Deep Learning

Structured Pruning

郑煜伟
zheng.yuwei@foxmail.com

Network Pruning

01 Why & What

For an efficient & suitable network

02 How

APoZ, Group LASSO, Gamma factor, Scalable factor …

03 Rethink

Initial weight? Over-parameterized Structure? Sparsity?

/01

Why & What

Kind of neural architecture search but more operable

To obtain an efficient & suitable network

Why network pruning

- 大多时候，深度神经网络结构都是过参数化的，但实际数据集都比较小；
- **高效网络**：精度提高或基本持平，但存储、FLOPs更低；
- **自动化**：人工设计复杂：人工成本、计算成本、时间成本，且极具经验性；
- **结构适配**：结构搜索，结构正则；

- 低秩近似
- 量化
- 蒸馏
- 编码

What

- 2 categories roughly:
  - Unstructured pruning (Connection level);
  - Structured pruning: **Filter level** & Layer level.
- Channel Magnitude， Reconstruction， Loss sensitivity

2017

Title
Pruning Filters for Efficient ConvNets
Pruning Convolutional Neural Networks for Resource
Net-Trim: Convex Pruning of Deep Neural Networks
Learning to Prune Deep Neural Networks via Layer-w
Runtime Neural Pruning
Designing Energy-Efficient Convolutional Neural Net
ThiNet: A Filter Level Pruning Method for Deep Neur
Channel pruning for accelerating very deep neural ne
Learning Efficient Convolutional Networks Through N

2016

Title
Deep Compression: Compressing Deep Neural Netw Quantization and Huffman Coding
Dynamic Network Surgery for Efficient DNNs

2015

Title
Learning both Weights and Connections for Efficient

2018

Title
Rethinking the Smaller-Norm-Less-Informativ Convolution Layers
To prune, or not to prune: exploring the effica
Discrimination-aware Channel Pruning for De
Frequency-Domain Dynamic Pruning for Con
Amc: Automl for model compression and acc
Data-Driven Sparse Structure Selection for De
Coreset-Based Neural Network Compression
Constraint-Aware Deep Neural Network Com
A Systematic DNN Weight Pruning Framewor Multipliers
PackNet: Adding Multiple Tasks to a Single N
NISP: Pruning Networks using Neuron Import
CLIP-Q: Deep Network Compression Learning
“Learning-Compression” Algorithms for Neur
Soft Filter Pruning for Accelerating Deep Con

2019

Title	Venue	Type	Code
Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration	CVPR **(Oral)**	F	github
Towards Optimal Structured CNN Pruning via Generative Adversarial Learning	CVPR	F	github
Centripetal SGD for Pruning Very Deep Convolutional Networks with Complicated Structure	CVPR	F	github
On Implicit Filter Level Sparsity in Convolutional Neural Networks, Extension1, Extension2	CVPR	F	github
Structured Pruning of Neural Networks with Budget-Aware Regularization	CVPR	F	-
Importance Estimation for Neural Network Pruning	CVPR	F	github
OICSR: Out-In-Channel Sparsity Regularization for Compact Deep Neural Networks	CVPR	F	-
Partial Order Pruning: for Best Speed/Accuracy Trade-off in Neural Architecture Search	CVPR	Other	github
Variational Convolutional Neural Network Pruning	CVPR	-	-
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks	ICLR **(Best)**	W	github
Rethinking the Value of Network Pruning	ICLR	F	github
Dynamic Channel Pruning: Feature Boosting and Suppression	ICLR	F	github
SNIP: Single-shot Network Pruning based on Connection Sensitivity	ICLR	F	github
Dynamic Sparse Graph for Efficient Deep Learning	ICLR	F	github
Collaborative Channel Pruning for Deep Networks	ICML	F	-
Approximated Oracle Filter Pruning for Destructive CNN Width Optimization github	ICML	F	-
EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis4	ICML	W	github

/02

How

Reconstruction: Minimize reconstruction loss

Feature Maps Reconstruction[1]

- 2步重构特征层：
  - LASSO regression based channel selection：基于LASSO回归，保留最具代表性的channel。
  - Least square reconstruction：剪枝后，基于线性最小二乘误差重构输出。

- 然后逐层剪枝、优化重构误差，降低误差累积的影响。

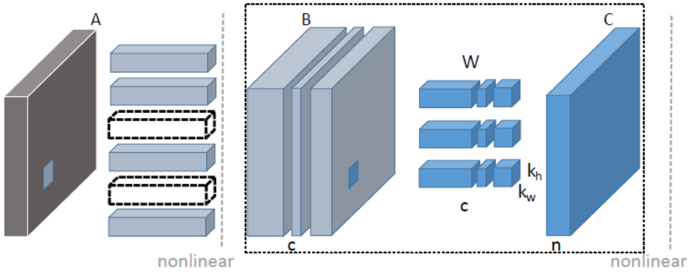


Figure 2. Channel pruning for accelerating a convolutional layer. We aim to reduce the number of channels of feature map B, while minimizing the reconstruction error on feature map C. Our optimization algorithm (Sec. 3.1) performs within the dotted box, which does not involve nonlinearity. This figure illustrates the situation that two channels are pruned for feature map B. Thus corresponding channels of filters W can be removed. Furthermore, even though not directly optimized by our algorithm, the corresponding filters in the previous layer can also be removed (marked by dotted filters). c, n: number of channels for feature maps B and C, $k_h \times k_w$: kernel size.

[1] Channel Pruning for Accelerating Very Deep Neural Networks, 2017, ICCV

Feature Maps Reconstruction

- Whole Goal:（一开始说两步交叉迭代，最后为了训练简单，多次运行i，最后运行一次ii，表示效果和迭代差不多；并不重构整个feature map，而是5000 images都采样10 neurons重构）

$$\arg_{\beta,W}\min\frac{1}{2N}\left\|Y-\sum_{i=1}^{c}\beta_i X_i W_i^T\right\|_F^2$$
$$subject\ to\ \|\beta\|_0\le c'$$

- 1. （β 优化）LASSO regression based channel selection

$$\hat{\beta}^{LASSO}(\lambda)=\arg_{\beta}\min\frac{1}{2N}\left\|Y-\sum_{i=1}^{c}\beta_i Z_i\right\|_F^2$$
$$subject\ to\ \|\beta\|_0\le c', \forall i\ \|W_i\|_F=1$$

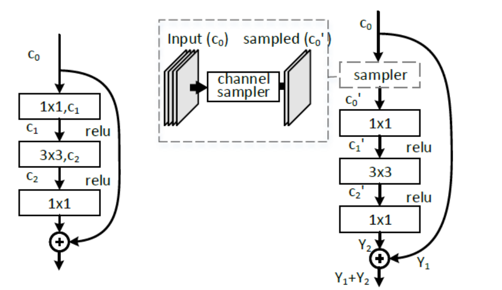


Figure 3. Illustration of multi-branch enhancement for residual block. **Left**: original residual block. **Right**: pruned residual block with enhancement, c_x denotes the feature map width. Input channels of the first convolutional layer are sampled, so that the large input feature map width could be reduced. As for the last layer, rather than approximate Y_2, we try to approximate $Y_1 + Y_2$ directly (Sec. 3.3 Last layer of residual branch).

