

- Headline
- Highlight
- Information
  - \* Title
  - \* Link
  - \* Source
  - \* Publication
- Introduction
- Keys
  - \* 1. 问题
  - \* 2. 背景
  - \* 1.  $L_1$ -norm based channel pruning。
  - \* 2. ThiNet
  - \* 3. regression based Feature Reconstruction
  - \* 4. Network Slimming
  - \* 5. Sparse Structure Selection
  - \* 6. Non-structured Weight Pruning
- Experiments
  - \* 1. 实验分类
  - \* 2. 训练细节
  - \* 3. 实验结果
- Further Exploration
  - \* 1. 对结果的猜想
  - \* 2. 猜想的观察性验证
  - \* 3. 猜想的设计性验证
  - \* 4. 本文结果的局限性
- Insights
  - \* 1. 文章的现实价值
  - \* 2. 关于Pruning和NAS方向融合趋势的一点分析

# Headline

大家好：

这是2018年度第8篇Arxiv Weekly。

本文是 模型压缩 方向的文章。

# Highlight

对网络压缩技术的探究性工作，指出了几个旧有的错误印象，得出了新的指导性结论，能够辅助优化网络压缩过程

# Information

## Title

*Rethinking the Value of Network Pruning*

## Link

<https://arxiv.org/pdf/1810.05270.pdf>

## Source

- 加州伯克利大学 (University of California, Berkeley)
- 清华大学 (Tsinghua University)

## Publication

Under review as a conference paper at International Conference on Machine Learning (ICLR) 2019

## Introduction

神经网络压缩作为实现神经网络小型化的主流技术之一，引发了广泛的关注。传统的神经网络压缩算法架构主要是三段式的：训练大型网络->剪枝->**finetune**。其核心是在剪枝过程中设计某种策略或者评价标准，使得不重要的参数被剪掉；而重要的参数被保留。从而实验压缩模型、acc基本不降低。

本文中通过实验发现了“反常识”的现象。首先我们通过对6个state-of-the-art的pruning算法的实验，发现对训练好的大网络进行剪枝-**finetune**，其性能不如对剪枝后的网络进行重训（此时用random initialization）来得好。故而如果剪枝算法中假设了目标网络的结构，那么之前复杂的pipeline没有太大的意义。

通过对多个剪枝算法、多个数据集、多个任务的交叉组合验证，我们发现了如下的现象/结论：

- 获取高效的小网络 不需要 训练一个大规模的、参数/描述能力过冗余的网络
- 经过剪枝的小网络参数 不需要 参考充分训练获得的原网络的对应参数。
- 一个经过剪枝的小网络是否高效，核心不在于保留了大网络中的哪些参数，而是剪枝后是什么结构。传统的剪枝pipeline之所以work，不是因为挑选和保留了大网络中足够重要的参数，而是这个挑选的过程耦合了一个网络结构搜索过程。换言之，**pruning其实是一种NAS技术**。

# Keys

## 1. 问题

在传统上，我们认为神经网络的“over parameterization”是显然的和需要解决的。而pruning则是解决这个问题的最直接手段之一。

而在之前的观念里，以及之前pruning相关的文章中，训练出一个准确的，强大的大网络对剪枝后的网络很重要。因为小网络的参数基本是使用大网络剪枝后获得的参数进行finetune得到的，而这么做是因为主流认为大网络经过充分训练能够得到强有力的参数，值得继承。

本文认为不然。

本文的核心想法是，pruning后小网络效果好，不依赖于大网络的参数，而依赖于小网络的结构本身。大网络参数在完成了“挑选小网络结构特性”的使命后就成为了累赘，没有太大保留价值。

## 2. 背景

为了证明这个观点，本文参考了六种最新的pruning算法/架构作为实验的背景。现列举如下：

### 1. *L<sub>1</sub>-norm based channel pruning*。

这篇文章是ICLR 2017的会议文章，也是最早提出做channel pruning的文章。[关于weight/channel/layer pruning，可参考之前的Arxiv Weekly 2018/003中的论述]

本文作为channel pruning的开山之作，自然使用的是很简洁的方式。具体来说就是使得每一层当中 $L_1$ 范数最小的某个比例  $p$  的channel被剪枝。

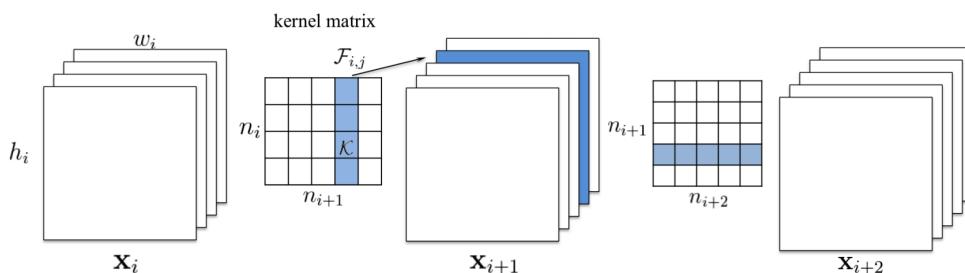


Figure 1: Pruning a filter and its corresponding feature map. We remove the kernels and feature maps and do not introduce any sparsity in the network.

