ThiNet：一种用于深度神经网络压缩的滤波器级修剪方法

摘要

提出了一个高效统一的框架——ThiNet

解决问题：在训练和推理阶段加速和压缩CNN模型，不同于之前的方法只能根据当前filter中kernel值来修剪filter

主要思路：filter级剪枝——丢弃不重要的filter，将filter剪枝视为一个优化问题，根据下一层的统计信息来修剪filter，而不是根据当前层；**不改变模型结构**，使得剪枝后的模型依然支持现有的深度学习库

表现：

1. 成为当时比较sota的模型
2. 使$VGG-16$在$ILSVRC-12$基准测试集上实现了$3.31\times$$FLOPs$减少和$16.63\times$压缩率；$top-5$准确率下降了$0.52\%$；也可进一步压缩为只有5.05MB的更小的模型，精度与AlexNet相当，但泛化能力更强的模型

   $FLOPs$：floating point operations——浮点运算数，即计算量，用来衡量模型的复杂度
3. 使更加紧凑的模型$ResNet-50$在相同的基准测试集上减少了一半的参数和计算量；$top-5$准确率下降了0.52%

1 介绍

贡献

- 提出了一个简单而有效的框架，即 ThiNet，以同时加速和压缩 CNN 模型。ThiNet 在许多任务上对现有方法的显着改进。
- 将过滤器修剪变为优化问题，使用从下一层计算的统计信息，而不是当前层，它将ThiNet与现有方法区分开来
- 在实验中，VGG-16 模型可以修剪为 5.05MB，在迁移学习方面表现出有希望的泛化能力。

2 相关工作

过去的一些剪枝方法存在的缺点与不足，结构化剪枝、非结构化剪枝

3 模型

3.1 ThiNet框架

- 评估每个神经元的重要性，删除那些不重要的神经元，并对整个网络进行微调
- 给定一个预训练的框架，根据预定义的压缩率逐层修剪，分为以下三个步骤
  - **通道选择**：根据下一层的统计信息来选择本层的filter，选择下一层的通道形成子集，以该子集作为输入近似等于输出，则其他通道被认为不重要可以被删除，下一层的通道由本层的某个过滤器产生，因此茨过滤器可以删除
  - **剪枝**：修剪后的网络具有相同的结构，但过滤器和通道更少，网络由宽变薄
  - **微调**：通过微调恢复网络的泛化能力。但是对于大型数据集和复杂模型来说，这将花费很长时间。出于节省时间的考虑，论文在修剪一层后微调一个或两个 epoch。为了获得准确的模型，当所有层都被修剪时，将执行更多额外的 epoch

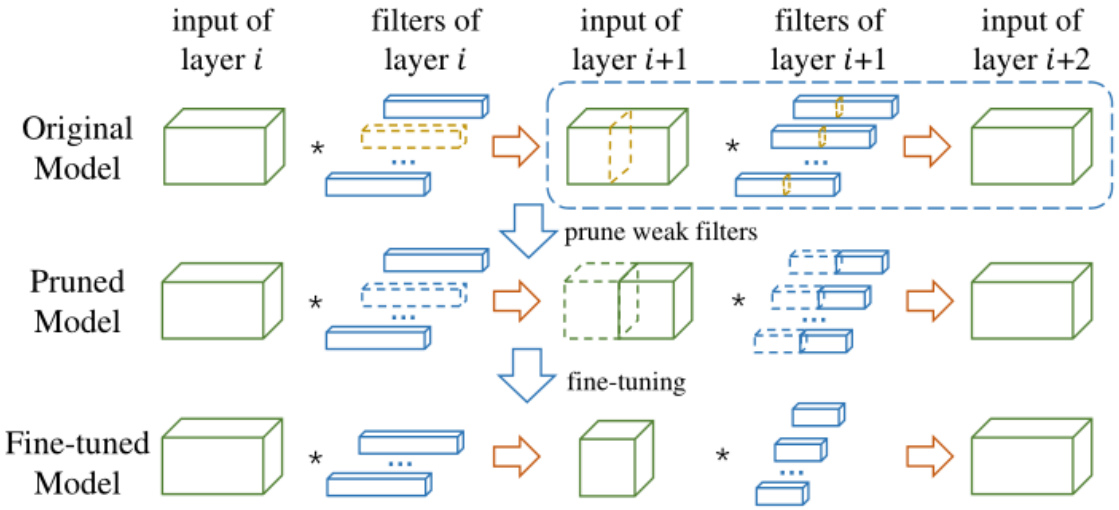

Figure 1. Illustration of ThiNet. First, we focus on the dotted box

3.2 数据驱动的选择方法

最小化重构误差——修建之后的下一层的输出与未修剪的相同层的输出的误差

- 收集训练实例

$$y = \sum_{c=1}^{C} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \widehat{\mathcal{W}}_{c,k_1,k_2} \times x_{c,k_1,k_2} + b.$$

$$\hat{x}_c = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \widehat{\mathcal{W}}_{c,k_1,k_2} \times x_{c,k_1,k_2},$$

$$\hat{y} = \sum_{c=1}^{C} \hat{x}_c,$$

In other words, if we can find a subset $S \subset \{1, 2, \ldots, C\}$ and the equality

$$\hat{y} = \sum_{c \in S} \hat{x}_c \tag{4}$$

- 贪心算法

Algorithm 1 A greedy algorithm for minimizing Eq. 6

Input: Training set $\{(\hat{\mathbf{x}}_i, \hat{y}_i)\}$, and compression rate r
Output: The subset of removed channels: T

