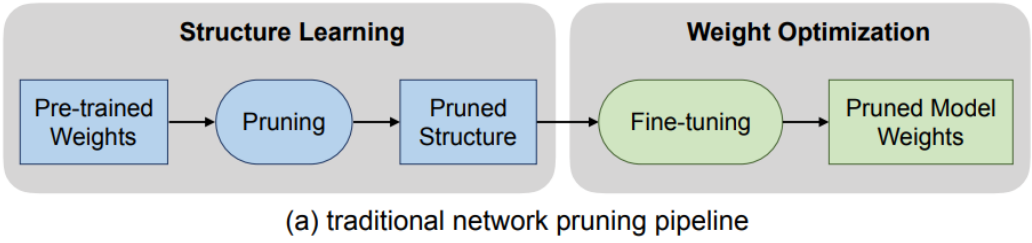
神经网络剪枝调研报告

刘伟健

神经网络剪枝概要

神经网络在图像分类、目标检测、语音翻译等很多方面都达到了 SOTA 水平。但这些深度网络参数大，在进行推理预测时有着较大的计算开销。模型剪枝的作用就是裁剪模型大小且最小化精度损失。

一般剪枝流程如图 1 所示，根据预训练的权重对模型结构进行剪枝后再通过微调训练恢复模型精度，得到参数更少精度相近的模型。随着深度模型剪枝领域研究的不断深入，更多的新颖的剪枝压缩流程被提出来并证明有效。但核心问题始终没变，即用什么作为剪枝的衡量标准和根据衡量如何进行剪枝。



(a) traditional network pruning pipeline

图 1：一般剪枝流程

Pruning Filters for Efficient ConvEnts (ICLR 2017)

提出一种 CNN 加速方法，从 CNN 中删减对输出精度影响小的卷积核。通过整体去除卷积核降低了计算量，相比于权重剪枝方法，不会产生稀疏矩阵，不需要稀疏卷积库的支持。可以减少 VGG-16 34%的推理时间和 ResNet-101 38%的推理时间,同时通过微调训练能恢复到原模型精度。

核心思想：用卷积核权重的L_1范数作为重要性评价指标，按照重要性排序，裁剪掉一层网络中重要性低的卷积核和相关特征图。

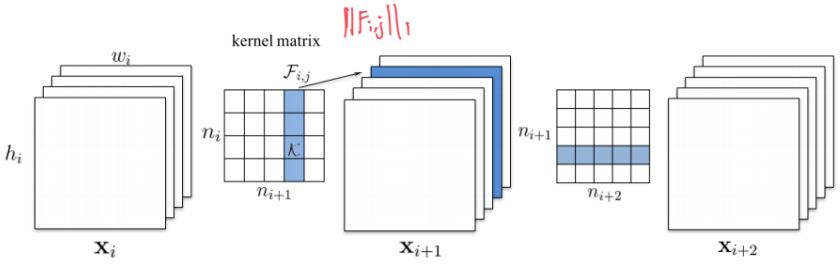
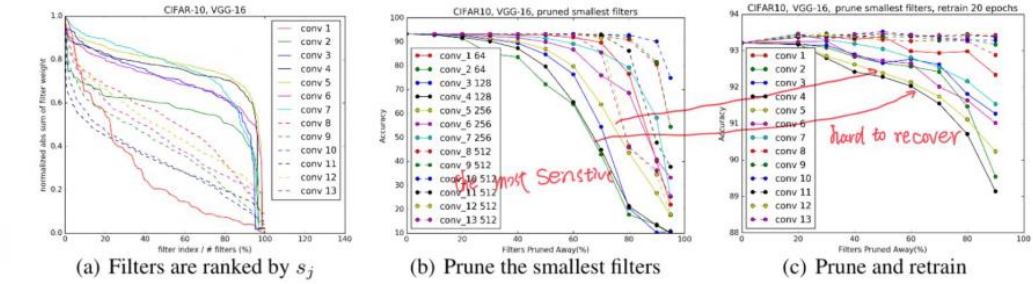


Figure 1: Pruning a filter results in removal of its corresponding feature map and related kernels in the next layer.