可以看到，channel pruning对计算量的节省是直接有效的，对第*i*个kernel剪枝一个通道将节省 $r_i$ 次计算。

$$r_i = n_i k^2 h_{i+1} w_{i+1} + n_{i+2} k^2 h_{i+2} w_{i+2}$$

在具体操作的时候，channel pruning还会一道一些问题。

首先是选择channel to prune。本文采用了所有pruning文章几乎通用的一个思路，就是prune掉不敏感(sensitive) 的那部分channel。敏感性的衡量方式有很多种，其中最consuming的就是分别剪枝训练后之间看掉点的数目。而本文在各个layer之间的L1 sum分布和敏感性之间建立了对应关系，故而不用训练就能知道个大概。

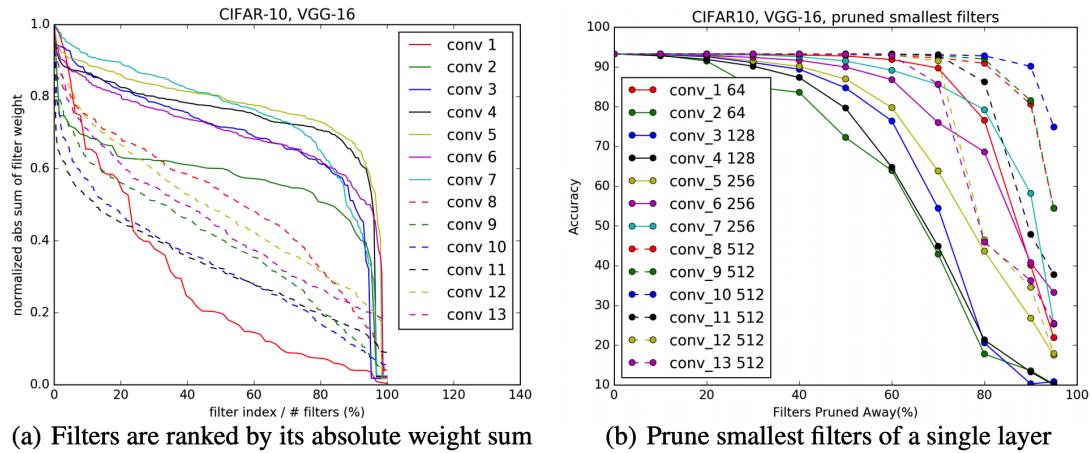


Figure 2: (a) Sorted filter weight sum for each layer of VGG-16 on CIFAR-10. The x-axis is the filter index divided by the total number of filters; The y-axis is the filter weight sum divided by the max sum value among filters in that layer. (b) Pruning filters with lowest absolute weights sum and their corresponding test accuracy on CIFAR-10.

另一个问题是channel pruning的一个细节。也即计算后续pruning的过程中是否考虑前面pruning带来的影响。

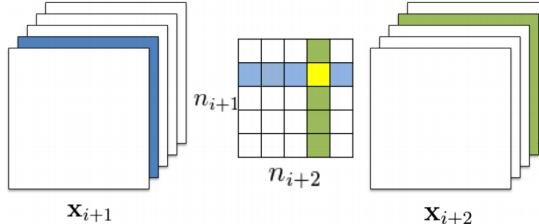


Figure 3: Pruning filters across consecutive layers. The independent pruning strategy calculates the filter sum (columns marked in green) without considering feature maps removed in previous layer (shown in blue), so the kernel weight marked in yellow is still included. The greedy pruning strategy considers the pruned filters in previous layers and does not count kernels from the already pruned feature maps (shown in yellow). Both approaches result in a  $(n_{i+1} - 1) \times (n_{i+2} - 1)$  kernel matrix.

最后值得一提的是，在channel pruning当中，由于pruning操作会影响下一轮feature map的大小，故在resnet等存在分枝的结构中，需要注意对应关系。本文认为Identity path相比Residual path更加重要，故而在冲突的时候以Identity path的选择为准。

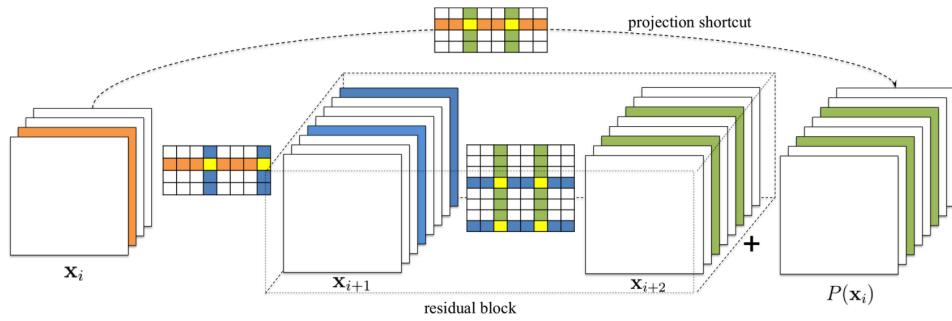
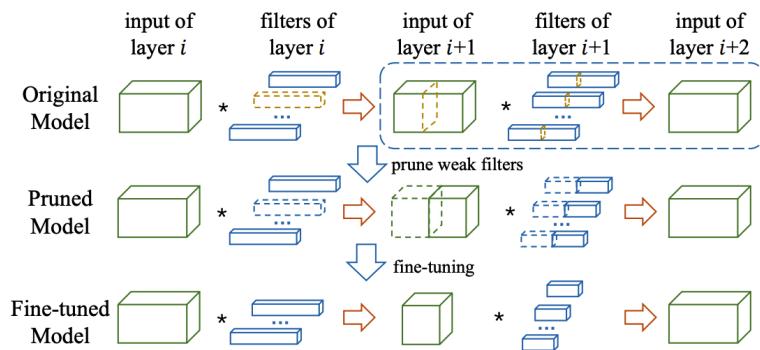