- 2. （W优化）Least square reconstruction

$$\arg_{W'}\min\|Y-X'(W')^T\|_F^2$$

- 针对shortcut：
  - 第一层添加一个sampler，做通道保留；
  - 最后一层则重构残差部分：$Y_1-Y_1'+Y_2$，Y_1+Y_2为原特征，Y_1'为前面逐层剪枝后的shortcut特征。

Experiments

- 两种保留channel机制：first k， max response[31] （权重的L1范数）
  - Max response有时比first k还差：channel相关性很重要；
  - 浅层的冗余性大一点，越深越难剪：浅层多剪一些（没有量化指导）；

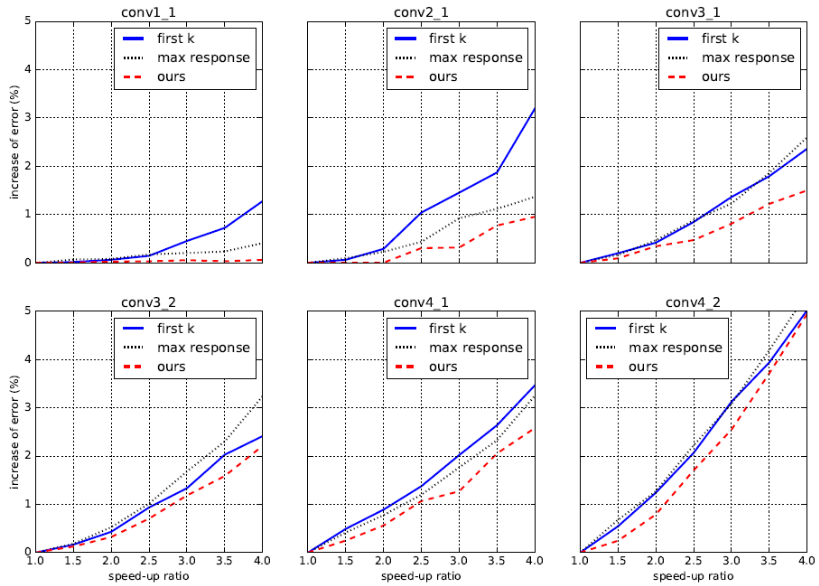


Figure 4. Single layer performance analysis under different speed-up ratios (without fine-tuning), measured by increase of error. To verify the importance of channel selection refered in Sec. 3.1, we considered two naive baselines. *first k* selects the first k feature maps. *max response* selects channels based on absolute sum of corresponding weights filter [31]. Our approach is consistently better (*smaller is better*).

Increase of top-5 error (1-view, baseline 89.9%)			
Solution	2×	4×	5×
Jaderberg *et al.* [22] ([53]'s impl.)	-	9.7	29.7
Asym. [53]	0.28	3.84	-
Filter pruning [31] (fine-tuned, our impl.)	0.8	8.6	14.6
Ours (without fine-tune)	2.7	7.9	22.0
Ours (fine-tuned)	0	1.0	1.7

Table 1. Accelerating the VGG-16 model [44] using a speedup ratio of 2×, 4×, or 5× (*smaller is better*).

[31] Pruning Filters For Efficient ConvNets, 2017, ICLR

Experiments

- 在同样4x加速的基础上，做以下实验：
  - 用LASSO regression剪枝后的模型，重头训练得到的模型；
  - 用每个channel均匀剪枝后的模型，重头训练得到的模型；
  - 用LASSO regression剪枝后的模型；
  - 用LASSO regression剪枝后微调的模型；
- 1. model exploration尚有探索空间；
- 2. 初始权重很重要；

Original (acc. 89.9%)	Top-5 err.	Increased err.
From scratch	11.9	1.8
From scratch (uniformed)	12.5	2.4
Ours	18.0	7.9
Ours (fine-tuned)	11.1	**1.0**

Table 4. Comparisons with training from scratch, under $4\times$ acceleration. Our fine-tuned model outperforms scratch trained counterparts (*smaller is better*).

Discussion and Conclusion

- 剪枝效果好
- 适用各种结构的模型；
- 模型训练完后，剪枝只需要少量样本便可进行，而后有多量样本还可以选择finetune；

- 针对每层逐层、渐进式剪枝，**没有综合全局信息**进行剪枝
- 需要**预定义网络结构**：每个层需要保留多少个channel？虽然指导说剪浅层容易点

/02

How

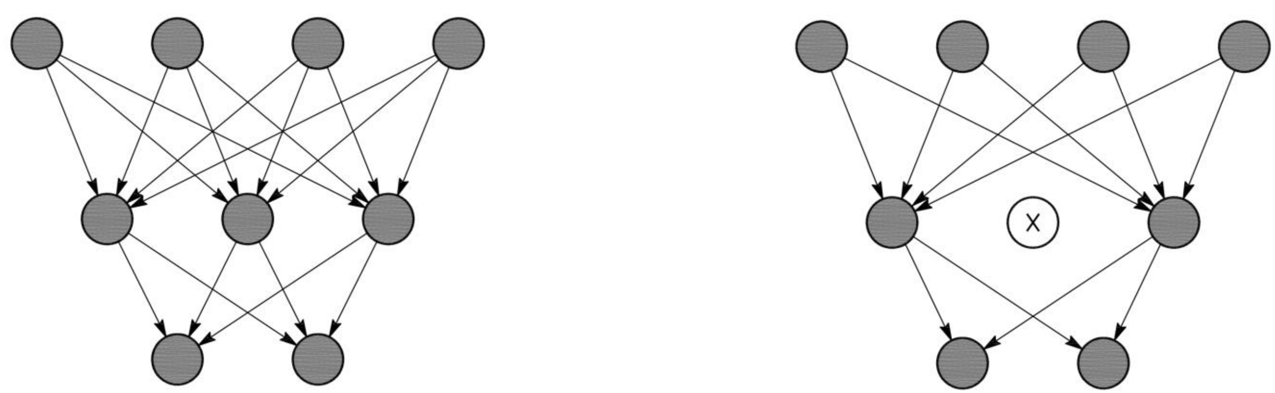
Magnitude: Prune Zero Activations

Zero Activations and Network Trimming[1]

- Average Percentage of Zeros (APoZ)：一个经ReLU的channel，其神经元平均失活率

$$APoZ_c^{(i)} = APoZ\left(O_c^{(i)}\right) = \frac{\sum_k^N \sum_j^M f\left(O_{c,j}^{(i)}(k) = 0\right)}{N \times M}$$

- 其中，c是channel，i是layer，N是验证集样本数，M是神经元数量（feature map维度）。
- 用APoZ评估网络中 neuron/channel 的重要性。



[1] Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures, 2016, ICLR

APoZ of VGG-16

Table 1: Mean APoZ of each layer in VGG-16

Layer	CONV1-1	CONV1-2	CONV2-1	CONV2-2	CONV3-1
Mean APoZ (%)	47.07	31.34	33.91	51.98	47.93
Layer	CONV3-2	CONV3-3	CONV4-1	CONV4-2	CONV4-3
Mean APoZ (%)	48.84	69.93	65.33	70.76	87.30
Layer	CONV5-1	CONV5-2	CONV5-3	FC6	FC7
Mean APoZ (%)	76.51	79.73	93.19	75.26	74.14