文章中对 VGG-16 网络中的每一层进行独立剪枝来通过 CIFAR-10 验证集上的正确率变化来观察各层网络的重要性。如下图（a）所示，计算每层网络中卷积核的权重绝对值和后进行标准化并排序。在下图（b）中，网络层对应的斜率低说明该层网络重要性较高。下

图（c）中，表明当重要性较高的网络层被删除后，模型很难通过再训练恢复到原来的精度。以 conv4 和 conv5 两层卷积网络为例，在（a）图中，他们的L_1范数是最大的两个，即重要程度较高。在（b）部分，conv4 和 conv5 对裁剪敏感，对应的曲线变化斜率低。在（c）部分也可以看见裁剪掉 conv4 和 conv5 的网络层后，模型精度经过再训练后也和原模型精度有着较大差异。综上所述，在 CIFAR-10 数据集上训练的 VGG-16 网络中，conv4 和 conv5 是重要程度较高的两层网络。



(a) Filters are ranked by s_j (b) Prune the smallest filters (c) Prune and retrain

Figure 2: (a) Sorting filters by absolute weights 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 the lowest absolute weights sum and their corresponding test accuracies on CIFAR-10. (c) Prune and retrain for each single layer of VGG-16 on CIFAR-10. Some layers are sensitive and it can be harder to recover accuracy after pruning them.

对于网络层的剪枝分为两种方式：

Independent pruning：是每一层网络独立进行剪枝，将下图中的黄色点考虑在权重计算之内。

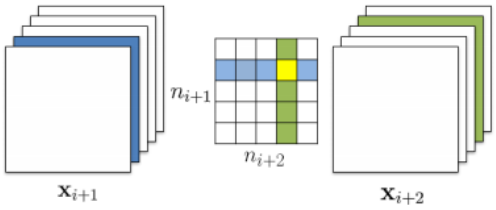


Figure 3: Pruning filters across consecutive layers. The independent pruning strategy calculates the filter sum (columns marked in green) without considering feature maps removed in previous layer (shown in blue), so the kernel weights marked in yellow are still included. The greedy pruning strategy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

Greedy pruning: 计算权重L_1范数时不计算前层已经修剪过的卷积核，即黄点不参与计算。

对于 VGG-16 这类较为简单的网络结构，可以进行任意的剪枝，但是在 ResNet 网络中由于残差块存在导致的维度限制，使得我们的剪枝需要考虑到一些限制条件。

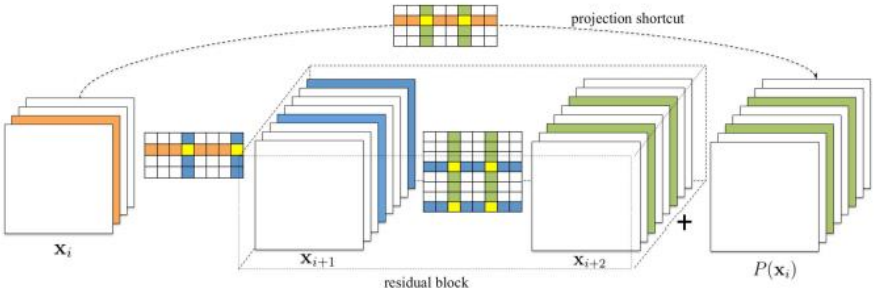


Figure 4: Pruning residual blocks with the projection shortcut. The filters to be pruned for the second layer of the residual block (marked as green) are determined by the pruning result of the shortcut projection. The first layer of the residual block can be pruned without restrictions.

