MultiAdapt: A Neural Network Adaptation For Pruning Filters Base on Multi-layers Group

Jie Yang¹, Zhihong Xie² and Ping Li ¹*

¹ School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, Hunan, Changsha 410114, China
² Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha, Hunan, Changsha 410114, China
*Corresponding author’s e-mail: jfs@csust.edu.cn

Abstract. Deep convolutional neural networks have been widely used in various AI applications. The most advanced neural networks are becoming deeper and wider, which has caused some large convolutional neural networks to exceed the size limit of the server or application. The pruning algorithm provides a way to reduce the size of the neural network while keeping the accuracy as high as possible. The automatic progressive pruning algorithm is one of the widely used pruning algorithms. The progressive pruning algorithm prunes a certain layer of the network in each iteration to reduce the sparsity while preserving the accuracy as much as possible. In this article, we design a new automatic progressive pruning algorithm named MultiAdapt. MultiAdapt combines the combination method and the greedy algorithm. This multi-layers progressive pruning method greatly increases the search space of the greedy algorithm, making it possible to obtain a better pruning network. We use MultiAdapt to prune large neural networks VGG-16 and ResNet. The experimental results show that the MultiAdapt algorithm is better than other mainstream methods in the balance of neural network model size and accuracy. For image classification tasks on the ImageNet dataset, our method achieved 88.72% and 90.55% TOP-5 accuracy on the 50% sparsity VGG-16 and ResNet, while obtaining nearly 2×reduction in parameters and floating point numbers. The operation is reduced, and the reduction is higher than the recent popular method.

1. Introduction

In recent years, CNNs have become an indispensable part of the field of artificial intelligence because of their accuracy close to or beyond humans in some challenging tasks, such as computer vision, object detection, and natural language processing. Driven by the ever-increasing amount of data and computing power, as Alexnet has 60 million parameters, vgg-16 has more than 130 million parameters, Bert has 350 million parameters, and gpt-3 With 170 billion parameters, the deep learning model becomes larger and deeper in order to better learn from the data. The over-parameterization of the neural network model causes significant redundancy in deep learning [11]. Therefore, it is very important to design an effective deep architecture to reduce computing and memory costs. However, designing an effective neural network is a labor-intensive task. Different pruning standards can obtain different precisions, requiring many experiments and fine-tuning. At this time, how to reduce the size of these trained neural network models, so that it can give as accurate predictions as possible in various scenarios, becomes particularly important.
Faced with these challenges, more and more research work has appeared, aiming to discover methods to compress neural network models while limiting any potential loss of model quality. Compressing neural networks is to reduce the complexity of the network model. Transfer learning and pruning are both widely studied and used methods. Among them, pruning is one of the most popular methods to reduce model complexity. For a neural network model, finding the optimal pruned model is actually an optimization process. The optimal pruning scheme should take into account all the parameters, resulting in obtaining it as an NP-hard problem. Therefore, how to find a solvable method in linear time is a problem we need to consider.

Based on the granularity of pruning, there are two types: unstructured pruning and structured pruning. Structured pruning has been a hot research hot area in recent years, mainly because it is easy to implement and friendly to hardware. Unstructured pruning methods, however, cannot be supported by ready-made library, thus traditional hardware and software are not easily compatible. The magnitude of model parameters is one of the key points in most previous pruning methods. For structured pruning, norms of filters or channels in a convolutional layer have been widely accepted as a criteria to determine the importance of a parameter. [12] and [13] iteratively prune the “least important” filters, and retain the pruned model layer by layer. Meanwhile, [16] and [17] are dedicated to pruning small filters during dynamic training. The threshold which is used to determine whether to prune a filter or a channel is very important, thus has to be chosen carefully.