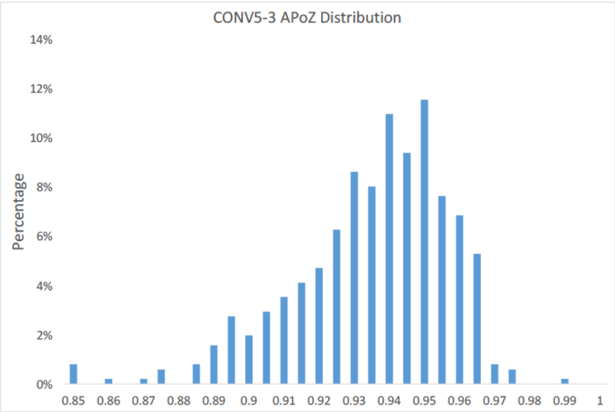


Figure 1: CONV5-3 APoZ Distribution

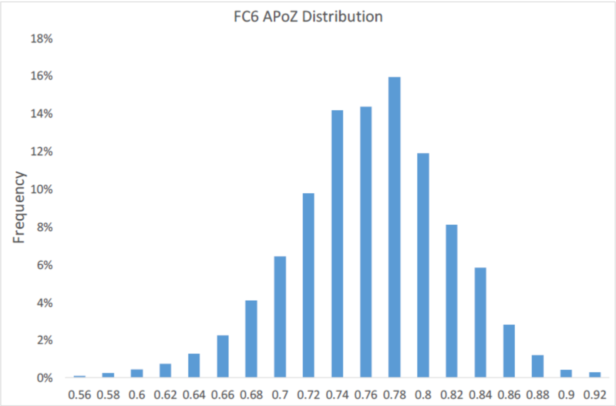
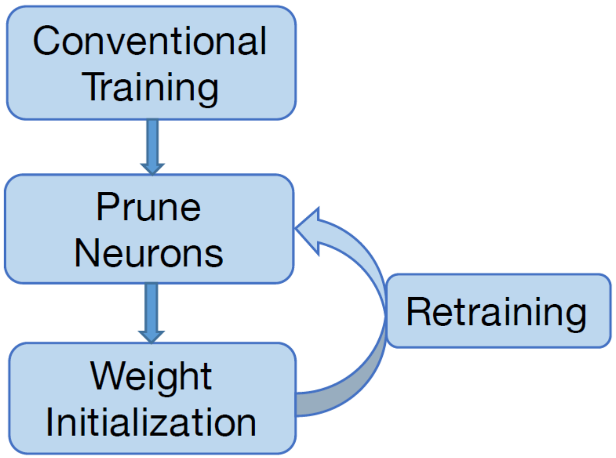
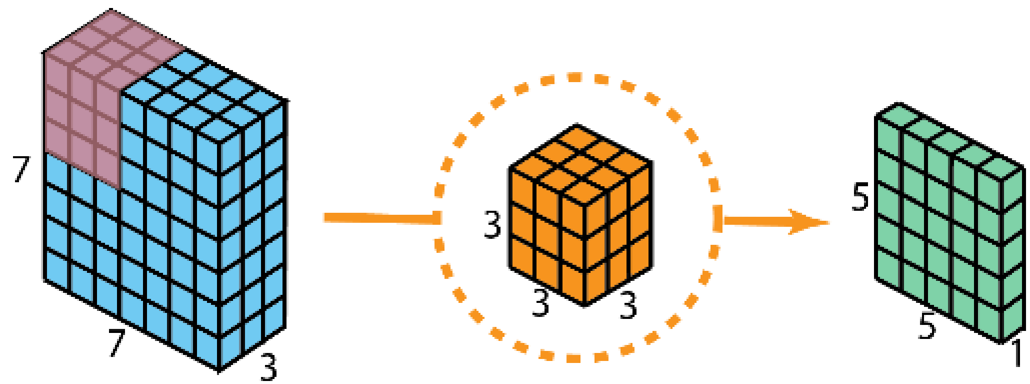


Figure 2: FC6 APoZ Distribution

Network Trimming and Retaining

- 训练三部曲：训练→剪枝→保留权重重新训练





用APoZ剪filter

- 剪枝后，保留权重，而不是重头开始重新训练：neurons更少失活，更高效
- 一次剪枝太多影响网络性能，所以要迭代进行
- 剪枝APoZ高于平均值一个标准差的neurons（约16%）

Experiments

- LeNet（20-50-500-10，2 conv + 2 full），VGG-16 (2~3× less) ref 原论文

Table 2: Iterative Trimming on LeNet

Network Config	Compression Rate	Initial Accuracy (%)	Final Accuracy (%)
(20-50-500-10)	1.00	10.52	99.31
(20-41-426-10)	1.41	98.75	99.29
(20-31-349-10)	2.24	95.34	99.30
(20-26-293-10)	3.11	88.21	99.25
(20-24-252-10)	3.85	96.75	99.26

Table 3: Iterative Trimming on LeNet with and without Weight Initialization

	With Weight Init			Without Weight Init		
Number of Neurons in FC1	500	426	349	500	420	303
Accuracy (%)	99.31	99.29	99.30	99.31	99.23	99.18
Mean APoZ (%)	52.30	45.85	42.70	52.30	55.08	55.08
#{APoZ>0.6}	154	77	17	154	160	110
#{APoZ>0.7}	87	10	0	87	102	78
#{APoZ>0.8}	54	0	0	54	61	49
#{APoZ>0.9}	33	0	0	33	40	32

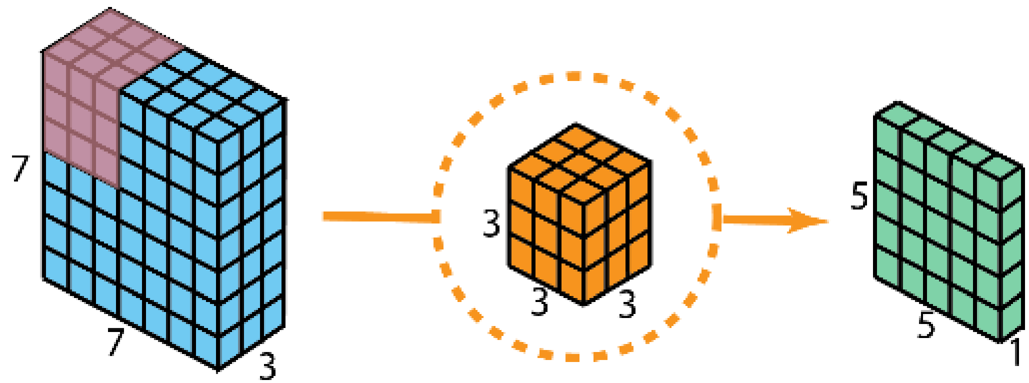
Discussion and Conclusion

- Connection Pruning:
  - 剪neurons比剪connections高效;
  - 剪conv layer的channel、full layer的neurons，适用性广，应用性强。
- 计算APoZ所用的数据集
  - 论文用validation set评估APoZ剪枝，然后用train set训练模型，迭代进行。这波渗透验证集的操作，有模型过拟合的风险。
  - 但是作者实验发现trimmed network可以收缩验证误差和测试误差间的gap：
    - 未剪枝：11.56% val，13.02% test
    - 剪枝：9.7% val， 10.02% test
  - 作者用train set评估APoZ进行剪枝，与用validation set的情况，剪枝weak neurons集中存在着95%的交叉。

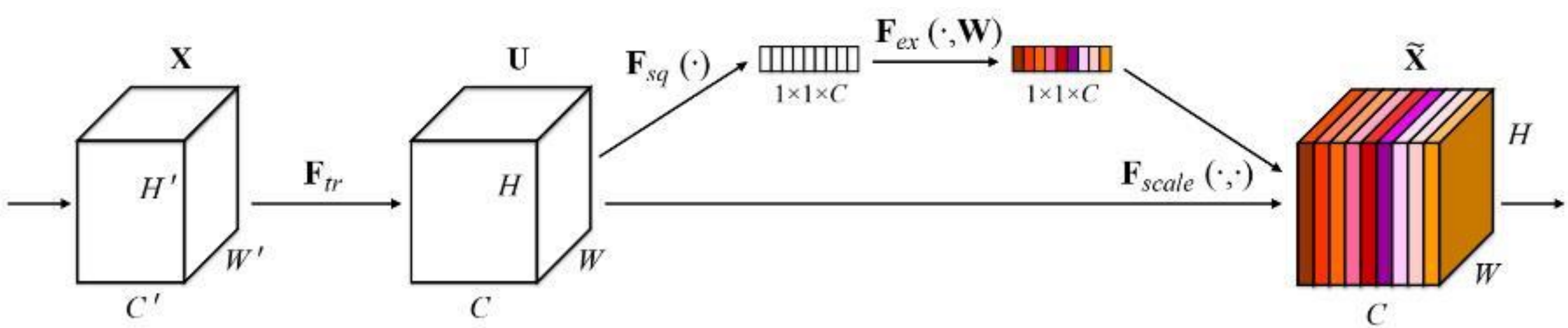
- 用神经元/通道的**平均失活率（mean APoZ）**剪枝
- 本文评估的LeNet和VGG-16，**没有BN操作**
- **被动剪枝**，而没有进行主动稀疏优化