论文也提供了两种重训练的方法：

Prune once and retrain:一次修剪多次卷积核并重新训练，并恢复到原始精度。

Prune and retrain iteratively:逐层修剪过滤器然后重新迭代训练，在下次剪枝之前对模型进行再训练，使权重适应剪枝过程。

论文表示，第一种方法可以用于删除网络的重要部分，并且可以通过短于原始训练时间的重新训练来恢复精度。但当敏感层的一些卷积核被剪掉或大部分网络被剪掉时，可能无法恢复原始精度。这时第二种方法可能会产生更好的结果，但需要更多的迭代次数。

Learning Efficient Convolutional Networks through Network Slimming (ICCV 2017)

论文提出一种通道级别的模型剪枝方法，在训练过程中，剪枝掉不重要的网络通道。该方法可以直接用于现代网络结构，且不需要特殊的软硬件加速。该方法使用一个宽而大的网络模型作为输入，在训练过过程中，根据不重要的通道会被自动识别并进行剪枝。

核心思想：通过批归一化层的缩放因子作为重要性评价指标，对缩放因子较小通道进行剪枝。

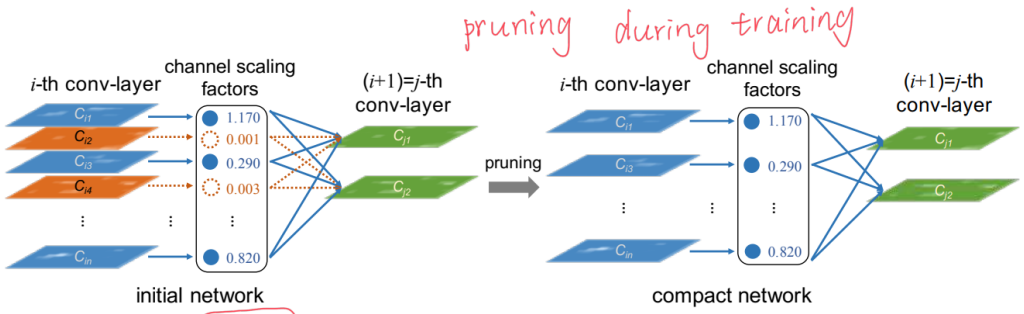


Figure 1: We associate a scaling factor (reused from a batch normalization layer) with each channel in convolutional layers. Sparsity regularization is imposed on these scaling factors during training to automatically identify unimportant channels. The channels with small scaling factor values (in orange 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.

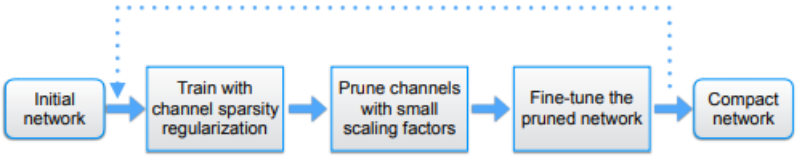


Figure 2: Flow-chart of network slimming procedure. The dotted-line is for the multi-pass/iterative scheme.

Batch Normalization 的输入为一次神经网络的权值，计算方式如下：

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

其中 γ 为剪枝参考的比较因子。可以通过论文中的损失函数进行迭代学习。

$$L = \sum_{(x,y)} l(f(x,W),y) + \lambda \sum_{\gamma\in\Gamma} g(\gamma) \tag{1}$$

λ 为超参数，可以用来控制 γ 的稀疏分布。当 λ=0 时，γ 的分布如下图左一所示，分布相对平坦，当 $\lambda=10^{-5}$，γ 的部分出现稀疏化趋势，当 $\lambda=10^{-4}$时，大部分 γ 的分布接近于 0。模型剪枝后可以通过微调训练恢复模型精度，但当剪枝比例过大时，模型测试误精度可能无法恢复。论文采用 DenseNet-40 在 CIFAR-10 上采用 $\lambda=10^{-5}$进行训练，实验发现，当剪枝阈值超过 80%时，微调模型的测试精度低于基线精度，无法恢复。

