Headline

大家好：

这是**2018年度第3篇Arxiv Weekly**。

本文是 网络压缩 方向的文章。

Highlight

> 通过**channel wise**的网络软剪枝保留更多网络描述能力，从而在同样的**pruning rate**下取得更高的**performance**！

Information

Title

Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks

Link

https://www.ijcai.org/proceedings/2018/0309.pdf

Codes

https://github.com/he-y/soft-filter-pruning

Accepted by

2018年度人工智能国际联合大会（International Joint Conference on Artificial Intelligence，IJCAI 2018）[为人工智能方向A类会]

Source

- 南方科技大学（Southern University of Science and Technology）
- 悉尼科技大学（University of Technology Sydney）
- 复旦大学（Fudan University）

Introduction

本文提出了一种通过pruning加速inference的方法：卷积核软剪枝（soft filter pruning ,SFP），并且在pruning后网络还能进一步训练。也就是说，**pruning**的过程是伴随着训练进行的。

SFP有两个主要的优势：

1. 保留更多模型性能。由于pruning后还能够训练，SFP能够保留更多模型性能，学习到更好的特征。

2. 对pre-trained model的依赖性更低。SFP能够同时完成training和pruning，而不是像之前的pruning一样需要在well pre-trained model上进行。

Keys

1.问题描述

传统的filter pruning技术有两个“流派”，分别存在一些问题。

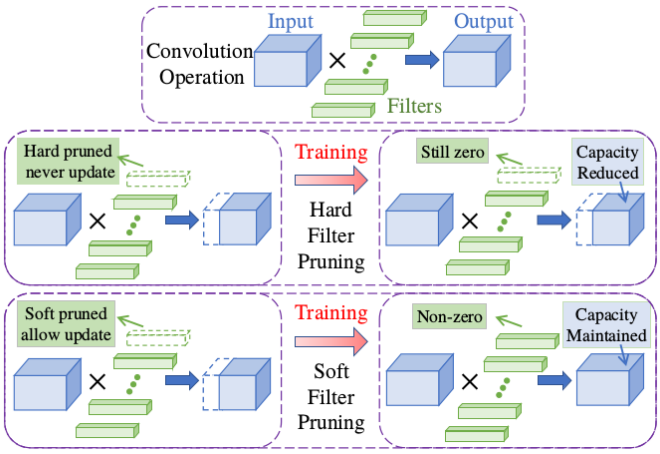
- 细粒度剪枝（**Fine-grained pruning**），指元素级的pruning，将值较小的或者经过测试不重要的elements置零。
- 粗粒度/结构化剪枝（**Coarse-grained / structured pruning**），指成块地整体剪枝。最常见的就是channel-wise pruning，也即一次剪掉一个channel。另外也有一次减掉一行、一列、一个小块这样的形式。

显然细粒度剪枝的可操作性更大，因此通常能够实现同样performance下更高的压缩率。

然而由于计算架构的优化现状，粗粒度剪枝（尤其是channel-wise pruning）相比细粒度剪枝而言计算效率更高。[例如Han、Guo等人提出的element-wise pruning会把模型弄得“千疮百孔”，导致需要稀疏编码和解码，并且针对完整filter的通用优化计算全会失效。]

而本文提出的SFP属于粗粒度剪枝中的channel-wise pruning的最新工作。相比于前序工作，主要解决了如下两个问题：

- model capacity reduction 传统的channel-wise pruning实质上是hard pruning，也即设定压缩率把完整的模型打残，让残废的模型自己往终点爬。[也即模型的描述能力受到了断崖式损伤] 而SFP则是把模型实时打残实时上药，而且看不行了就扶一下。[在训练过程中保留模型的全部描述能力，只不过限制描述的效果] 因此虽然大家最后都是残废了，但是SFP的方式打残的模型能跑得更远。[如下图]

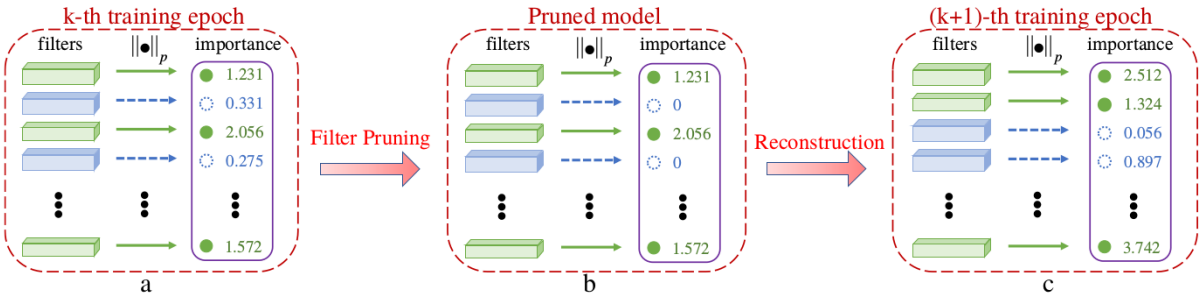


- dependence on pre-trained models。一般的hard pruning需要对完整模型的充分训练结果作为输入。而SFP由于是一边训练一边pruning，对pre-trained model没有需求，也自然不会被pre-trained model的性能影响。