Pay no Attention → Zero Activations

- APoZ完全体：APoZ=100%；
- 适应当前网络层；
- 主动稀疏化；



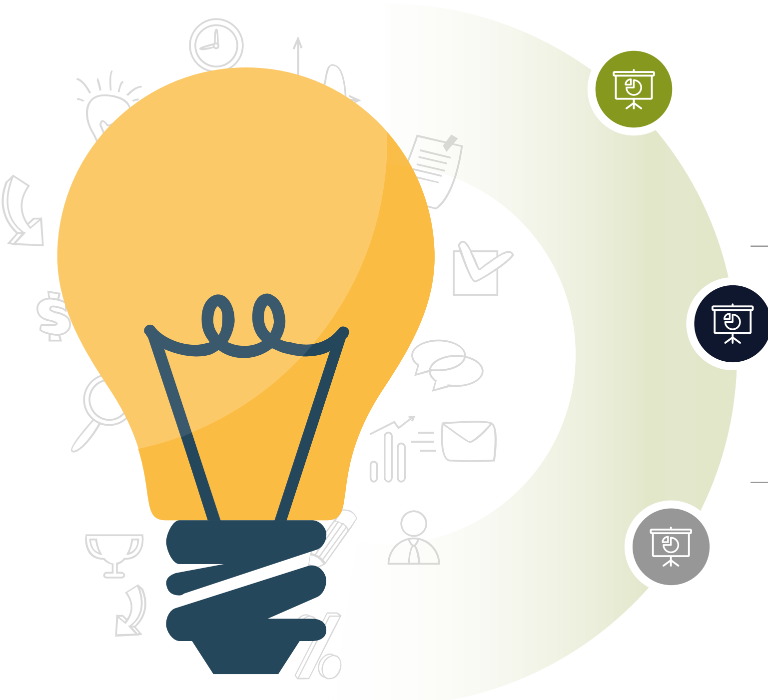
用APoZ剪filter



Squeeze-and-Excitation Networks

AutoPruner[1]

能否在fine-tuning阶段，引导weak filter及对其进行裁剪选择？ （pay no attention）



End-to-end

单模型内End-to-end训练，用fine-tuning阶段来引导剪枝

自适应压缩率

可以给定一个目标压缩率，然后会根据目标损失权衡，对多层进行训练得到一个最终的自适应压缩率

良好的泛化能力

多个数据集SOTA

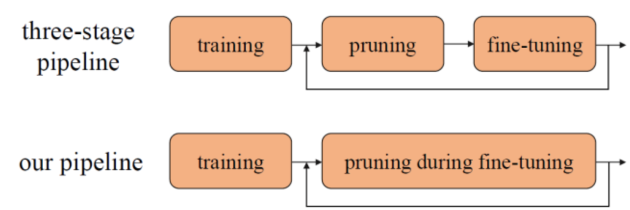


Figure 1. Overview of network pruning pipeline. The first row is a typical three-stage pruning pipeline, which regards pruning and fine-tuning as two independent processing steps. In the proposed AutoPruner method, we integrate filter selection into model fine-tuning. During fine-tuning, our method will gradually erase unimportant filters in an automatic manner.

[1] AutoPruner: An End-to-End Trainable Filter Pruning Method for Efficient Deep Model Inference, 2019, Pre-print in Arxiv

AutoPruner: pooling, coding, binarization

- Batch-wise average pooling: $X' = \frac{1}{N}\sum_{i=1}^{N} X_{i,:,:,:}$, N为batch size；
- Max-pooling with 2x2 filter size and stride 2 （省显存，但global average pooling又harmful，所以实验了一个还可以接受的pooling）

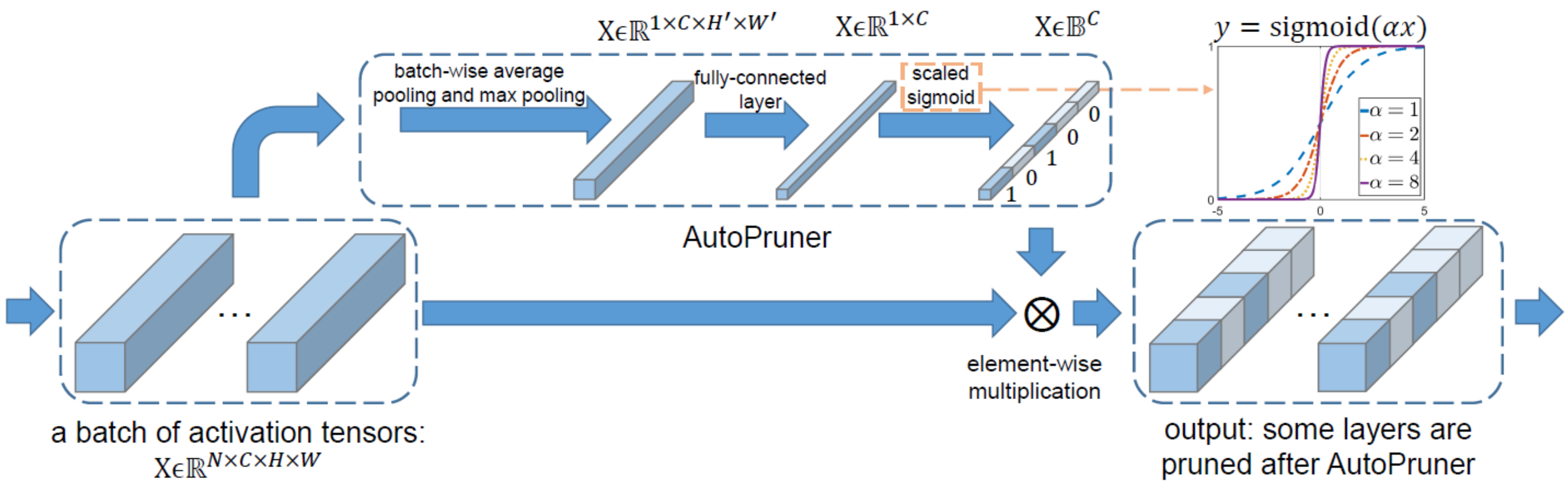


Figure 2. Framework of the proposed AutoPruner layer. Given a mini-batch of activation tensors, we use a new batch-wise average pooling and a standard max pooling to generate a single tensor. This tensor is projected into a C-dimensional vector via a fully-connected layer, where C is the number of channels. Finally, a novel scaled sigmoid function is used to obtain an approximate binary output. By gradually increase the value of α in scaled sigmoid function, the output of AutoPruner will gradually become a C-dimensional binary code. After training, all the filters and channels corresponding to the zeros index values will be pruned away to obtain a smaller and faster network. The new added AutoPruner layer will be removed too.

AutoPruner: pooling, coding, binarization

- Coding stage: 全连接层 $W \in \mathbb{R}^{C\times(CH'W')}$

$$X_{code} = f(WX+b), X \in \mathbb{R}^{CH'W'}$$

初始化权重不能太小，设置为：10x的标准差形式的MSRA $10 \times \sqrt{\frac{2}{n}}$。

- Binarization stage: 增加一个scaled sigmoid函数

$$y = sigmoid(\alpha x)$$

逐渐增大α的值，使得scaled函数趋向二值化。

$$\alpha = \alpha_{start} + \frac{\alpha_{stop} - \alpha_{start}}{TotalEpoches} * CurrentEpoch$$

α_{start}和α_{stop}的选取需要实验校验：fine-tuning前试一下α_{stop}能够实现二值化，α_{start}则足够的soft。反正设置比较heuristically

- 剪去binary code为0的channel，保留binary code为1的channel。

AutoPruner: Sparsity Control and Loss Function

- 用v表示index code vector，最常用的就是用ℓ_1-norm作为稀疏正则化表示（$\|v\|_1$），故压缩比例为$r \in [0,1]$的损失函数可以写为：

$$\min \mathcal{L}_{classification} + \lambda \left\| \frac{\|v\|_1}{C} - r \right\|_2^2$$

- 自适应调整$\lambda = 100 \times |r_b - r|$，$r_b$为当前实际压缩率。

Experiments

- CUB200-2011和ImageNet ILSVRC-12

Table 1. Compressing VGG16 on CUB200-2011 dataset using different algorithms and compression rates. For AutoPruner, we run it 3 times and report the mean±std values (%).