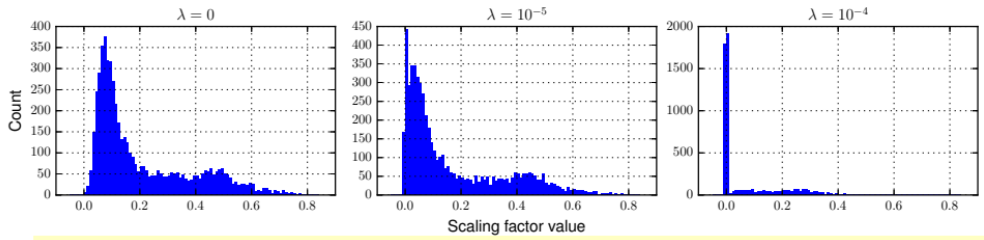


Figure 4: Distributions of scaling factors in a trained VGGNet under various degree of sparsity regularization (controlled by the parameter λ). With the increase of λ, scaling factors become sparser.

目前模型剪枝存在的问题：

Low-rank Decomposition(低秩分解方法):采用 SVD 等方法用低秩矩阵近似权重矩阵，在全连接层上效果比较好。

Weight Quantization(权重量化):HashNet 采用分组、共享权值的方法来压缩所需保存的参数数量，但在推理阶段没有效率提升。或将真实的权重值量化为二值/三值化权重，可以极大的压缩了模型大小，同时由于位操作库的使用速度也很快。然而，激进的低位近似方法通常会伴随着相当的精度损失。

Weight Pruning/ Sparsifying(权重剪枝/稀疏化):剪枝掉网络中较小的权值，会得到网络权值大多数为 0 的稀疏矩阵。但需要特殊的稀疏矩阵运算库或者硬件支持来实现加速

Structured Pruning/Sparsifying(结构剪枝/稀疏化):本文所属方法。

Neural Architecture Learning(网络结构学习): 有论文提出了使用强化学习自动学习神经网络结构，但这些方法的搜索空间非常大，效率较低。

Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration (CVPR 2019)

以前的工作利用“小范数重要程度低”的标准进行卷积核的剪枝，论文这种基于范数剪枝方法的有效性通常取决于两个不总是满足的条件

1) 卷积核的范数偏差要大
2) 卷积核的最小范数应该接近于 0

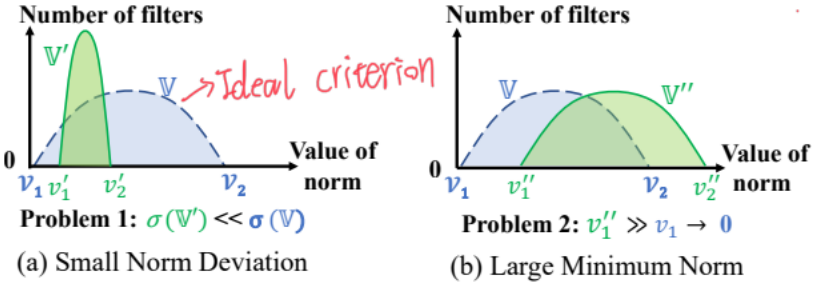
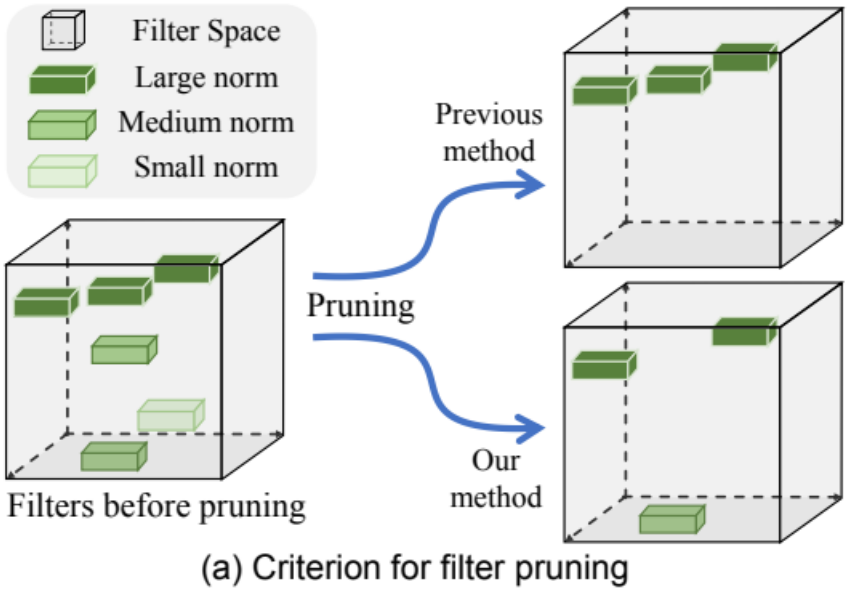