In reality, it is often difficult for us to know which layers and parameters should be pruned if it is not for setting measurement standards. Therefore, in the pruning algorithm, there should be a mechanism to automatically explore which layers should be pruned. NetAdapt[18] proposed an automatic and progressive pruning method. It defines a multi-dimensional target resource, and then gradually prunes it while meeting the target resource limit. However, NetAdapt is gradually pruning only one layer of the network at a time. Such a greedy algorithm may cause a certain important layer to be cut too much in the previous pruning, and the final model accuracy is not high enough, and the stability of the model is not high.

Therefore, in this article, in order to reduce the size of the model more within the tolerable range of accuracy loss, we propose a new automatic progressive pruning algorithm called MultiAdapt. Using a combination of combination methods and greedy algorithms, in each progressive pruning operation of a large-scale pre-trained neural network with limited resources, we try to prune every combination of two or more layers that can be reduced. Compare the accuracy of the training set to find the current most redundant two-layer or multi-layer network combination, and then prune them according to different filter pruning strategies, choose the most accurate, and each pruning makes the model a sparse Intermediate pruning network with large degree reduction but minimal accuracy loss. After each pruning is completed, we fine-tune the model. Repeat the above operation until the sparsity of the network reaches the final target sparsity.

Our main contribution:
- We combine the combination principle and the greedy algorithm, and propose a progressive multi-layers greedy algorithm MultiAdapt. As an automatic constrained network optimization algorithm, in order to meet the constraints while maximizing accuracy, in the process of iteratively optimizing a pre-trained neural network, the two most redundant layers are combined as the pruning object for one iteration. With the least possible loss of accuracy to achieve greater magnitude pruning.
- The pruning strategy of the internal filter in the selected two-layer combination adopts a variety of standard options to achieve the optimization of the accuracy within the group. With the multi-layers selection of the iterative process, the double optimization of the algorithm is realized.
- Experiments show that our method can achieve good results on different CNN architectures and different data sets. The sparse effect of this method is better than the general filtering and pruning methods. On the imagenet dataset, even if the sparsity rises to 87.5%, an accuracy of 87.26% can be obtained on the VGG-16 model, and an accuracy of 88.3% on the Resnet-50 model. The accuracy loss is less than 1%.
2. Related work

2.1. Pruning based on metrics

This type of method usually proposes a metric for judging whether a neuron is important, and cuts out the unimportant neurons. The neuron here can be a single weight in unstructured pruning or the entire filter in structured pruning.

Weight: The more classic method based on structured pruning is pruning filters\cite{12}, which judges the importance of filters based on L1-norm. \cite{19} pulls the absolute importance to a relative level, thinking that filters that are too similar to other filters are not important.

Activation: Network trimming \cite{20} uses the ratio of 0 in activations (Average Percentage of Zeros, APoZ) as a metric, and An Entropy-based Pruning Method \cite{21} uses information entropy for pruning.

Gradient: This type of method usually starts from Loss to find the neurons that have the least impact on the loss. The method of using Taylor expansion of the objective function can be traced back to the early 1990s, such as Lecun's Optimal Brain Damage\cite{22} and Second order derivatives\cite{23}. The most representative one in recent years is \cite{24}, which performs Taylor expansion of activation at 0 o'clock. Importance Estimation \cite{25}, replaced by the expansion of weight and added a square. A similar method \cite{26} uses Fisher information to approximate the Hessian matrix. SNIP \cite{27} directly uses the derivative to perform unstructured pruning of randomly initialized weights.

2.2. Pruning based on reconstruction error

This type of method determines which filters should be pruned by minimizing the reconstruction error of the feature output, finding the information that the current layer has no effect on the output of the subsequent network layer. ThiNet \cite{13} uses the greedy method, and Channel Pruning \cite{29} uses Lasso regression. NISP \cite{14} determines which filters need to be tailored by minimizing the reconstruction error of the penultimate layer of the network and taking into account the error accumulation of back propagation.