Method	compression rate $r = 0.5$			compression rate $r = 0.2$		
	top-1 (%)	top-5 (%)	#FLOPs	top-1 (%)	top-5 (%)	#FLOPs
fine-tuned VGG16	76.68	94.06	30.93B	76.68	94.06	30.93B
random selection	70.25	91.16	9.63B	57.28	83.52	2.62B
ThiNet [22] (Our implementation)	73.00	92.27	9.63B	63.12	87.54	2.62B
AutoPruner (Ours)	**73.45±0.26**	**92.56± 0.23**	9.63B	**65.06±0.32**	**87.93±0.34**	2.62B

Table 4. Comparison results among several state-of-the-art filter level pruning methods on ImageNet. All the accuracies are tested on validation set using the single view central patch crop. All the FLOPs numbers are calculated by Eq. 6 for a fair comparison.

Method	Top-1 Acc.	Top-5 Acc.	#FLOPs	speed up [1]
Original VGG16 model[2]	71.59%	90.38%	30.94B	1.00×
AutoPruner	**69.20%**	**88.89%**	8.17B	3.79×
SSS [12]	68.53%	88.20%	7.67B	4.03×
RNP (3×) [18]	-	87.58%	-	3.00×
RNP (4×) [18]	-	86.67%	-	4.00×
Channel Pruning (5×) [11][3]	67.80%	88.10%	7.03B	4.40×
Taylor expansion-1 [23]	-	84.50%	8.02B	3.86×
Taylor expansion-2 [23]	-	87.00%	11.54B	2.68×
Filter Pruning (impl. by [11]) [17]	-	75.30%	7.03B	4.40×
Original ResNet-50 model[2]	76.15%	92.87%	7.72B	1.00×
AutoPruner ($r = 0.3$)	**73.05%**	**91.25%**	2.64B	2.92×
ThiNet-30 [22]	68.42%	88.30%	2.20B	3.51×
AutoPruner ($r = 0.5$)	**74.76%**	**92.15%**	3.76B	2.05×
AutoPruner with block pruning ($r = 0.5$)	73.84%	91.75%	4.30B	1.80×
Channel Pruning (2×) [11][4]	72.30%	90.80%	5.22B	1.48×
SSS (ResNet-26) [12]	71.82%	90.79%	4.00B	1.93×
ThiNet-50 [22]	71.01%	90.02%	3.41B	2.27×

Discussion and Conclusion

- 增加了3个需要调的超参：λ, α_{start}，α_{stop}；
- Binarization的一致性问题：
  - 论文中虽然说 batch-wise average pooling 和 α-二值化 可以较好保证 scaled sigmoid 对不同图片输出二值结果的一致性，但没给出任何 theoretical 或 experimental proof
  - 单单取平均这个操作，就已经忽略了方差这个操作了（但是BN之后，可能方差更能代表一层的信息量）
  - 更别说后面又进行了full forward操作，这个影响就更乱了
  - 并且α-二值化使得α增大后对波动敏感，就更不鲁棒，更不好说这个一致性了

- 想法是好，但感觉说服力不够。

/02

How

Magnitude: Prune small Variance

Scaling Factors and Sparsity-induced Penalty[1]

- 对BN层的scaling factor进行稀疏诱导，然后剪枝small factor所在的通道：

$$L = \sum_{x,y} l(f(x,W),y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma)\,, g(s) = |s|$$

- BN层：

$$\hat{z} = \frac{z_{in} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \qquad z_{out} = \gamma \hat{z} + \beta$$

- 其中，γ, β是对输入进行仿射变换的可训练参数，γ负责scale标准激活输出。

- 对于residual block，使用channel selection layer做一个mask。

[1] Learning Efficient Convolutional Networks through Network Slimming, 2017, ICCV

Experiments

- 剪枝比例与测试误差，稀疏优化后γ的分布

(b) Test Errors on CIFAR-100

Model	Test error (%)	Parameters	Pruned	FLOPs	Pruned
VGGNet (Baseline)	26.74	20.08M	-	7.97×10^8	-
VGGNet (50% Pruned)	**26.52**	5.00M	75.1%	5.01×10^8	37.1%
DenseNet-40 (Baseline)	25.36	1.06M	-	5.33×10^8	-
DenseNet-40 (40% Pruned)	**25.28**	0.66M	37.5%	3.71×10^8	30.3%
DenseNet-40 (60% Pruned)	25.72	0.46M	54.6%	2.81×10^8	47.1%
ResNet-164 (Baseline)	23.37	1.73M	-	5.00×10^8	-
ResNet-164 (40% Pruned)	**22.87**	1.46M	15.5%	3.33×10^8	33.3%
ResNet-164 (60% Pruned)	23.91	1.21M	29.7%	2.47×10^8	50.6%

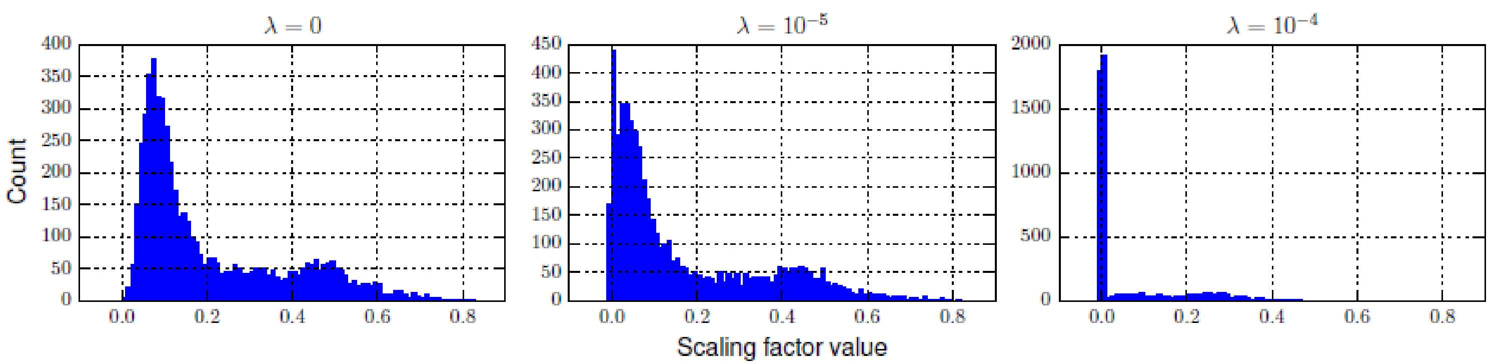


Figure 4: Distributions of scaling factors in a trained VGGNet under various degree of sparsity regularization (controlled by the parameter λ). With the increase of λ, scaling factors become sparser.