Figure 2. Ideal and Reality of the norm-based criterion: (a) Small Norm Deviation and (b) Large Minimum Norm. The blue dashed curve indicates the ideal norm distribution, and the green solid curve denotes the norm distribution might occur in real cases.

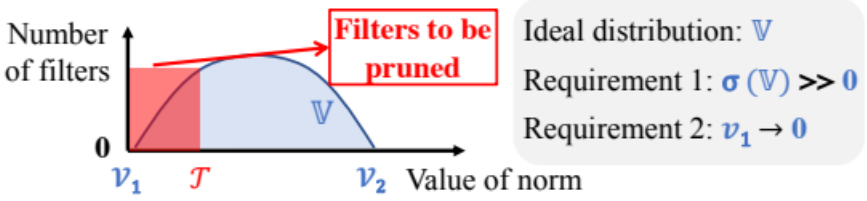
但在在实际情况下，两个前提条件通常较难满足，上图（a）反映了当范数分布偏差较小时，范数数值分布集中在较小的区间范围内，导致较难选择合适的阈值用于剪枝；如上图（b）所示，当最小范数的最小数值没有接近于 0 时，范数低于阈值的卷积核仍对网络性能有一定程度的贡献，剪枝掉这些卷积核将对网络性能带来较为明显的负面影响。

几何中位数是对于欧几里得空间的点的中心的一个估计。论文认为滤波器也是欧氏空间中的点，于是我们可以根据计算 GM 来得到这些滤波器的“中心”，也就是他们的共同性质。如果某个滤波器接近于这个 GM，可以认为这个滤波器的信息跟其他滤波器重合，甚至是冗余的，于是我们可以去掉这个滤波器而不对网络产生大的影响。去掉它后，它的功能可以被其他滤波器代替中。因此提出 FPGM 评价指标，得到一种跟范数无关的滤波器评价方法消除了范数评价指标的局限性。

下图中绿色方框表示网络的过滤器，其中更深的颜色表示更大的范数的卷积核。如下图（a）所示，基于较小范数的过滤器重要程度低的假设，只保留范数最大的几个卷积核。用本文提出的方法对网络中的冗余信息进行剪枝。可以保留不同范数规模的过滤器。下图(b)中，蓝色曲线表示网络的理想范数分布，v_1和v_2是范数分布的最小值和最大值。选择合适的阈值$\mathcal{T}$，对卷积核进行剪枝。



(a) Criterion for filter pruning



(b) Requirements for norm-based criterion

论文最后可视化了 ResNet-50 第一层的网络通道特征，红圈标注的特征图为裁剪掉的通道，作者给出带有红色标注的（7，23，27，46，56，58）特征图包含竹子和大熊猫头部和身体的轮廓特征，这些被裁剪掉的特征图可以被包含竹子轮廓的特征图（5，12，16，18，22 等）以及包含熊猫轮廓的特征图（0，4，33，34，47 等）替代。而不会损失大量的特征信息。