Figure 4: Pruning residual blocks with projection shortcut. The filters to be pruned for the second layer of the residual blocks (marked as green) is determined by the pruning result of the shortcut projection. The first layer of the residual blocks can be pruned without restrictions.

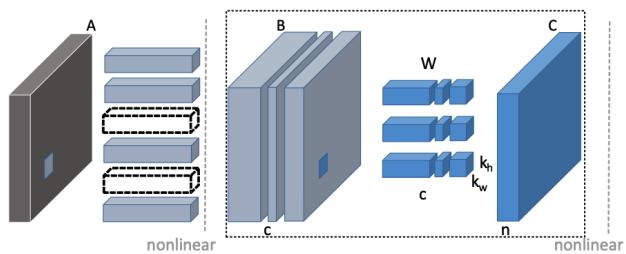
## 2. ThiNet

这篇文章是ICCV 2017的会议文章，思路更进一步。同样按比例剪枝channel，但是criterion改为优先选择下一层的feature map中参数最小的channel对应的卷积核进行剪枝。从直觉上，这样会更加靠近我们pruning同时不影响acc性能的意图。



## 3. regression based Feature Reconstruction

这篇文章是ICCV 2017的会议文章，思路更进一步。同样按比例剪枝channel，但是criterion改为尽量使得剪枝后下下层的feature map受影响最小。显然这又进一步试图保留原网络的性能。



## 4. Network Slimming

这篇文章是ICCV 2017的会议文章，与前面文章不同，本文是在训练过程中自动产生每个层的pruning rate，所以结构不经过训练无法得到。

实际上，本文采用的方式是引用BN层的scaler  $\alpha$ 作为scaling factor，然后动态剪枝去除其中不重要的channel。因此是一种动态稀疏化网络的操作。

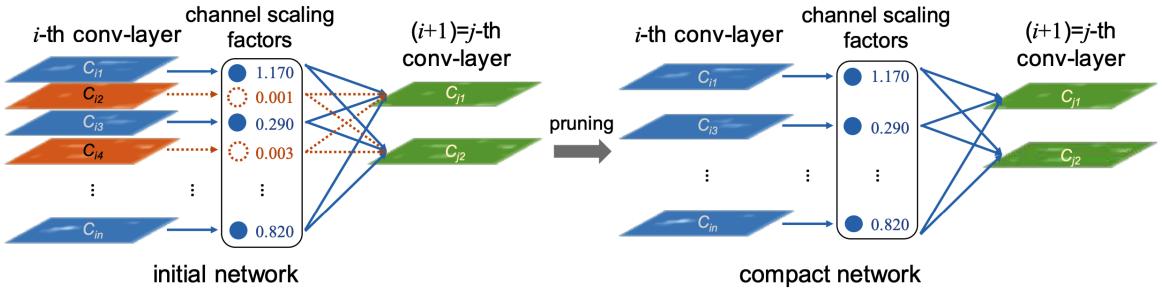


Figure 1: We associate a scaling factor (reused from a batch normalization layer) with each channel in convolutional layers. Sparsity regularization is imposed on these scaling factors during training to automatically identify unimportant channels. The channels with small scaling factor values (in orange color) will be pruned (left side). After pruning, we obtain compact models (right side), which are then fine-tuned to achieve comparable (or even higher) accuracy as normally trained full network.

## 5. Sparse Structure Selection

这篇文章是ECCV 2018会议文章。文章是Network Slimming的变体，仍然是生成了scaling factor指导pruning，只不过不是channel pruning而是block pruning。有些像[BlockDrop: Dynamic Inference Paths in Residual Networks](#)一文，当然，本文没有引入Reinforcement Learning。