2.3. Pruning based on sparse training

This type of method uses training, combined with various regularizers to make the weight of the network sparse, so the values close to 0 can be cut off. Learning Structured Sparsity \cite{30} uses group Lasso for structured sparseness, including filters, channels, filter shapes, and depth. \cite{31} introduces a learnable mask and uses the APG algorithm to sparse the mask to achieve structured pruning. The idea of \cite{32} is similar, using the classic algorithm ADMM in constrained optimization to solve it. Since the output of each channel will go through BN, the scaling factor of BN can be sparsed directly. For example, \cite{33} uses L1 regularizer, and \cite{34} uses ISTA to do it. Sparse. MorphNet\cite{36} also directly uses L1 regularizer, but combines the width-multiplier in MobileNet[1][2], with the shrink-expand operation, which can better meet resource constraints.

2.4. Automated pruning towards NAS

Since AMC \cite{36}, reinforcement learning has been introduced into pruning, and the research on pruning has begun to incorporate various concepts of Auto. AutoSlim \cite{37} first trains a slimmable model (similar to SuperNet\cite{38}), and then gradually tailors the network in a greedy way.

Network Pruning \cite{39} migrated a set of NAS guides for pruning. Approximated Oracle Filter Pruning \cite{40} operates all layers of the network in parallel, using a binary search to determine the number of pruning for each layer. Fine-Grained Neural Architecture Search \cite{6} reduces the granularity of NAS to channels, including empty operations, namely pruning. In addition, there are methods\cite{3} based on information bottleneck, clustering method Centipetal SGD \cite{4}, etc.
3. Methods

3.1. Method proposed

When pruning a neural network, usually the requirement is to limit the size of the pre-trained model to a specific size while maximizing the accuracy. For example, reducing a 10GB pre-trained model to 500MB, where the 500MB is the size limit to fit in a server or an app, while achieving the accuracy as high as possible.

Given the size of a trained neural network and the pruned model, we can get the sparsity $S_0$ that we ultimately need to achieve. Assuming that $m$ is the model obtained after the initial pre-training neural network pruning, we solve the following optimization problems:

$$\max \text{Accuracy}(m),$$

$$\text{s.t. } \text{Sparsity}(m) \geq S_0. \quad (1)$$

To solve the optimization problem, we can view a convolutional model with $N$ layers as a list of layer parameters $\{W_1, W_2, \ldots, W_N\}$. The weight tensor $W_l \in \mathbb{R}^{c_l \times k_l \times k_l}$ in $l$-th convolutional layer is a set of $c_l$ filters of $c_{l-1} \times k_l \times k_l$ size each. To find the optimal model $m_{\text{opt}}$ which solves equation (1), we approximate the problem as finding the most important set of filters in each layer. In other words, we prune the unimportant filters in each layer and obtain a new model with a layer parameter of $\{W_1, W_2, \ldots, W_N\}$, as shown in formula (2).

$$\{W_1, W_2, \ldots, W_N\} = \arg \max (\{w_1, w_2, \ldots, w_N\}) \text{Accuracy}(\{W_1, W_2, \ldots, W_N\})$$

$$\text{s.t. } \text{Sparsity}(\{W_1, W_2, \ldots, W_N\}) \geq S_0 \quad (2)$$

[18] is a coordinate descent style greedy scheme to tackle this optimization problem, where it tries to prune one layer at a time and reach a desired intermediate sparsity level in each iteration. Specifically, it tries to solve optimization problem (1) into

$$\max \text{Accuracy}(m_i),$$

$$\text{s.t. } \text{Sparsity}(m_i) \geq S_i \quad (3)$$

in $i$-th iteration, where $S_i$ is the intermediate sparsity level. In each iteration, [18] first obtains $N$ pruned models by trying to prune each layer of the model, select the layer with the highest verification accuracy for pruning and proceed to the next iteration. According to equation (2), we can think of one layer as a coordinate, and NetAdapt works similar to coordinate descent, where it prunes one layer in each iteration to increase sparsity while maintaining the highest accuracy.