2.SFP思路和算法

SFP的关键思路可以总结如下：

1. 整个网络设置一个压缩率$P%$，目标是把每个layer的filter数都压到原本的(1-P%)倍。
2. 于是在正常的训练每个epoch的训练后，都对每层各个filter中l_2范数最小的$P%$个强行置零。但是仍然允许它们参与下一轮的训练和参数更新（称为Reconstruction）。这样一来，其中有一些在后续迭代中较为重要的filter还有机会“重出江湖”。[如下图]



SFP的算法伪代码如下图所示

Algorithm 1 Algorithm Description of SFP

Input: training data: $\mathbf{X}$, pruning rate: P_i
the model with parameters $\mathbf{W} = \{\mathbf{W}^{(i)}, 0 \leq i \leq L\}$.
Initialize the model parameter $\mathbf{W}$
for $epoch = 1; epoch \leq epoch_{max}; epoch++$ **do**
  Update the model parameter $\mathbf{W}$ based on $\mathbf{X}$
  for $i = 1; i \leq L; i++$ **do**
    Calculate the ℓ_2-norm for each filter $\|\mathcal{F}_{i,j}\|_2, 1 \leq j \leq N_{i+1}$
    Zeroize $N_{i+1}P_i$ filters by ℓ_2-norm filter selection
  end for
end for
Obtain the compact model with parameters $\mathbf{W}^*$ from $\mathbf{W}$
Output: The compact model and its parameters $\mathbf{W}^*$

其中

$$\|\mathcal{F}_{i,j}\|_p = \sqrt[p]{\sum_{n=1}^{N_i}\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}|\mathcal{F}_{i,j}(n,k_1,k_2)|^p}, \qquad (2)$$

p.s.在整个网络训练结束后，将最后一轮SFP中pruning掉的filter直接干掉就能得到实际的小模型了。这里会存在一个前后对应的问题，而这个问题在ResNet结构的网络中会变得尤其突出，甚至影响到了SFP的实际pruning rate和效果对比可靠性。

3. SFP的压缩效率理论值

我们设第i个layer输出的feature map size为$C_i \times W_i \times H_i$，设第i个layer中filter的pruning rate为P_i，显然有： 在pruning之前计算第i+1层的feature map的计算量为： $$Cal=N_{i+1}\times N_i \times k^2 \times H_{i+1} \times W_{i+1}$$ 而经过SPF后的计算量为： $$Cal=(1-P_{i+1})N_{i+1}\times (1-P_i)N_i \times k^2 \times H_{i+1} \times W_{i+1}$$ 也即这一层的压缩率为： $$1-(1-P_i)(1-P_{i+1})$$ 如果整体SPF只有一个pruning rate P，则整体理论压缩率为 $$Ratio=1-(1-p)^2=2p-p^2$$

Results

在CIFAR上超过了很多传统的pruning算法。

在ImageNet+ResNet的标准benchmark上，基本能稳住Acc掉1个点的时候，prune掉40%左右的flops。 其中最好的一个case是ImageNet+ResNet101 ，在prune掉42.2%的flops时，反而使得top1/top5 Acc上升了0.14/0.2个点。