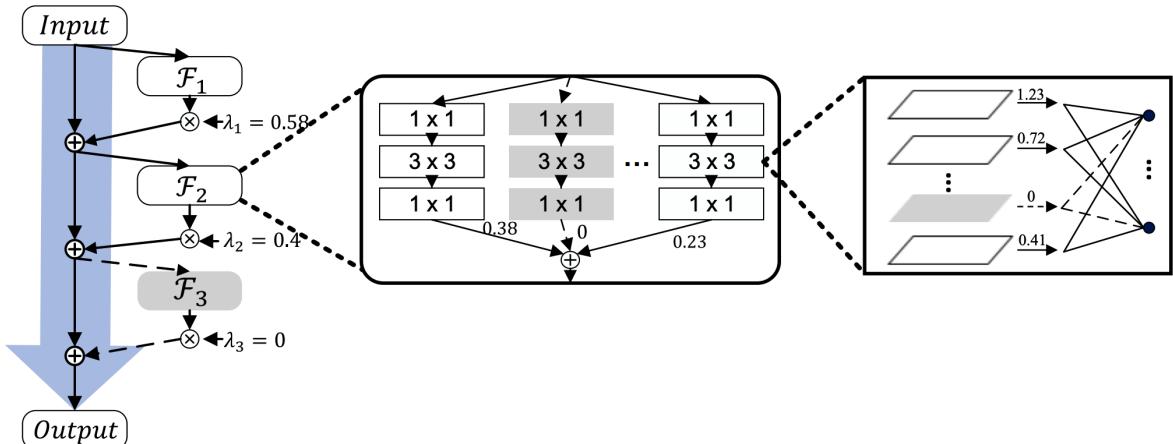


Fig. 1: The network architecture of our method.  $\mathcal{F}$  represents a residual function. Gray block, group and neuron mean they are inactive and can be pruned since their corresponding scaling factors are 0.

## 6. Non-structured Weight Pruning

这篇文章为NIPS 2015会议文章，韩松代表作之一，引用已经800+。核心方法是“啥也不说，就是硬干”，直接进行element-wise pruning，然后finetune。后续还有所谓的DSD等方法，其实就是反复硬干。至于pruning rate就是手工大量实验堆叠出一个最佳值即可。

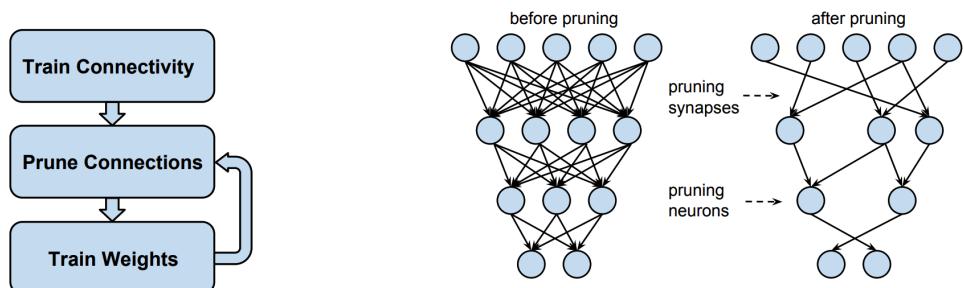


Figure 2: Three-Step Training Pipeline.

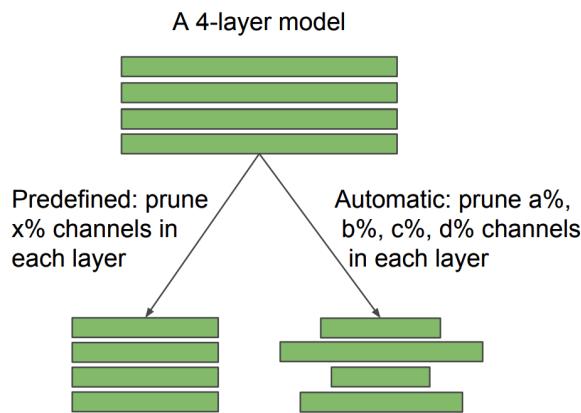
Figure 3: Synapses and neurons before and after pruning.

# Experiments

## 1. 实验分类

本文属于探究性质的论文，所以实验的作用是探索和证明文章的观点。

文章把6个baseline网络分为两类，一类是所谓的Pre-defined Target Architecture（前三个实验）；另一类是所谓的Automatically Discovered Target Architecture（后三个实验）。其区别就是不训练能不能画出来大概的结构，如下图所示：



**Figure 2:** Difference between predefined and non-predefined (automatically discovered) target architectures. The sparsity  $x$  is user-specified, while  $a, b, c, d$  are determined by the pruning algorithm.

## 2. 训练细节

在训练参数上，文章采用了CIFAR10/100和ImageNet的经典参数，和VGG、ResNet、DenseNet的经典backbone，pruning method就是前文所述的6个。

需要特别注意的一点是，文章对 **training budget** 重点进行了测试。所谓的**training budget**也即当模型压缩后，进行重新训练的时候，是按照相同的epochs数训还是按照相同的FLOPs训。[文章认为至少要允许进行同FLOPs的训练才有公平的比较可能。故文章中所有的重训网络都有两组对照，也即Scratch-E和Scratch-B，前者保证训练epochs数相同；后者保证训练FLOPs数相同。](#)

[关于这一点，李翔在Arxiv Insights中推送了他们的网络压缩经验，佐证了小网络确实需要更多的epochs进行迭代，保持FLOPs基本不变，才能收敛到最好的结果。]