\[\text{Algorithm 1: MultiAdapt}\]
\[\text{Input: Pre-trained CNN } M_0 \text{ with } N \text{ layers, target sparsity } S_0, \text{ number of iteration } K, \text{ validation dataset } D\]
\[\text{Output: Pruned network } M_f\]
\[\begin{array}{l}
M_1 = M_0 \\
\text{for } i = 1, \ldots, K \text{ do} \\
\quad S_i = S_0 \cdot i / K \\
\quad \text{for } m = 1, \ldots, N - 1 \text{ do} \\
\quad \quad \text{for } n = 1, \ldots, N \text{ do} \\
\quad \quad \quad M_{(m,n,0)} = \text{Independent_Prune}(M_i, m, n, S_i) \\
\quad \quad \quad M_{(m,n,1)} = \text{Forward_Prune}(M_i, m, n, S_i) \\
\quad \quad \quad M_{(m,n,2)} = \text{Combine_Prune}(M_i, m, n, S_i) \\
\quad \quad \quad M_{(m,n,3)} = \text{Lasso_Prune}(M_i, m, n, S_i) \\
\quad \quad \quad M_{(m,n)} = \text{get_best_accuracy}(D, M_{(m,n)}) \\
\quad \quad \text{end for} \\
\quad \text{end for} \\
\quad M_i = \text{get_best_accuracy}(D, M_i) \\
\end{array}\]

$M_f = \text{FineTune } M_K \text{ with remaining parameters}$

Return $M_f$
However, neural network pruning problem is a non-convex problem, thus a simple coordinate descent style approach could result in bad performance. In NetAdapt[18], only one layer is considered heuristically in each iteration. The output model of NetAdapt[18] will be simply wrong if a less-desired layer is chosen in the middle of the pruning. Also, once a layer is chosen to prune in an iteration, all subsequent layers from $l+1$ to $N$ will be changed. This means that NetAdapt[18] is also very sensitive to the order of the layers being pruned.

Thus, we propose a new approach, MultiAdapt. In each iteration, we consider 2 or more layers at the same time and consider different pruning orders and approaches of the selected multiple layers. Doing so, we evaluate multiple candidates of layer combinations in different orders in each iteration, which greatly improve the robustness to find a more optimal pruning solution.

3.2. Framework Overview

We proposed a greedy scheme called MultiAdapt. The core idea behind MultiAdapt is to combine two different principles: combinatoric approach and greedy algorithm. For simplicity, Like [5], we introduced an adaptive pruning algorithm, we remove the filter from the convolution (CONV) or fully connected (FC) layer, and in $k$ iterations, increase the sparsity from the initial sparsity value (usually 0) to the final sparsity value $S_0$.

The specific algorithm details of MultiAdapt can be found in the pseudo code of Algorithm 1 and Figure 1. Each iteration solves Eq. 3 by reducing the number of filters in the two layers of the convolutional neural network. MultiAdapt deletes the entire filter instead of a single weight, because most platforms can take advantage of deleting the entire filter, and this strategy allows to reduce filters and feature mapping, which plays an important role in the resource consumption of [29]. Then fine-tune the simplified network in a short time to restore a certain accuracy.

MultiAdapt generates $A_k$ network proposals (the combination of the number of CONV layers and the number of FC layers) in one iteration. Each network proposal has two layers that are modified on the basis of the previous iteration. The network proposal with the highest accuracy is transferred to the next iteration (use the validation data set to evaluate each pruned model, record its accuracy, and select the highest accuracy block). In order to further improve the accuracy of the precision block, it is also important to choose a suitable filter selection strategy. When pruning the 2-layers combination on network proposals, the following pruning methods will be used: a). Independent pruning: For the 2 layers, prune filters in each layer by sum of absolute weights, regardless of the other layer. After pruned 2 layers, obtain the resulting model by combing the pruned layers. b). Forward pruning: In the 2 layers, prune the layer in the front first, let the change propagate to the network after the pruned layer, then prune the other layer. Layers are pruned by the sum of absolute weights. This is different from independent pruning when the 2 layers are subsequent. c). combine pruning: Rank the absolute sum of weights for filters in 2 layers, then prune the ones which are needed to reach the desired sparsity. d). Lasso pruning: Conduct a group lasso which allows pruning on the 2 layers only.