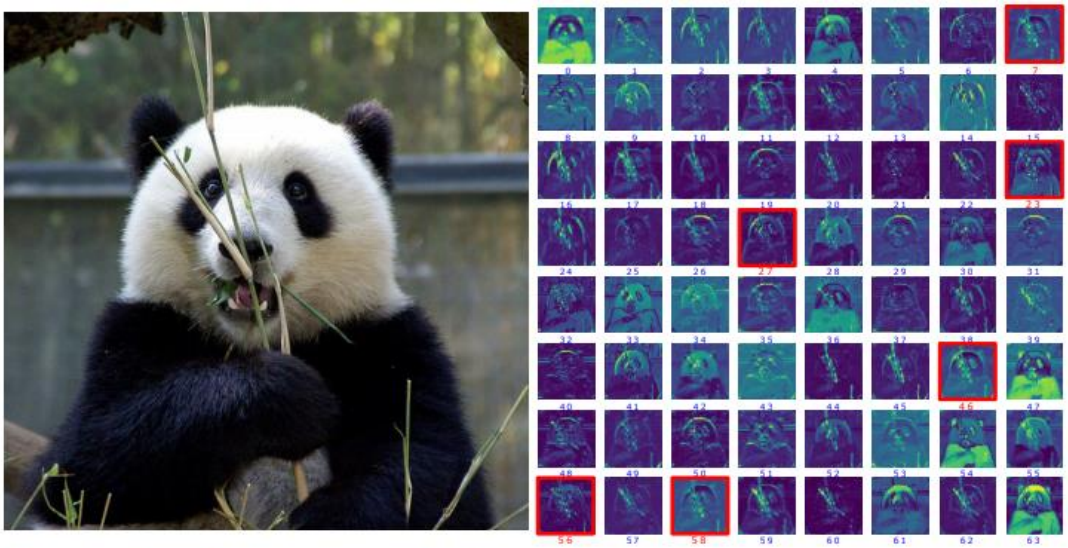


Figure 5. Input image (left) and visualization of feature maps (right) of ResNet-50-conv1. Feature maps with red bounding boxes are the channels to be pruned.

NetworkTrimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures (CoRR 2016)

文章指出，无论接受什么样的输入，一个大型网络中相当一部分神经元的输出大多为零。作者认为这些零激活的神经元是冗余的，可以在不影响网络精度的情况下移除。本文提出了一个新的指标应用于剪枝方法： Average Percentage of Zeros (APoZ)。首先使用确定的数据集对已有的神经网络进行测试，找出弱神经元，进行剪枝，生成新的模型，这些新的模型可以获得跟原先网络相同甚至更好的表现能力。

APOZ 的定义如下，表示经过 Relu 函数映射后零激活神经元的所占百分比。

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

其中c表示第c个通道，i表示第i个网络层。M表示特征图维度，N表示验证样本的数量。

作者对 VGG-16 网络的每一层都计算了 APoZ，结果如下表：

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

由于 VGG-16 的倒金字塔结构，网络中的大部分冗余发生在较高的卷积层和全两连接层。经过计算 VGG-16 中有 631 个神经元的 APoZ 值大于 90%。

论文中给出的具体做法分为三个步骤，首先对网络进行常规训练，然后在一个大型验证集上进行 APoZ 剪枝，剪枝完后进行利用之前的权值进行初始化训练。需要注意的是，作者在实验中发现，如果一次剪枝过多的神经元会导致性能的严重损失，且不能通过再训练恢复，因此需要采用迭代剪枝的方法逐步修剪邻近网络层。

作者在 LeNet 的剪枝实验结果如下表所示：

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

对 LeNet 进行了四次迭代剪枝，未经过压缩的原模型准确率为 99.31%，经过第一次 APoZ 剪枝后,模型精度降低到 98.75%，通过再训练恢复到 99.29%，经过四次迭代剪枝后，压缩比达到 3.85，准确率下降了 0.05%。