另外值得一提的是，文章复现了6个benchmark，对于文章中没有提到训练参数的情况都重新跑了。并且diss了一些文章作者，表示复现的大网络性能明显好于文章中提到的性能，也即怀疑原文为了凸显pruning的性能压低了大网络的性能。具体没跑，不得而知。

## 3. 实验结果

1. [L<sub>1</sub>-norm based channel pruning](#)。

Dataset	Model	Unpruned	Pruned Model	Fine-tuned	Scratch-E	Scratch-B
CIFAR-10	VGG-16	93.63 ( $\pm 0.16$ )	VGG-16-A	93.41 ( $\pm 0.12$ )	93.62 ( $\pm 0.11$ )	<b>93.78</b> ( $\pm 0.15$ )
	ResNet-56	93.14 ( $\pm 0.12$ )	ResNet-56-A	92.97 ( $\pm 0.17$ )	92.96 ( $\pm 0.26$ )	<b>93.09</b> ( $\pm 0.14$ )
			ResNet-56-B	92.67 ( $\pm 0.14$ )	92.54 ( $\pm 0.19$ )	<b>93.05</b> ( $\pm 0.18$ )
ImageNet	ResNet-110	93.14 ( $\pm 0.24$ )	ResNet-110-A	93.14 ( $\pm 0.16$ )	<b>93.25</b> ( $\pm 0.29$ )	93.22 ( $\pm 0.22$ )
			ResNet-110-B	92.69 ( $\pm 0.09$ )	92.89 ( $\pm 0.43$ )	<b>93.60</b> ( $\pm 0.25$ )
ImageNet	ResNet-34	73.31	ResNet-34-A	72.56	72.77	<b>73.03</b>
			ResNet-34-B	72.29	72.55	<b>72.91</b>

**Table 1:** Results (accuracy) for  $L_1$ -norm based channel pruning (Li et al., 2017). “Pruned Model” is the model pruned from the large model. Configurations of Model and Pruned Model are both from the original paper.

## 2. ThiNet

Dataset	Unpruned	Strategy			Pruned Model
		VGG-Conv	VGG-GAP	VGG-Tiny	
ImageNet	VGG-16				
	71.03	Fine-tuned	-1.23	-3.67	-11.61
	71.51	Scratch-E	-2.75	-4.66	-14.36
		Scratch-B	<b>+0.21</b>	<b>-2.85</b>	<b>-11.58</b>
	ResNet-50		ResNet50-30%	ResNet50-50%	ResNet50-70%
	75.15	Fine-tuned	-6.72	-4.13	-3.10
	76.13	Scratch-E	-5.21	-2.82	-1.71
		Scratch-B	<b>-4.56</b>	<b>-2.23</b>	<b>-1.01</b>

**Table 2:** Results (accuracy) for ThiNet (Luo et al., 2017). Names such as “VGG-GAP” and “ResNet50-30%” are pruned models whose configurations are defined in Luo et al. (2017). To accommodate the effects of different frameworks between our implementation and the original paper’s, we compare relative accuracy drop from the unpruned large model. For example, for the pruned model VGG-Conv, -1.23 is relative to 71.03 on the left, which is the reported accuracy of the unpruned large model VGG-16 in the original paper; -2.75 is relative to 71.51 on the left, which is VGG-16’s accuracy in our implementation.

## 3. regression based Feature Reconstruction

Dataset	Unpruned	Strategy	Pruned Model
			VGG-16-5x
ImageNet	71.03	Fine-tuned	-2.67
		Scratch-E	-3.46
	71.51	Scratch-B	<b>-0.51</b>
	ResNet-50		ResNet-50-2x
	75.51	Fine-tuned	-3.25
	76.13	Scratch-E	-1.55
		Scratch-B	<b>-1.07</b>

**Table 3:** Results (accuracy) for Regression based Feature Reconstruction (He et al., 2017b). Pruned models such as “VGG-16-5x” are defined in He et al. (2017b). Similar to Table 2, we compare relative accuracy drop from unpruned large models.

## 4. Network Slimming

Dataset	Model	Unpruned	Prune Ratio	Fine-tuned	Scratch-E	Scratch-B
CIFAR-10	VGG-19	93.53 ( $\pm 0.16$ )	70%	93.60 ( $\pm 0.16$ )	93.30 ( $\pm 0.11$ )	<b>93.81</b> ( $\pm 0.14$ )
	PreResNet-164	95.04 ( $\pm 0.16$ )	40%	94.77 ( $\pm 0.12$ )	94.70 ( $\pm 0.11$ )	<b>94.90</b> ( $\pm 0.04$ )
			60%	94.23 ( $\pm 0.21$ )	94.58 ( $\pm 0.18$ )	<b>94.71</b> ( $\pm 0.21$ )
CIFAR-100	DenseNet-40	94.10 ( $\pm 0.12$ )	40%	94.00 ( $\pm 0.20$ )	93.68 ( $\pm 0.18$ )	<b>94.06</b> ( $\pm 0.12$ )
	VGG-19	72.63 ( $\pm 0.21$ )	60%	<b>93.87</b> ( $\pm 0.13$ )	93.58 ( $\pm 0.21$ )	93.85 ( $\pm 0.25$ )
			50%	72.32 ( $\pm 0.28$ )	71.94 ( $\pm 0.17$ )	<b>73.08</b> ( $\pm 0.22$ )
ImageNet	PreResNet-164	76.80 ( $\pm 0.19$ )	40%	76.22 ( $\pm 0.20$ )	76.36 ( $\pm 0.32$ )	<b>76.68</b> ( $\pm 0.35$ )
			60%	74.17 ( $\pm 0.33$ )	75.05 ( $\pm 0.08$ )	<b>75.73</b> ( $\pm 0.29$ )
	DenseNet-40	73.82 ( $\pm 0.34$ )	40%	<b>73.35</b> ( $\pm 0.17$ )	73.24 ( $\pm 0.29$ )	73.19 ( $\pm 0.26$ )
	60%	72.46 ( $\pm 0.22$ )	72.62 ( $\pm 0.36$ )	<b>72.91</b> ( $\pm 0.34$ )		
ImageNet	VGG-11	70.84	50%	68.62	70.00	<b>71.18</b>