After using the above 4 methods, we obtain 4 networks. We also use validation accuracy to choose the best one as the candidate. When pruning a 2-layer combination, we try to prune equal numbers of parameters each layer, which can be easily translated to the number of filters needed to be pruned. Continue on the $i+1$-th iteration, and prune the 2-layer combination that achieves the highest accuracy. Finally, we fine-tune the pruned network using the original loss (without L1 regularization) to recover the performance drop due to sparsity, until convergence.
Figure 1. This work proposes an adaptive algorithm called MultiAdapt. This algorithm can automatically and gradually simplify the pre-trained network under a given resource budget (memory) until the resource requirements are met. Maximize accuracy. Experimental results show that, compared with the current advanced network simplification algorithm, MultiAdapt achieves a better precision and delay trade-off. S_0 is the final network sparsity value to be reached.

Let I_l and W_l denote the input tensor and parameters of l-th convolutional layer. Here I_l ∈ R^{c_{l-1} × h_l × w_l} has c_{l-1} channels, h_l rows and w_l columns. The weight tensor W_l ∈ R^{c_l × k_l × k_l} is a set of filters of c_{l-1} × k_l × k_l size each. This convolutional layer produces the output tensor O_l ∈ R^{c_l × h_{l+1} × w_{l+1}}, which is a set of c_l feature maps. The pre-trained model m_0 can be represented as a set of layer weights \{W_1, W_2, \ldots, W_N\}. After pruning a layer, we obtain \overrightarrow{\text{W}_l} with c_{l-1} filters, the sparsity of layer l is \frac{c_{l-1}}{c_l} and the sparsity of layer l+1 is also reduced to \frac{c_{l-1}}{c_l}. We split the whole optimization process into K iterations, where each step tries to increase the overall sparsity of the network by \frac{S_0}{K}. In other words, in iteration i we try to achieve sparsity \frac{S_0}{i} / K. The overall optimization is

\[
\max_{\{W_1, W_2, \ldots, W_N\}} \text{Accuracy}(\{W_1, W_2, \ldots, W_N\}),
\]

s. t. Sparsity(\{W_1, W_2, \ldots, W_N\}) ≥ S_0 (4)

Using a Lagrangian multiplier, we are solving:

\[
\max \text{ Accuracy}(\{W_1, W_2, \ldots, W_N\}) + \lambda(\text{Sparsity}(\{W_1, W_2, \ldots, W_N\} - S_0)) (5)
\]

4. Experiments
In this part, we first introduce the data set. In this experiment, we will use two databases (CIFAR-10 and ImageNet) to evaluate the effectiveness of MultiAdapt on the published pre-trained model. Secondly, elaborate the actual performance on the pre-training model. Finally, we compare the performance of MultiAdapt with other advanced and widely used methods.

4.1. Datasets
In this experiment, We evaluated MultiAdapt on several datasets with different scales, including CIFAR-10 and ImageNet. CIFAR-10 consists of RGB 32×32 images from 10 categories, have 50,000 training instances and 10,000 testing instances. We choose CIFAR datasets because they are complex
enough to train most of the capabilities of TensorFlow’s large-scale models. Also, they are small enough to train quickly. ImageNet is one of the largest image recognition datasets. It is a large-scale labeled image dataset organized in accordance with the WorldNet architecture through a recognition system which simulates humans. There are approximately 15 million images from 22,000 categories. There are strict manual screening and marking for each image. We tested on ImageNet to see the full potential of MultiAdapt.