结果大表如下

Depth	Method	Fine-tune?	Baseline Accu. (%)	Accelerated Accu. (%)	Accu. Drop (%)	FLOPs	Pruned FLOPs(%)
20	[Dong *et al.*, 2017a]	N	91.53	91.43	0.10	3.20E7	20.3
	Ours(10%)	N	**92.20 ± 0.18**	**92.24 ± 0.33**	**-0.04**	3.44E7	15.2
	Ours(20%)	N	**92.20 ± 0.18**	91.20 ± 0.30	1.00	2.87E7	29.3
	Ours(30%)	N	**92.20 ± 0.18**	90.83 ± 0.31	1.37	**2.43E7**	**42.2**
32	[Dong *et al.*, 2017a]	N	92.33	90.74	1.59	4.70E7	31.2
	Ours(10%)	N	**92.63 ± 0.70**	**93.22 ± 0.09**	**-0.59**	5.86E7	14.9
	Ours(20%)	N	**92.63 ± 0.70**	90.63 ± 0.37	0.00	4.90E7	28.8
	Ours(30%)	N	**92.63 ± 0.70**	90.08 ± 0.08	0.55	**4.03E7**	**41.5**
56	[Li *et al.*, 2017]	N	93.04	91.31	1.75	9.09E7	27.6
	[Li *et al.*, 2017]	Y	93.04	93.06	-0.02	9.09E7	27.6
	[He *et al.*, 2017]	N	92.80	90.90	1.90	-	50.0
	[He *et al.*, 2017]	Y	92.80	91.80	1.00	-	50.0
	Ours(10%)	N	**93.59 ± 0.58**	**93.89 ± 0.19**	**-0.30**	1.070E8	14.7
	Ours(20%)	N	**93.59 ± 0.58**	93.47 ± 0.24	0.12	8.98E7	28.4
	Ours(30%)	N	**93.59 ± 0.58**	93.10 ± 0.20	0.49	7.40E7	41.1
	Ours(30%)	Y	**93.59 ± 0.58**	93.78 ± 0.22	-0.19	7.40E7	41.1
	Ours(40%)	N	**93.59 ± 0.58**	92.26 ± 0.31	1.33	**5.94E7**	**52.6**
	Ours(40%)	Y	**93.59 ± 0.58**	93.35 ± 0.31	0.24	**5.94E7**	**52.6**
110	[Li *et al.*, 2017]	N	93.53	92.94	0.61	1.55E8	38.6
	[Li *et al.*, 2017]	Y	93.53	93.30	0.20	1.55E8	38.6
	[Dong *et al.*, 2017a]	N	93.63	93.44	0.19	-	34.2
	Ours(10%)	N	**93.68 ± 0.32**	93.83 ± 0.19	-0.15	2.16E8	14.6
	Ours(20%)	N	**93.68 ± 0.32**	**93.93 ± 0.41**	**-0.25**	1.82E8	28.2
	Ours(30%)	N	**93.68 ± 0.32**	93.38 ± 0.30	0.30	**1.50E8**	**40.8**
	Ours(30%)	Y	**93.68 ± 0.32**	93.86 ± 0.21	-0.18	**1.50E8**	**40.8**

CIFAR上几种方法对不同深度的ResNet进行pruning的结果对比

Depth	Method	Fine-tune?	Top-1 Accu. Baseline(%)	Top-1 Accu. Accelerated(%)	Top-5 Accu. Baseline(%)	Top-5 Accu. Accelerated(%)	Top-1 Accu. Drop(%)	Top-5 Accu. Drop(%)	Pruned FLOPs(%)
18	[Dong *et al.*, 2017a]	N	69.98	66.33	89.24	86.94	3.65	2.30	34.6
	Ours(30%)	N	**70.28**	**67.10**	**89.63**	**87.78**	**3.18**	**1.85**	**41.8**
34	[Dong *et al.*, 2017a]	N	73.42	**72.99**	91.36	**91.19**	**0.43**	**0.17**	24.8
	[Li *et al.*, 2017]	Y	73.23	72.17	-	-	1.06	-	24.2
	Ours(30%)	N	**73.92**	71.83	**91.62**	90.33	2.09	1.29	**41.1**
50	[He *et al.*, 2017]	Y	-	-	92.20	90.80	-	1.40	**50.0**
	[Luo *et al.*, 2017]	Y	72.88	72.04	91.14	90.67	**0.84**	**0.47**	36.7
	Ours(30%)	N	**76.15**	**74.61**	**92.87**	**92.06**	1.54	0.81	41.8
	Ours(30%)	Y	**76.15**	62.14	**92.87**	84.60	14.01	8.27	41.8
101	Ours(30%)	N	**77.37**	77.03	**93.56**	93.46	0.34	0.10	**42.2**
	Ours(30%)	Y	**77.37**	**77.51**	**93.56**	**93.71**	**-0.14**	**-0.20**	**42.2**

ImageNet上几种方法对不同深度的ResNet进行pruning的结果对比

我们自己也有相应的复现结果，总体来看是不错的，但是对比上还需要进一步的实验。

Insights

可以看到，这里的“soft” filter pruning，主要“软”在不在最开始就把网络弄小，也不在最终定型了之后一股脑儿压缩，而是一边训练一边约束。

比较有意思的是，在约束了之后，模型的结构和filter真正的层数是没有改变的，被prune掉的filter后面还有“复活”的希望。因此pruning操作对模型的描述能力的损伤显然是比硬pruning小的。

不过这样的pruning方法在ResNet结构上会面临两个支路挑选的channel不能重合的问题，需要进一步计算实际的pruning rate。