0%
	0.75	1.4%
Prune_depth	0.5	<1.0%
	0.75	1.3%

We observed the change of loss value with sparsity in the pruning process of three variants. From Fig. 6, we can see where the difference between the three variants starts. It shows that the sparsity of 0.5 is a barrier to pruning, No matter which method has a sparsity of more than 0.5, the accuracy is reduced.

Among these three variants, the method of using network depth to distinguish layer sparsity shows the best effect in InceptionV3 and ResNet50. This shows that when processing pruning, it is more important to consider the expression condition of the network than to consider the parameter of each layer.

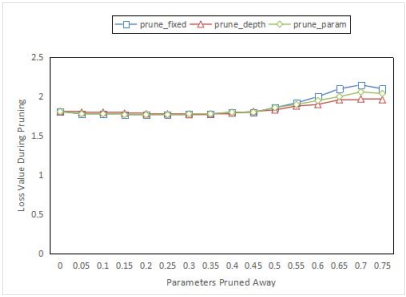


Fig. 6. The figure shows the change of loss value with the increase of sparsity. Before the sparsity reached 0.5, the variation trend of accuracy of the three variants was almost identical. But after the sparsity reaches 0.5, the accuracy of the three variants changes, and the loss value of the Prune_fixed increases dramatically, which means the accuracy loss is relatively severe.

D. PRUNE_FIXED EXPERIMENT

Because of the sparsity over 0.5 will results in the loss of accuracy of the convolutional neural network in the previous experiments, we used the Prune_fixed, which is the most common way to conduct pruning with the sparsity of 0.5 on CNNs in different computer vision fields including image classification and object detection.

As shown in Fig. 7, we evaluate Prune_fixed for the popular InceptionV1 [23], ResNet [24] and VGG16 [25]. The increase of accuracy after pruning in the ResNet series is because the original dense model is not sufficiently trained, so the iterative pruning and retraining on them are equivalent to continuing training the original model.

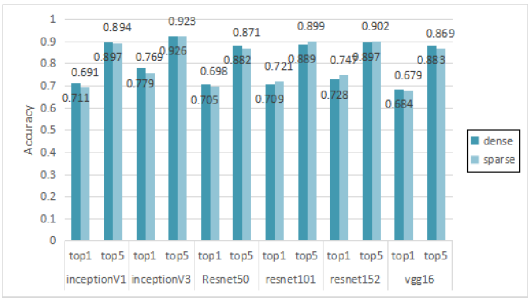


Fig. 7. The accuracy of variant Prune_fixed on different image classification CNNs when the sparsity is 0.5.

Object detection tasks are a very delicate assignment. We evaluate Faster R_CNN [28] and SSD [29] object detection models with pruning InceptionV2 which sparsity is 0.5. We use COCO [30] object detection dataset contains 80 object categories. The performance is evaluated by mean Average Precision(mAP), and the Intersection over Union(IoU) is 0.5. In our experiments, we directly pruned object detection models on COCO rather than using pre-pruned models on the ImageNet. As shown in Fig. 8, object detection CNNs is exceedingly sensitive to pruning than image classification.

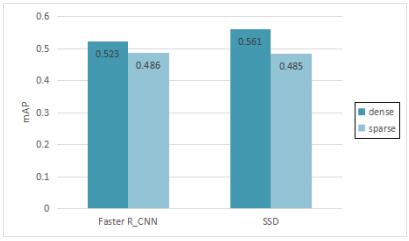


Fig. 8. The accuracy of variant Prune_fixed on different object detection CNNs when the sparsity is 0.5.

E. PRUNE_PARAM EXPERIMENT

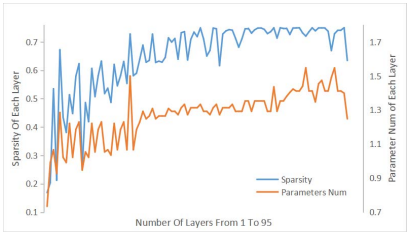


Fig. 9. The sparsity of each layer in InceptionV3 after pruning using Prune_param.