| Network                  | Top 1 Accuracy | Top 5 Accuracy | FLOPs |
|--------------------------|----------------|----------------|-------|
| VGG-16 on cifar10        | 93.7%          | -              | 61.4% |
| VGG-16 on imagenet       | 68.81%         | 87.26%         | 47%   |
| ResNet-50 on imagenet    | 72.96%         | 88.30%         | 45.9% |

4.2. Method evaluation
In this part, We applied MultiAdapt on a few pre-trained networks, i.e. ResNet[8] and VGG[9]. These networks are large-scale and have superior performance and difficult to deploy on mobile applications because of their size. When pruning these networks, their performance usually degrades as well. We use accuracy as the metric to judge the performance of algorithms when changing the sparsity level during pruning. The FLOPs and accuracy of the pruned network are shown in Table 1.

4.2.1. Vgg-16 on Cifar-10
The VGG (on CIFAR-10) network has only a few parameters in the fully connected layer, so the two layers as the pruning object are selected in the convolutional layer. as the Figure 2 shows. As the proportion of pruning increases, most convolutional layers have experienced varying degrees of accuracy degradation. Validated by the cifar-10 data set, when the pruning ratio reaches 20%, conv_2 and conv_8 show a slight decrease in accuracy, and there is basically no change in the accuracy of the previous layers. When the pruning ratio reaches 40%, the accuracy of the convolutional layer that originally suffered a loss of accuracy continues to decrease, and the conv_3, conv_5, conv_6, and conv_7 start to lose sequentially. When the pruning ratio reaches 60%, the vgg-16 convolutional neural network is highly sparse, and most convolutional layers have experienced a sharp decline in accuracy, but the conv_11 still has no obvious changes. This experiment can verify that most of the redundant parameters of Vgg-16 on Cifar-10 are concentrated in the last three layers of the convolutional layer; the lower the redundant parameters, the higher the sensitivity of the layer, and the greater the accuracy loss caused by pruning. Deploying the MultiAdapt algorithm in the pre-trained neural network In order to ensure that the network size is minimized and the accuracy is maximized, when selecting the two-layer combination, the latter few layers of the experiment should be emphasized.
4.2.2. VGG-16 on imagenet
VGG-16 is a large network which was designed solely for the ImageNet dataset[9]. VGG-16 contains many convolutional layers and 3 fully connected layers. Among them, the fully connected layers have small input size and few hidden units, thus they are very small compared to the convolutional layers. Most of the parameters are in the convolutional layers, which are our main focus to prune, especially in the last three layers. Now, we evaluate the performance of the proposed algorithm MultiAdapt on the large-scale data set imagenet pruning the VGG-16 model. As shown in Figure 3.

4.2.3. ResNet on imagenet
We also discussed the performance of MultiAdapt on another powerful CNN architecture: ResNet [8]. ResNet is another powerful CNN architecture. We choose ResNet-50 as the representation of the ResNet family to test MultiAdapt.
Similar to VGG-16, in each iteration of pruning, we select the verified two layers with the most redundant parameters in the convolutional layer as a set of pruning objects, and use the four proposed criteria to select the best combination. We report the performance of MultiAdapt when pruning ResNet-50, as shown in Figure 4. When the sparsity increases from 0 to 75% iteratively by 25%, every time the iteration accuracy will not change more than 1%, and even if the sparsity iteration increases to 87.5%, the accuracy loss does not exceed 5%. Prove that our method is effective and superior in performance.

### 4.3. Comparison of different Pruning Metrics

In this section, we also compared its performance with other pruning methods when reaching the same 50% sparsity on the imagenet dataset to prove that our method is effective.

#### 4.3.1. Comparison on VGG-16 on imagenet at 50% sparsity

As shown in the results shown in Table 2. We compare our method with the other two filter lever methods under the same experimental conditions. These methods all use L2 regularization, but the choice of low-weight filters is different. MultiAdapt adopts the same automatic iteration as NetAdapt[18], but it uses two layers as a block. Choose to verify the highest precision combination.