Discussion and Conclusion

- 简单易实现
- 用BN层的γ来衡量channel，直观

- 几乎**不可能将$\boldsymbol{\gamma}$惩罚至0**，所以还是依赖于阈值设置、剪枝比例设置等等
- 用γ来衡量channel本来就是比较heuristic的事情，而**不同层的$\boldsymbol{\gamma}$**放一起比较可能不太合适（[1]说合适，同时还说BN避免了跨层后的重参数化影响，所以$\boldsymbol{\gamma}$的scale效果及优化是层间独立的，也就是说，smaller norm less informative assumption是可行的）

[1] Rethinking the Smaller-Norm-Less-Informative Assumption in Channel Pruning of Convolution Layers, 2018, ICLR

Sparse Structure Selection[1]

- 将γ提取出来，作为独立一层，添加到可以剪枝的地方：block, group, neuron

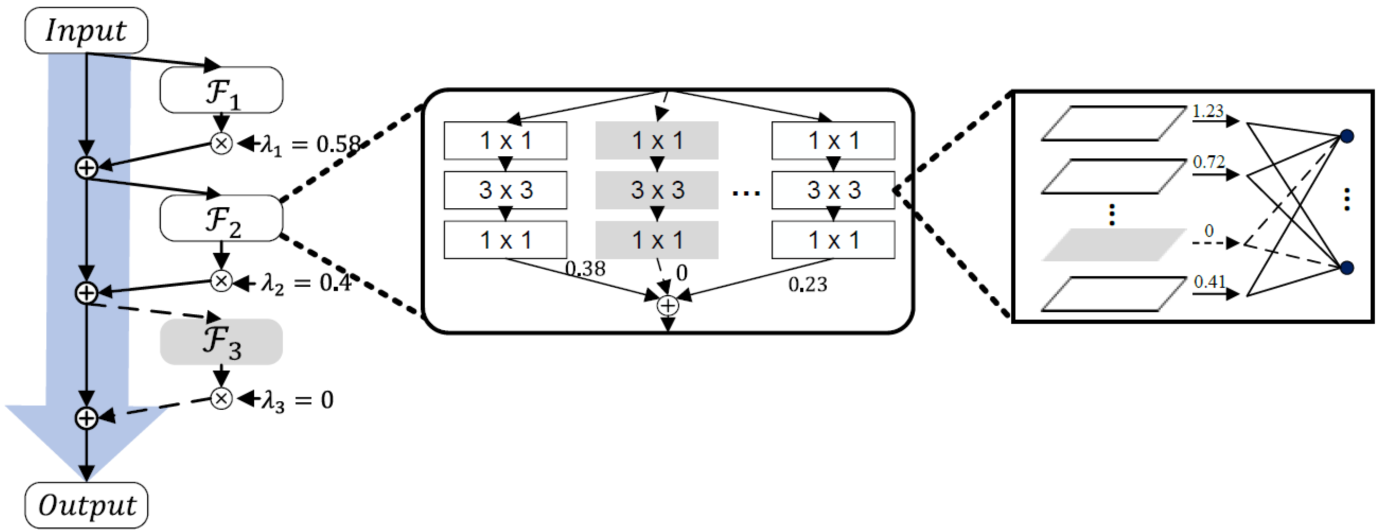


Fig. 1: The network architecture of our method. $\mathcal{F}$ represents a residual function. Gray block, group and neuron mean they are inactive and can be pruned since their corresponding scaling factors are 0.

[1] Data-Driven Sparse Structure Selection for Deep Neural Networks, 2018, ECCV

Optimization

- APG (accelerated proximal gradient): proximal gradient + momentum

MXNet implementation of APG

```
import mxnet as mx
def apg_updater(weight, lr, grad, mom, gamma):
    z = weight - lr * grad
    z = soft_thresholding(z, lr * gamma)
    mom[:] = z - weight + 0.9 * mom
    weight[:] = z + 0.9 * mom
def soft_thresholding(x, gamma):
    y = mx.nd.maximum(0, mx.nd.abs(x) - gamma)
    return mx.nd.sign(x) * y
```

Experiments

- VGG & ResNet & ResNeXt & PeeleNet

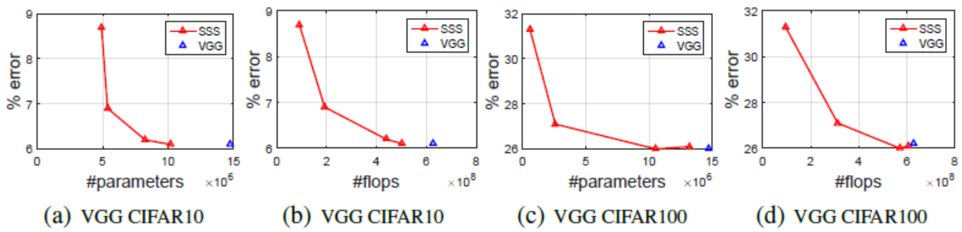


Fig. 2: Error vs. number of parameters and FLOPs after SSS training for VGG on CIFAR-10 and CIFAR-100 datasets.

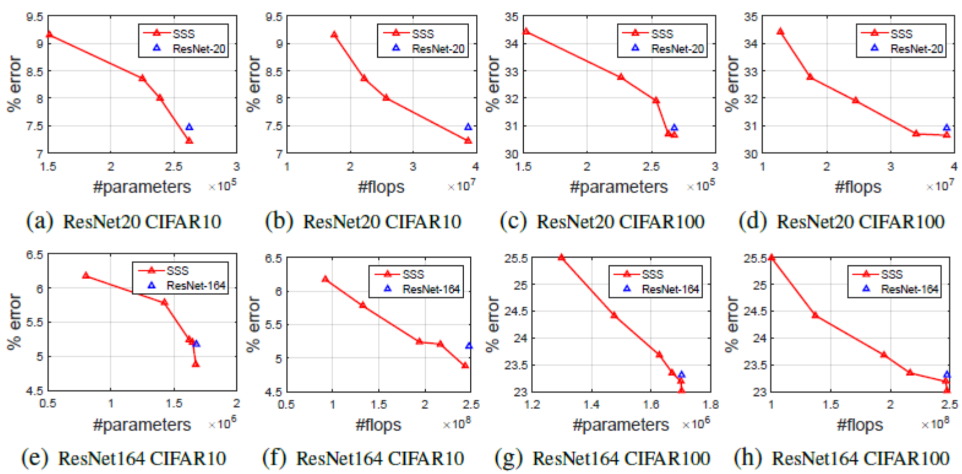


Fig. 3: Error vs. number of parameters and FLOPs after SSS training for ResNet-20 and ResNet-164 on CIFAR-10 and CIFAR-100 datasets.

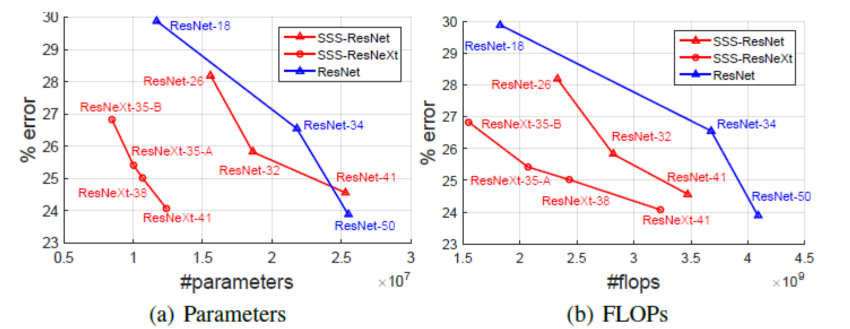


Fig. 5: Top-1 error vs. number of parameters and FLOPs for our SSS models and original ResNets on ImageNet validation set.

Conclusions

- 增加一层，结构灵活
- 算法易于实现

/02

How

Connection Sensitivity: gradient reflects the impact of connection

Connection Sensitivity

- $\mathbf{\Delta}\mathcal{L}$ with respect to connection strength $\boldsymbol{c} \in \{0,1\}^m$

[1] SNIP：single shot network pruning based on connection sensitivity，2019, ICLR

/03

Rethink

Network pruning as architecture search?