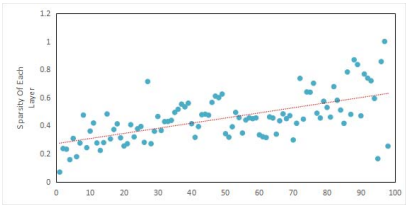
Fig. 9 shows the layer sparsity after pruning when the total sparsity is 0.75, and compare the layer sparsity with the number of parameters of each layer of the original network. The $Parameter\ Num$ is calculated by logarithm with a base of 10,000.

As shown in Fig. 9, the difference in sparsity between layers is noticeable. Most of the layers that stopped pruning were stopped above 0.5, which meant that when the sparsity was above 0.5, the precision began to decline rapidly. This variant is useful for CNNs with significant differences in the parameters of the layers.

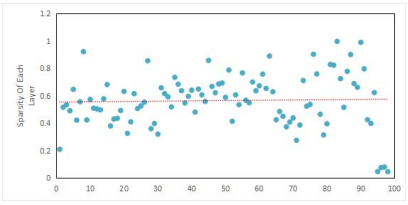
F. PRUNE_DEPTH EXPERIMENT

The Prune_depth uses the original parameters as the object for determining the threshold. We also discussed whether it is needed to normalize the weights. We use a normalization formula to precess weights before computing threshold:

$$x = \frac{|x| - |x|_{min}}{|x|_{max} - |x|_{min}} \tag{2}$$



(a) non-normalization



(b) normalization

Fig. 10. Recorded layer sparsity of each layer in InceptionV3 after pruning with normalization and non-normalization in the case that the total sparsity is 0.6. (a)When we used original weights, the sparsity of each layer in the network changes with the increase of depth. (b)The sparsity of each layer after normalization damage apparent trend with the deepening of the number of layers, indicating that the difference between each layer is no longer significant.

$|x|_{min}$ is the parameter value with the smallest absolute value in the parameters of each layer, and $|x|_{max}$ is the parameter value with the largest absolute value in the parameters of each layer. By such normalization, the parameters are limited to the range of [0, 1] to improve the comparability between the layers. But the disadvantages of the normalization can be discussed under two headings, which are:

- Damage Characteristic: As can be seen from the Fig. 10, the normalization destroys the characteristic already possessed by the original weight that the average value of each layer parameter decreases as the number of layers deepens.
- Individual effect: Because the absolute value of some individual weights of some layers is much larger than the average weight of this layer, the $|x|_{max}$ in these layers will be very large, resulting in that normalized weights

of this layer concentrated around 0. During the pruning process, when the total sparsity is still small, these layers have reached a very high layer sparsity.

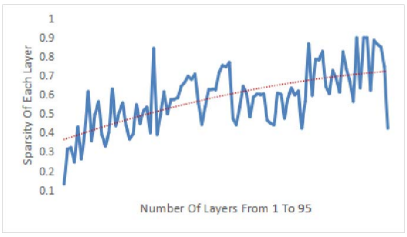


Fig. 11. The sparsity of each layer in InceptionV3 after pruning using Prune_depth. Layer sparsity tends to increase with depth.

As shown in Fig. 11, the sparsity of each layer using a uniform threshold varies greatly, and the sparsity of the layer increases with the increase of the number of layers. The effect of layer sparsity increasing with layer depth is achieved.

V. DISCUSSIONS AND CONCLUSIONS

In this paper, We proposed a method with three variants which can automatically assign specific sparsity to each layer according to the depth and the number of parameters. Under the same number of retraining steps and target sparsity, our method is simple and straightforward to take advantage of the characteristics of the network itself.

We explored the traits of CNN itself and used the different depths and different parameters for each layer to guide pruning. We found that differential pruning for convolutional layers is better than pruning without distinction. We show through our analysis that, from the perspective of the depth, layers close to the original image are more important, from the perspective of the number of parameters, layers with more parameters have more redundancy for pruning. Our work is the first to consider the depth and number of parameters of all layers of the entire network during pruning. Therefore, our proposed method with three variants provides a novel perspective for neural network acceleration. Moreover, the experimental results demonstrate the effectiveness of our scheme.