Compared with the original source model, the pruning accuracy of the three methods has a certain degree of loss. The accuracy performance retention level is: Filter Pruning<NetAdapt<MultiAdapt, whether it is top-1 accuracy or Top-5 accuracy. The top-1 accuracy of MultiAdapt is 68.53%, the top-5 accuracy is 88.72%, and the loss is not more than 1%. The parameter levels after pruning of each method are the same, and the FLOPS reduction is still the best of our method, and the reduction rate exceeds 50%.

| Model       | top-1 accuracy | Top-5 accuracy | Params  | FLOPS   |
|-------------|----------------|----------------|---------|---------|
| Original    | 69.45%         | 89.34%         | 138.3M  | 31.02B  |
| Filter Pruning | 66.83%         | 86.50%         | 69.2M   | 18.78B  |
| NetAdapt    | 68.10%         | 88.52%         | 69.1M   | 14.53B  |
| MultiAdapt  | 68.53%         | 88.72%         | 69.2M   | 13.27B  |

#### 4.3.2. Comparison on on Resnet-50 on imagenet at 50% sparsity

Similar to the operation on the VGG-16 model, we compare several state-of-the-art pruning methods on the Resnet-50, as shown in Table 3. The selected data set is also imagenet with a sparsity of 50%.

Compared with the initial model, our method can obtain excellent overall performance in terms of accuracy and FLOP trimming. The floating point number (FLOP) of the operation is used to measure
the consumption of computing resources. In the Resnet-50 model on the imagenet at 50% sparsity data set, the parameters of each method change little, but our method has the smallest loss of accuracy, less than 1%, and the floating-point number operation drops by about 50.6%.

Table 3: Comparison on Resnet-50 on imagenet at 50% sparsity

| Model        | top-1 accuracy | Top-5 accuracy | Params | FLOPS  |
|--------------|----------------|----------------|--------|--------|
| Original     | 73.36%         | 92.17%         | 25.8M  | 7.74B  |
| Filter Pruning | 70.22%         | 86.35%         | 13.0M  | 4.86B  |
| NetAdapt     | 71.90%         | 89.84%         | 12.7M  | 4.21B  |
| MultiAdapt   | 72.73%         | 90.55%         | 12.9M  | 3.83B  |

5. Conclusion

We propose a new automated algorithm called MultiAdapt, which adapts the pre-trained network to the mobile platform given the actual resource budget. MultiAdapt can integrate multi-standard selection and multiple iterations of pruning into the optimization, each iteration will get a new more optimized intermediate compromise simplified network, achieve better neural network scale and maximize the accuracy of the generated model Trade-offs. On the imagenet dataset, experiments have proved that, even if the sparsity rises to 87.5%, an accuracy of 87.26% can be obtained on the VGG-16 model, and an accuracy of 88.3% on the Resnet-50 model. The accuracy loss is less than 1%. In this work, we aim to achieve a better balance between neural network scale and maximum accuracy; we hope that future research work will consider more layer combinations to further improve the performance of effective networks.

References

[1] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).
[2] Howard, Andrew, et al. "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation." (2018).
[3] Dai, Bin, et al. "Compressing neural networks using the variational information bottleneck." International Conference on Machine Learning. PMLR, 2018.
[4] Ding, Xiaohan, et al. "Centripetal sgd for pruning very deep convolutional networks with complicated structure." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.
[5] Zhu, Michael, and Suyog Gupta. "To prune, or not to prune: exploring the efficacy of pruning for model compression." arXiv preprint arXiv:1710.01878 (2017).
[6] Kim, Heewon, et al. "Fine-Grained Neural Architecture Search." arXiv preprint arXiv:1911.07478 (2019).
[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
[8] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[9] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
[10] Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv preprint arXiv:1510.00149 (2015).
[11] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, “Predicting parameters in deep learning,” in Advances in neural information processing systems, 2013, pp. 2148–2156.
[12] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” arXiv preprint arXiv:1608.08710, 2016.
[13] J.-H. Luo, J. Wu, and W. Lin, “Thinet: A fifilter level pruning method for deep neural network compression,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 5058–5066.