VGG -16 的剪枝实验结果如下表：

Table 4: Iterative Trimming Result on VGG-16 {CONV5-3, FC6}

Network (CONV5-3, FC6)	Compression Rate	Before Fine-tuning (%)		After Fine-tuning (%)	
		Top-1 Accuracy	Top-5 Accuracy	Top-1 Accuracy	Top-5 Accuracy
(512, 4096)	1.00	68.36	88.44	68.36	88.44
(488, 3477)	1.19	64.09	85.90	71.17	90.28
(451, 2937)	1.45	66.77	87.57	71.08	90.44
(430, 2479)	1.71	68.67	89.17	71.06	90.34
(420, 2121)	1.96	69.53	89.49	71.05	90.30
(400, 1787)	2.28	68.58	88.92	70.64	89.97
(390, 1513)	2.59	69.29	89.07	70.44	89.79

正如作者所说，在某些情况下，通过 APoZ 剪枝方法剪去冗余神经元不仅可以减少计算，还可以获得更好的精度。作者认为，这是由于原网络出现了过拟合的情况导致。

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks (ICLR 2019 best paper)

作者发现标准的剪枝技术，自然的揭示了初始化后能够有效训练的子网络。基于这些结果，本文提出了彩票假设：一个随机初始化的稠密神经网络包含一个子网络，该子网络经过初始化和隔离训练后，可以在相似的迭代次数内达到与原始网络相当的测试精度。并提出了一种算法来识别中奖彩票和一系列支持彩票假设的实验。

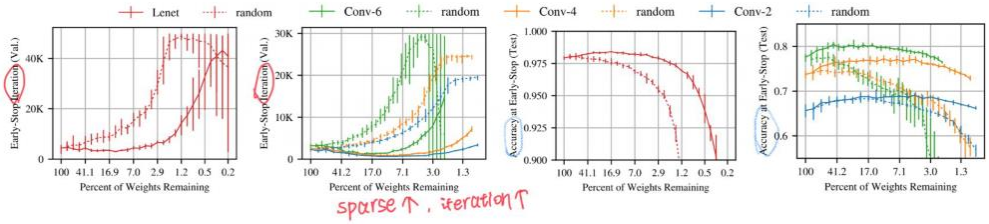


Figure 1: The iteration at which early-stopping would occur (left) and the test accuracy at that iteration (right) of the Lenet architecture for MNIST and the Conv-2, Conv-4, and Conv-6 architectures for CIFAR10 (see Figure 2) when trained starting at various sizes. Dashed lines are randomly sampled sparse networks (average of ten trials). Solid lines are winning tickets (average of five trials).

如上图所示，虚线表示随机剪枝得到的子网表现，实现表示作者采用彩票理论找到的中奖彩票子网表现。实验表明，中奖彩票找到的子网能够更快的训练，并达到跟原网络相似的精度。同时表明网络越稀疏学习越慢，并且最终的测试精度越低。

文章提出的识别中奖彩票的方法如下：

1.随机初始化神经网络

2.网络经过迭代训练后，得到权重参数θ

3.对权重参数θ剪枝后，生成一个 mask

3.剩余结构中用原始参数初始化后进行训练，以产生中奖彩票。

Identifying winning tickets. We identify a winning ticket by training a network and pruning its smallest-magnitude weights. The remaining, unpruned connections constitute the architecture of the winning ticket. Unique to our work, each unpruned connection's value is then reset to its initialization from original network *before* it was trained. This forms our central experiment:

1. Randomly initialize a neural network $f(x; \theta_0)$ (where $\theta_0 \sim \mathcal{D}_\theta$).
2. Train the network for j iterations, arriving at parameters θ_j.
3. Prune $p\%$ of the parameters in θ_j, creating a mask m.
4. Reset the remaining parameters to their values in θ_0, creating the winning ticket $f(x; m \odot \theta_0)$.