一直以来的剪枝定律

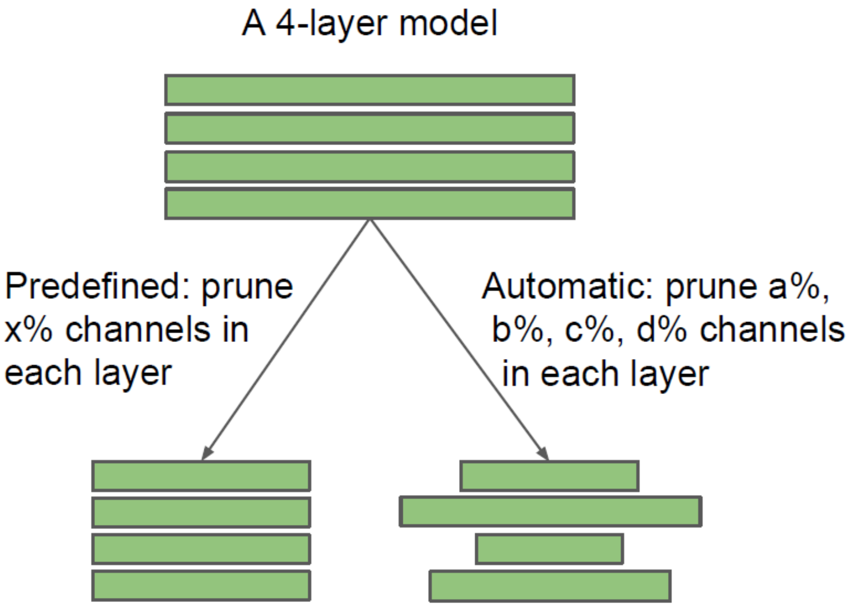
- 网络剪枝过程：
  - 设计一个过参数化的大网络模型；
  - 根据一定的准则，剪枝已训练模型；
  - 微调剪枝后的模型。

- Two common beliefs:
  - 一个过参数化的大模型可以为提供一个高performance的冗余模型，该精度可以为剪枝模型打包票；
  - 剪枝后的模型和权重值，对后续regain剪枝损失掉的精度很重要。

实验检验准则[1]

- 大模型是否必要?
- 剪枝后的权重值对剪枝后的模型精度是否至关重要?

- 对6中剪枝方式进行实验:
  - L_1-norm based Filter Pruning
  - ThiNet
  - Regression based Feature Reconstruction
  - Network Slimming
  - Sparse Structure Selection
  - Unstructured magnitude-based pruning



[1] Rethinking the value of network pruning, 2019, ICLR

Experiments

- ThiNet & Regression

Dataset	Unpruned	Strategy	Pruned Model		
ImageNet	VGG-16		VGG-Conv	VGG-GAP	VGG-Tiny
	71.03	Fine-tuned	−1.23	−3.67	−11.61
	71.51	Scratch-E	−2.75	−4.66	−14.36
		Scratch-B	**+0.21**	**−2.85**	**−11.58**
	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	**−4.56**	**−2.23**	**−1.01**

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

Experiments

- Network Slimming & Sparse Structure Selection

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

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

Network Pruning as Architecture Search

- 剪枝有效性

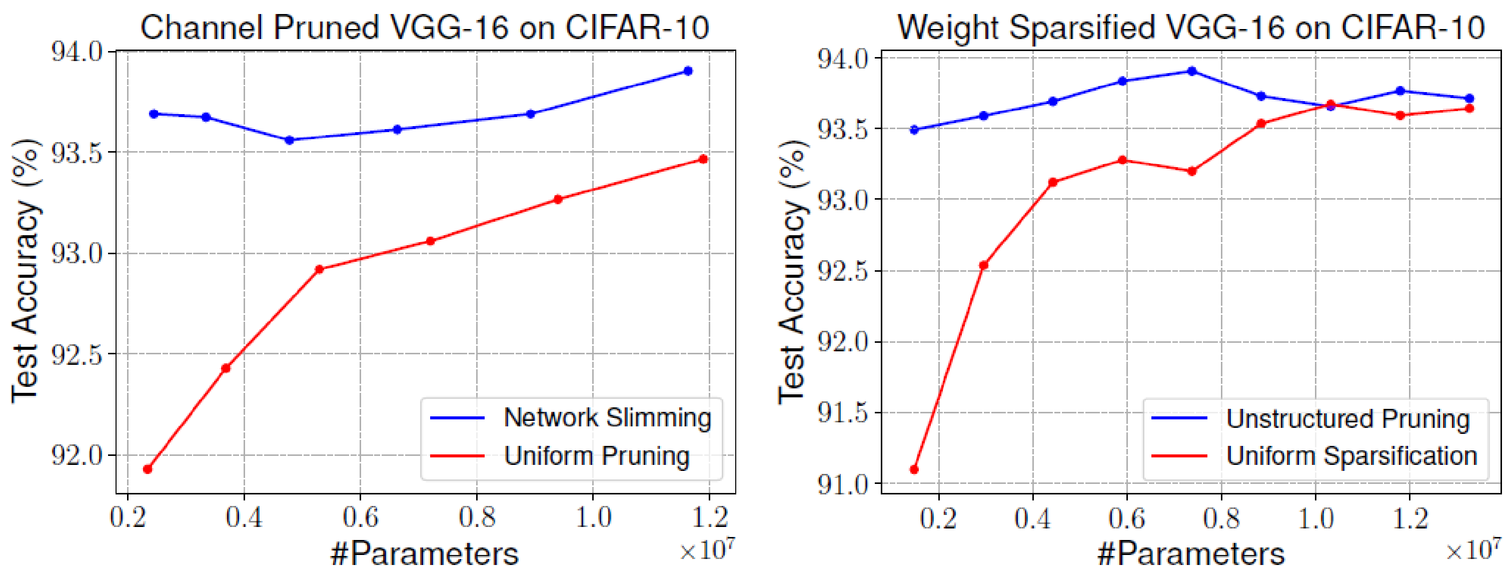


Figure 3: Pruned architectures obtained by different approaches, all *trained from scratch*, averaged over 5 runs. Architectures obtained by automatic pruning methods (*Left:* Network Slimming (Liu et al., 2017), *Right:* Unstructured pruning (Han et al., 2015)) have better parameter efficiency than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

Network Pruning as Architecture Search

- 剪枝一致性：搜索到一致的结构？自动剪枝的结构搜索意义？

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 7: Network architectures obtained by pruning 60% 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.