[14] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, “Nisp: Pruning networks using neuron importance score propagation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9194–9203.

[15] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, and B. Zhang, “Accelerating convolutional networks via global & dynamic filter pruning,” in IJCAI, 2018, pp. 2425–2432.

[16] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, “Soft filter pruning for accelerating deep convolutional neural networks,” arXiv preprint arXiv:1808.06866, 2018.

[17] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C.-z. Xu, “Dynamic channel pruning: Feature boosting and suppression,” arXiv preprint arXiv:1810.05331, 2018.

[18] Yang, Tien-Ju, et al. "Netadapt: Platform-aware neural network adaptation for mobile applications." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[19] He, Yang, et al. "Filter pruning via geometric median for deep convolutional neural networks acceleration." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[20] Hu H, Peng R, Tai Y W, et al. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures[J]. arXiv preprint arXiv:1607.03250, 2016.

[21] Luo J H, Wu J. An entropy-based pruning method for cnn compression[J]. arXiv preprint arXiv:1706.05791, 2017.

[22] LeCun, Yann, John Denker, and Sara Solla. "Optimal brain damage." Advances in neural information processing systems 2 (1989): 598-605.

[23] Hassibi B, Stork D. Second order derivatives for network pruning: Optimal brain surgeon[J]. Advances in neural information processing systems, 1992, 5: 164-171.

[24] Molchanov P, Tyree S, Karras T, et al. Pruning convolutional neural networks for resource efficient inference[J]. arXiv preprint arXiv:1611.06440, 2016.

[25] Molchanov, Pavlo, et al. "Importance estimation for neural network pruning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[26] Theis, Lucas, et al. "Faster gaze prediction with dense networks and fisher pruning." arXiv preprint arXiv:1801.05787 (2018).

[27] Lee, Namhoo, Thalaiyasingam Ajanthan, and Philip HS Torr. "Snip: Single-shot network pruning based on connection sensitivity." arXiv preprint arXiv:1810.02340 (2018).

[28] Yang, Tien-Ju, Yu-Hsin Chen, and Vivienne Sze. "Designing energy-efficient convolutional neural networks using energy-aware pruning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[29] He, Yihui, Xiangyu Zhang, and Jian Sun. "Channel pruning for accelerating very deep neural networks." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[30] Wen, Wei, et al. "Learning structured sparsity in deep neural networks." Advances in neural information processing systems 29 (2016): 2074-2082.

[31] Huang, Zehao, and Naiyan Wang. "Data-driven sparse structure selection for deep neural networks." Proceedings of the European conference on computer vision (ECCV). 2018.

[32] Zhang, Tianyun, et al. "A systematic dnn weight pruning framework using alternating direction method of multipliers." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[33] Liu, Zhuang, et al. "Learning efficient convolutional networks through network slimming." Proceedings of the IEEE International Conference on Computer Vision. 2017.
[34] Ye, Jianbo, et al. "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers." arXiv preprint arXiv:1802.00124 (2018).

[35] Gordon, Ariel, et al. "Morphnet: Fast & simple resource-constrained structure learning of deep networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[36] He, Yihui, et al. "Amc: Autorml for model compression and acceleration on mobile devices." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[37] Yu, Jiahui, and Thomas Huang. "AutoSlim: Towards One-Shot Architecture Search for Channel Numbers." arXiv preprint arXiv:1903.11728 (2019).

[38] Cai, Han, et al. "Once-for-all: Train one network and specialize it for efficient deployment." arXiv preprint arXiv:1908.09791 (2019).

[39] Dong, Xuanyi, and Yi Yang. "Network pruning via transformable architecture search." Advances in Neural Information Processing Systems. 2019.

[40] Ding, Xiaohan, et al. "Approximated oracle filter pruning for destructive cnn width optimization." arXiv preprint arXiv:1905.04748 (2019).