When in fact, sparse neural networks after unstructured pruning which called fine-grained pruning are usually stored with formats like Compressed Row Storage(CRS), which both values and indices are stored, the storage occupied by sparse parameter indexes needs to be taken into account when discussing compressing model size. Hence the network model size after pruning could be smaller only at the appropriate sparsity. Our method can also be applied to coarse-grained pruning, but the larger the granularity of pruning, the more difficult it is to control the accuracy. In the future, we hope to explore coarse-grained pruning. We want to get more regular structures after pruning and consider more in terms of speed and energy consumption.

REFERENCES

[1] J. Schmidhuber, "Deep learning in neural networks: An overview",Neural Netw., vol. 61, pp. 85-117, 2015.

[2] NVIDIA. Whitepaper: GPU-based deep learning inference: A performance and power analysis.

[3] A. Krizhevsky, I. Sutskever, G. E. Hinton, "Imagenet classification with deep convolutional neural networks", NIPS, 2012.

[4] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition", arXiv:1512.03385, 2015.

[5] Y. Le Cun, J.S. Denker, S.A. Solla, "Optimal brain damage" in Advances in Neural Information Processing Systems, CA, San Mateo:Morgan Kaufmann, vol. 2, pp. 598-605, 1990.

[6] S. Han, H. Mao, W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning Trained Quantization and Huff-man Coding", ICLR San Juan Puerto Rico, oct 2016.

[7] M. Zhu and S. Gupta. "To prune, or not to prune: exploring the efficacy of pruning for model compression. ", arXiv preprint arXiv:1710.01878, 2017.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, "Rethinking the inception architecture for computer vision", Proc. CVPR, pp. 2818-2826, 2016.

[9] S. Han, J. Pool, J. Tran, W. Dally, "Learning both weights and connections for efficient neural network", Proc. Adv. Neural Inf. Process. Syst., vol. 28, pp. 1135-1143, 2015.

[10] T.-J. Yang, Y.-H. Chen, V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning", Proc. CVPR, 2017.

[11] Y. Guo, A. Yao, Y. Chen, "Dynamic network surgery for efficient dnns", NIPS, 2016.

[12] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, "Pruning filters for efficient convnets", 2016.

[13] S. Anwar, W. Sung. "Compact Deep Convolutional Neural Networks With Coarse Pruning.", arXiv preprint arXiv:1512.08571, 2015.

[14] H. Mao et al., "Exploring the regularity of sparse structure in convolutional neural networks", Proc. CVPR Workshop Tensor Methods In Comput. Vis., 2017.

[15] S. Anwar, K. Hwang, W. Sung, "Structured pruning of deep convolutional neural networks", ACM J. Emerg. Technol. Comput. Syst., vol. 13, no. 3, pp. 32, 2017.

[16] Y. He, X. Zhang, and J. Sun. "Channel pruning for accelerating very deep neural networks.", ICCV, 2017.

[17] J.-H. Luo, J. Wu, and W. Lin. "Thinet: A filter level pruning method for deep neural network compression.", ICCV, 2017.

[18] W. Wen, C. Wu, Y. Wang, Y. Chen, H. Li, "Learning structured sparsity in deep neural networks", Proc. NIPS, pp. 2074-2082, 2016.

[19] M. Courbariaux, Y. Bengio, J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations", Proc. NIPS, pp. 3123-3131, 2015.

[20] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets", Proc. ICASSP, 2013.

[21] G. E. Hinton, O. Vinyals, J. Dean, "Distilling the knowledge in a neural network", NIPS Deep Learning Workshop, 2014.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", Proc. Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 248-255, 2009.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, A. Rabinovich, "Going deeper with convolutions", Proc. IEEE Conf. Comput. Vis. Pattern Recognition, 2015.

[24] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition", arXiv:1512.03385, 2015.

[25] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition", arXiv:1409.1556, 2014.

[26] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications," in ICLR, 2016.

[27] B. Hassibi and D. G. Stork, "Second order derivaties for net-work prunning: Optimal brain surgeon," in NIPS, 1993.

[28] S. Ren, K. He, R. Girshick, J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", Proc. 28th Int. Conf. Neural Inf. Process. Syst., pp. 91-99, 2015.

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, "Ssd: Single shot multibox detector", ECCV, 2016.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, "Microsoft COCO: Common objects in context", Proc. Eur. Conf. Comput. Vis., pp. 740-755, 2014.