**Table 4:** Results (accuracy) for Network Slimming (Liu et al., 2017). “Prune ratio” stands for total percentage of channels that are pruned in the whole network. The same ratios for each model are used as the original paper.

## 5. Sparse Structure Selection

Dataset	Model	Unpruned	Pruned Model	Pruned	Scratch-E	Scratch-B
ImageNet	ResNet-50	76.12	ResNet-41	75.44	75.61	<b>76.17</b>
			ResNet-32	74.18	73.77	<b>74.67</b>
			ResNet-26	71.82	72.55	<b>73.41</b>

**Table 5:** Results (accuracy) for residual block pruning using Sparse Structure Selection (Huang & Wang, 2018). In the original paper no fine-tuning is required so there is a “Pruned” column instead of “Fine-tuned” as before.

## 6. Non-structured Weight Pruning

Dataset	Model	Unpruned	Prune Ratio	Fine-tuned	Scratch-E	Scratch-B
CIFAR-10	VGG-19	93.50 ( $\pm 0.11$ )	30%	93.51 ( $\pm 0.05$ )	<b>93.71</b> ( $\pm 0.09$ )	93.31 ( $\pm 0.26$ )
			80%	93.52 ( $\pm 0.10$ )	<b>93.71</b> ( $\pm 0.08$ )	93.64 ( $\pm 0.09$ )
	PreResNet-110	95.04 ( $\pm 0.15$ )	30%	95.06 ( $\pm 0.05$ )	94.84 ( $\pm 0.07$ )	<b>95.11</b> ( $\pm 0.09$ )
CIFAR-100	DenseNet-BC-100	95.24 ( $\pm 0.17$ )	30%	95.21 ( $\pm 0.17$ )	95.22 ( $\pm 0.18$ )	<b>95.23</b> ( $\pm 0.14$ )
			80%	95.04 ( $\pm 0.15$ )	94.42 ( $\pm 0.12$ )	<b>95.12</b> ( $\pm 0.04$ )
	VGG-19	71.70 ( $\pm 0.31$ )	30%	71.96 ( $\pm 0.36$ )	72.81 ( $\pm 0.31$ )	<b>73.30</b> ( $\pm 0.25$ )
ImageNet	PreResNet-110	76.96 ( $\pm 0.34$ )	30%	76.88 ( $\pm 0.31$ )	76.36 ( $\pm 0.26$ )	<b>76.96</b> ( $\pm 0.31$ )
			50%	<b>76.60</b> ( $\pm 0.36$ )	75.45 ( $\pm 0.23$ )	76.42 ( $\pm 0.39$ )
	DenseNet-BC-100	77.59 ( $\pm 0.19$ )	30%	77.23 ( $\pm 0.05$ )	77.58 ( $\pm 0.25$ )	<b>77.97</b> ( $\pm 0.31$ )
	VGG-16	71.59	30%	73.68	72.75	<b>74.02</b>
			60%	<b>73.63</b>	71.50	73.42
	ResNet-50	76.15	30%	<b>76.06</b>	74.77	75.70
	60%	<b>76.09</b>	73.69	74.91		

**Table 6:** Results (accuracy) for non-structured pruning (Han et al., 2015). “Prune Ratio” denotes the percentage of parameters pruned in the set of all convolutional weights.