Network Pruning as Architecture Search

- 也存在“剪枝算法” 和“均匀剪枝”效果无差别的情况（大多出现在现代网络结构）
- 剪枝后的结构，不同层也基本呈现**均匀稀疏**的现象

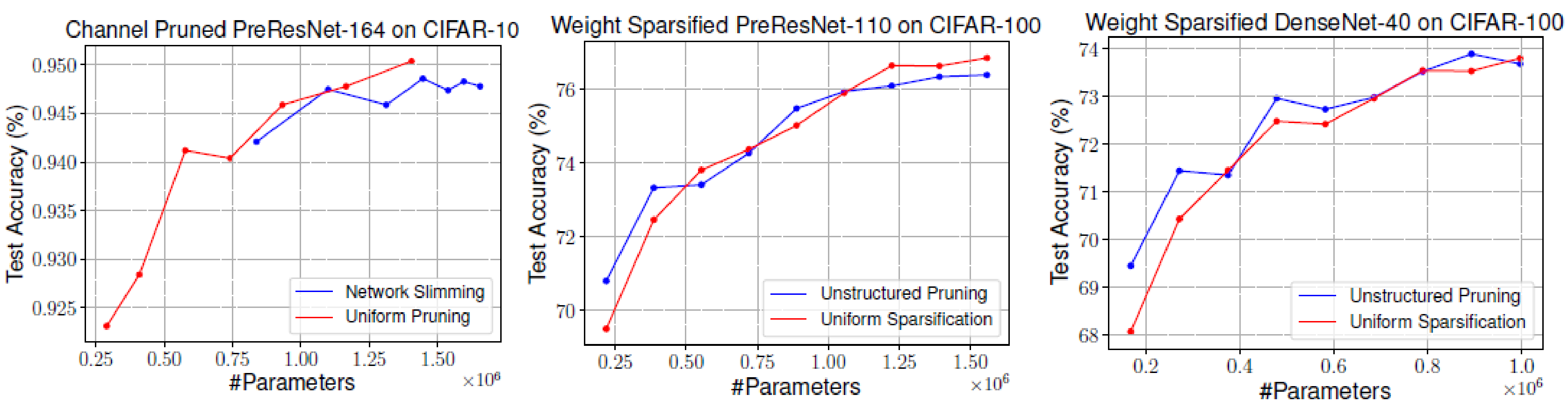


Figure 5: Pruned architectures obtained by different approaches, *all trained from scratch*, averaged over 5 runs. *Left:* Results for PreResNet-164 pruned on CIFAR-10 by Network Slimming (Liu et al., 2017). *Middle* and *Right*: Results for PreResNet-110 and DenseNet-40 pruned on CIFAR-100 by unstructured pruning (Han et al., 2015).

Network Pruning as Architecture Search

- 若剪枝是一种结构搜索，那剪枝后的结构可以指导结构设计么？可以的

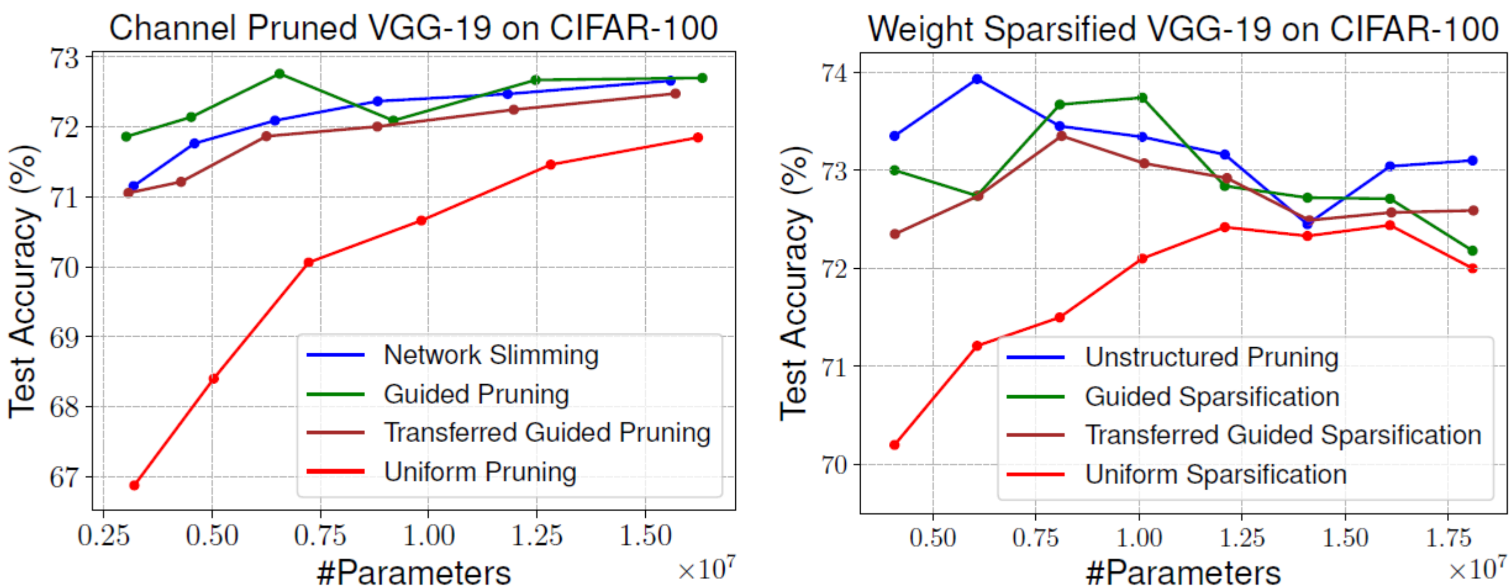


Figure 6: Pruned architectures obtained by different approaches, *all trained from scratch*, averaged over 5 runs. "Guided Pruning/Sparsification" means using the average sparsity patterns in each layer stage to design the network; "Transferred Guided Pruning/Sparsification" means using the sparsity 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.

Discussion and Conclusion

- 大模型不是并要的，除非：已有预训练大模型，需要根据情况获取不同大小的模型；
- 剪枝后的权重不是必要的：剪枝更像一种**结构搜索**，得到的是有效的结构，与权重无关；
- 将剪枝后的网络结构作为指导，进行结构调整，依然可以获得收益；
- 对**现代网络结构，目前的剪枝算法并不比均匀剪枝强**；（待考证，需要更多实验）
- 将**均匀剪枝作为baseline**，验证剪枝算法是必要的！

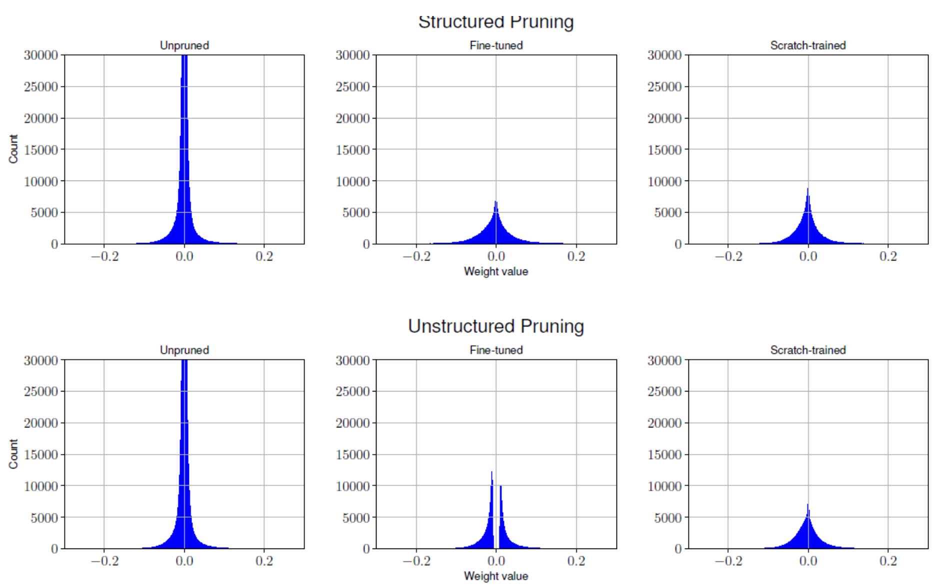


Figure 8: Weight distribution of convolutional layers for different pruning methods. We use VGG-16 and CIFAR-10 for this visualization. We compare the weight distribution of unpruned models, fine-tuned models and scratch-trained models. *Top*: Results for Network Slimming (Liu et al., 2017). *Bottom*: Results for unstructured pruning (Han et al., 2015).

/04

个人总结

如果我做实验，我会…

Sparse Structure Selection

- 易于实现
- 目前剪枝算法中，算是比较被认同的。
- 大佬背书：Tusimple提出的，naiyan是信得过的大佬

- 易于修改：
  - SSS + α-sigmoid
  - SSS + gradient-based sensitivity analysis

- 粗略看了腾讯的Discrimination-aware Channel Pruning for Deep Neural Networks，说：
  - reconstruction以前只考虑误差而忽略了通道的判别能力，也就是channel重构的效益不一致也要考虑；
  - 加稀疏项然后进行优化的方法，计算量大并且难以收敛（所以在我看来，是比较认同SSS这类算法的）

THANKS

Structured Pruning

Zheng Yuwei
Zheng.yuwei@foxmail.com