其结论基本是统一的，就是或者Scratch-E，或者Scratch-B，能够持平或者超出利用原模型参数finetune的结果。

除了在ImageNet上进行实验6时，SongHan的原始方案高于Scratch-B/E。

# Further Exploration

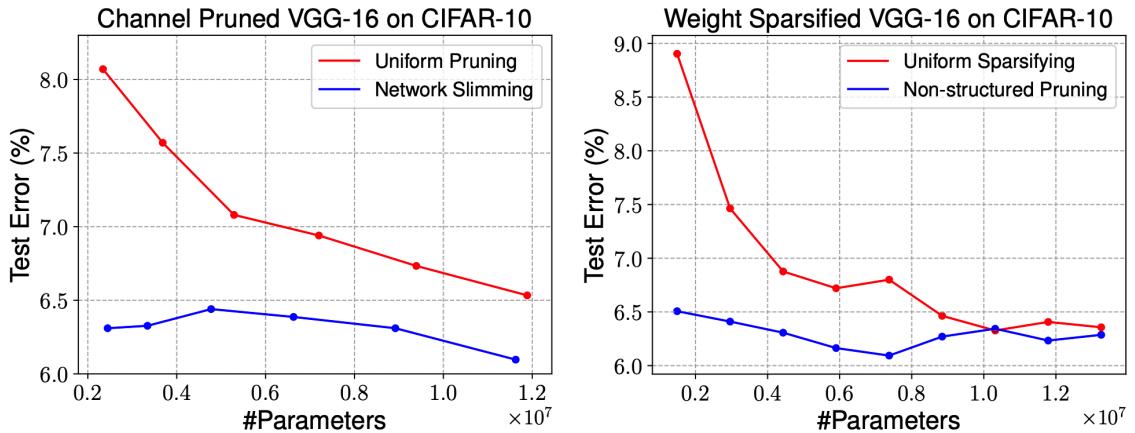
# 1. 对结果的猜想

文章对实验验证进行了进一步的分析探究。也即解释既然传统的pruning没有能够真正选出“有意义”的最佳参数，为什么还是非常work。

主要的解释是，pruning操作天然的是非常高效和合理的网络结构搜索方案，能够得出更优化的小网络结构，因此做到了“模型压缩，精度不降”。

## 2. 猜想的观察性验证

为了验证这个想法，作者测试了“胡乱压缩”和用验证有效的压缩策略压缩，带来的“网络参数效率”的不同，如下图所示：

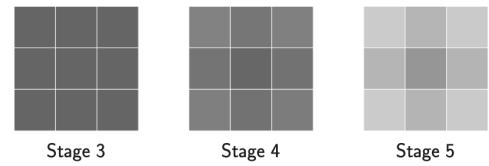


**Figure 3:** Pruned architectures obtained by different approaches, all *trained from scratch*, averaged over 5 runs. Architectures obtained by automatic pruning methods (*Left*: Network Slimming (Liu et al., 2017), *Right*: Non-structured weight pruning (Han et al., 2015)) have better parameter efficiency than uniformly pruning channels or sparsifying weights in the whole network.

另外，作者发现经过优秀的压缩pipeline得到的小网络结构，其实是倾向于“有规律可循”的，也就是说其实压缩类似于一个天然的小网络设计过程，设计出来的东西符合一定的规律，而不是乱七八糟无章可循。

Layer	Width	Width*	Layer	Width	Width*
1	64	39.0±3.7	8	512	217.3±6.6
2	64	64.0±0.0	9	512	120.0±4.4
3	128	127.8±0.4	10	512	63.0±1.9
4	128	128.0±0.0	11	512	47.8±2.9
5	256	255.0±1.0	12	512	62.0±3.4
6	256	250.5±0.5	13	512	88.8±3.1
7	256	226.0±2.5	Total	4224	1689.2

**Table 8:** Network architectures obtained by pruning 40% channels on VGG-16 (in total 13 conv-layers) using Network Slimming. Width and Width\* are number of channels in the original and pruned architectures, averaged over 5 runs.



**Figure 4:** The average sparsity pattern of all  $3 \times 3$  convolutional kernels in certain layer stages in a non-structured pruned VGG-16. Darker color means higher probability of weight being kept.

值得注意的是，文章中给出了上图所示的element-wise pruning最终的结果。[可以看到随着迭代次数的增多和层数的加深， \$3 \times 3\$ 的卷积核渐渐退化成了对称的“十字架”的模样](#)。启发我们可以设计类似模样的“异形卷积核”或者“渐变卷积核”来减少网络的计算量同时保持网络性能。

### 3. 猜想的设计性验证

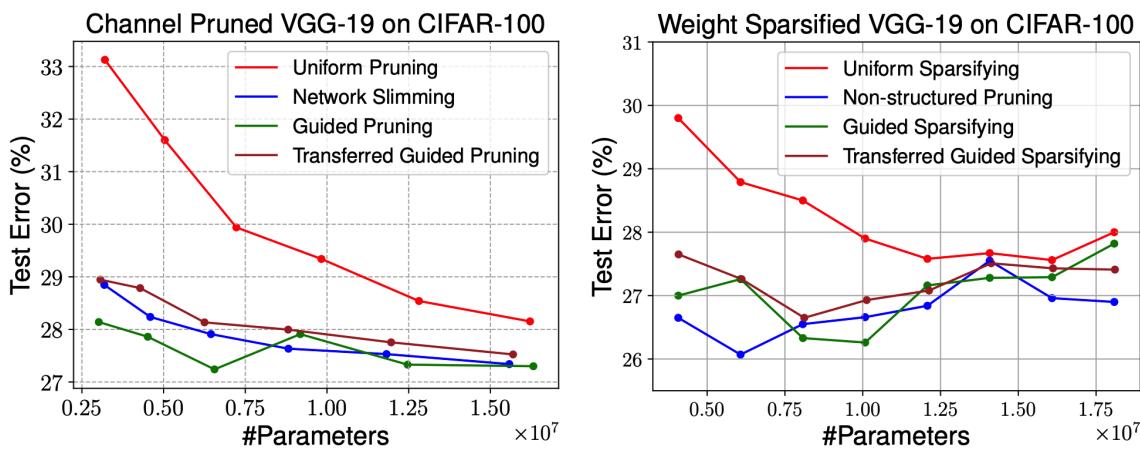
为了进一步证明上述说法是成立的，作者利用上面发现的规律，基于VGG19和CIFRA100，设计了一些小网络进行测试。具体来说：

- 设计Guided Pruning系列网络，方式为基于Network Slimming方法先得到一个pruning好的网络，然后统计每个“layer stage”[也即下图中的同颜色layer]中的平均channel剪枝数。利用这个剪枝比例直接对VGG19进行重新剪枝得到的网络即为Guided Pruning Network



- 设计Guided Sparsify系列网络，方式为基于Non-structured Weight Pruning方法观察pruning结束后卷积核的平均强度[如上上图中Figure4所示]。然后根据这个强度拟合一个大概规律直接搭建稀疏化的网络结构，称为Guided Sparsify Network。
- 设计transferred Guided系列对比网络。这是基于一个有趣的观察，**同一个系列网络经过pruning后会得到相似的规律，这样的规律有可迁移性**。故设计实验利用VGG16+CIFAR10进行训练+pruning+获得Guiding超参数，然后在VGG19上根据超参数进行网络搭建，并在CIFRA100上测试。

其结果如下图所示：



**Figure 5:** Pruned architectures obtained by different approaches, *all trained from scratch*, averaged over 5 runs. “Guided Pruning/Sparsifying” means using the average sparsity patterns in each layer stage to design the network; “Transferred Guided Pruning/Sparsifying” means using the sparsity patterns obtained by a pruned VGG-16 on CIFAR-10, to design the network for VGG-19 on CIFAR-100. Following the design guidelines provided by the pruned architectures, we achieve better parameter efficiency, even when the guidelines are transferred from another dataset and model.

可以看到，不但Guided系列的网络能够达到和pruning pipeline获得的最优网络相似的性能，连Transferred系列的网络也几部不分上下。

这进一步有力佐证了pruning其实是一种提取网络结构信息，给定条件在大网络上搜索最佳小网络的方法。而不是给定大网络学习到的参数信息，从其中搜索关键信息剔除无关信息的类似蒸馏的方法。

## 4.本文结果的局限性

当然，作者也列举了**本文提出方法的不可用场景**（也即传统pruning pipeline的不可替代性）

- 当已经给出了完备的pre-trained network，不用白不用，而且训练新网络的计算资源受限的时候。
- 当有必要把一个网络结构进行多个不同pruning rate的压缩的时候，用一个训好的大网络进行不同pruning rate的压缩操作最高效。

[实际上，本文也不以提出解决方案为核心，而是以认知的更新为核心，所以这样的局限性并无关紧要。]

# Insights

## 1.文章的现实价值

本文的结论对现实训练还是有不少直接价值的：

1. 进行网络压缩的时候没必要一定训练一个完美的大网络。而且在经过剪枝得到小网络后，至少要重训一下网络再做出性能的论断。不能把finetune的结果就当做小网络性能的最终结果。
2. 可以通过pruning操作的结果观察网络结构当中起作用的主要部分，从而指导下一步的结构优化设计。例如十字架形状的卷积核、漏斗状的block等。
3. 对于得到的小网络，如果重训，需要设置对照组，保证和大网络的FLOPs几乎持平，也即需要用更多的epochs进行一组训练。
4. 其实channel pruning的作用和SE十分类似，只不过SE是以涨点为目的，pruning是以小型化为目的，但是仔细思考手段和思路其实是十分相似的，也许可以相互借鉴。**例如利用下一个layer的权重甚至是下下个layer的改变量判断这层卷积核的重要性就是很值得尝试的命题。**

## 2.关于Pruning和NAS方向融合趋势的一点分析

不难看到，pruning和NAS在渐渐向着互相结合彼此参考的方向发展，这方面也可能会给我们后续的研究工作一些参考和突破的可能。

众所周知传统的NAS一般基于强化学习（Reinforcement Learning, RL）或者进化算法（Evolutionary Algorithm, EA）。在复杂度上显然超过pruning算法的几个数量级。

而近来：

- [CVPR2018: MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks](#)一文当中引入了类似Network Slimming的结构进行网络自动化设计；

- [ICML2018: Efficient neural architecture search via parameter sharing](#)等文章也利用大网络参数继承的方式提高NAS的效率。这显然是传统pruning思路应用到NAS方向的体现。

另一方面，我们之前也提到过，利用RL来进行pruning就在18年9月份左右大大火了一把，谷歌、腾讯、MIT（韩松组）等机构都推出了基于RL的网络自动剪枝算法/平台。我们还在[Arxiv Weekly 2018/004](#)中推送过韩松组的相关工作，大家可以回去翻翻回忆一下。