$T \leftarrow \emptyset$; $I \leftarrow \{1, 2, \ldots, C\}$;
while $|T| < C \times (1 - r)$ **do**
  $min\_value \leftarrow +\infty$;
  for each item $i \in I$ **do**
    $tmpT \leftarrow T \cup \{i\}$;
    compute $value$ from Eq. 6 using $tmpT$;
    if $value < min\_value$ **then**
      $min\_value \leftarrow value$; $min\_i \leftarrow i$;
    end if
  end for
  move min_i from I into T;
end while

$$\arg\min_{S} \sum_{i=1}^{m} \left(\hat{y}_i - \sum_{j \in S} \hat{\mathbf{x}}_{i,j} \right)^2$$
$$\text{s.t.} \quad |S| = C \times r, \quad S \subset \{1, 2, \ldots, C\}.$$

$$\arg\min_{T} \sum_{i=1}^{m} \left(\sum_{j \in T} \hat{\mathbf{x}}_{i,j} \right)^2$$
$$\text{s.t.} \quad |T| = C \times (1 - r), \quad T \subset \{1, 2, \ldots, C\}.$$

- 最小化重构误差

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{i=1}^{m} (\hat{y}_i - \mathbf{w}^{\mathrm{T}} \hat{\mathbf{x}}_i^*)^2,$$

3.3 剪枝策略

- 对于VGG-16，前10层中存在超过90%的FLOPs，而全连接层有超过近 86.41% 的参数；修剪前 10 层加速考虑，将 FC 层替换为全局平均池化层
- 对于 ResNet，由于其特殊结构，存在一些限制。例如，同一组中每个块的通道数需要是一致的，才能完成求和运算，很难直接修剪每个残差块的最后一个卷积层。由于大多数参数位于前两层，因此主要修剪前两层参数

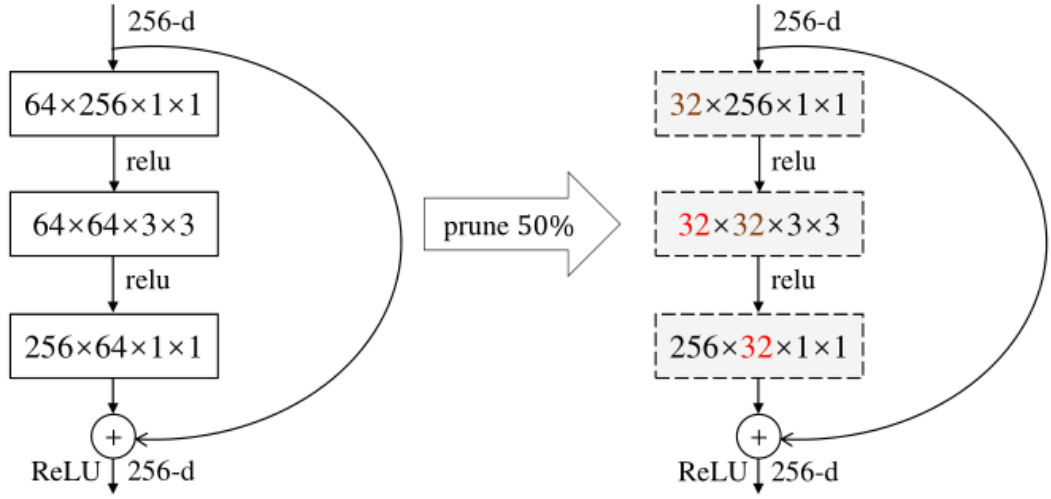

Figure 3. Illustration of the ResNet pruning strategy. For each

4 实验

首先，比较了不同filter选择方法；其次，介绍了两个被剪枝的网络$VGG-16$和$ResNet-50$在基准测试集$ILSCVR-12$上的模型表现；最后，展示了ThiNet在实用场景中的效果。

4.1 不同的filter选择方法

- 方法：
  - **Weight sum**：拥有较小的kernel权重的filter趋向于会产生更小的激活值，被认为是重要程度较低的filter，该方法计算每一个filter的权重和作为衡量filter的重要性分数；$s_i = \sum |W(i,:,:,:)|$
  - **APoZ(Average Percentage of Zeros)**：通过计算输出激活中每个通道的稀疏性作为重要性分数；$s_i = \frac{1}{|\Gamma(i,:,:,:)|} \sum \sum \Pi(\Gamma(i\ :,:,:) == 0)$
- 实验
  - 广泛使用的细粒度数据集——CUB-200

    该数据集包含11788张200类不同鸟类的照片，5994张训练集，5794张测试集
  - 用全局平均池化代替$VDD-16$的全连接层，并在新的数据集上进行微调，从微调模型开始，每层设置不同的压缩率进行剪枝；每个剪枝后跟一个epoch进行微调，最后一层执行12个epoch来提高准确性；
  - 使用不同的filter选择方法得到剪枝后的模型并将其应用到CUB-200进行图片分类，多次重复上述步骤，将实验结果取平均值

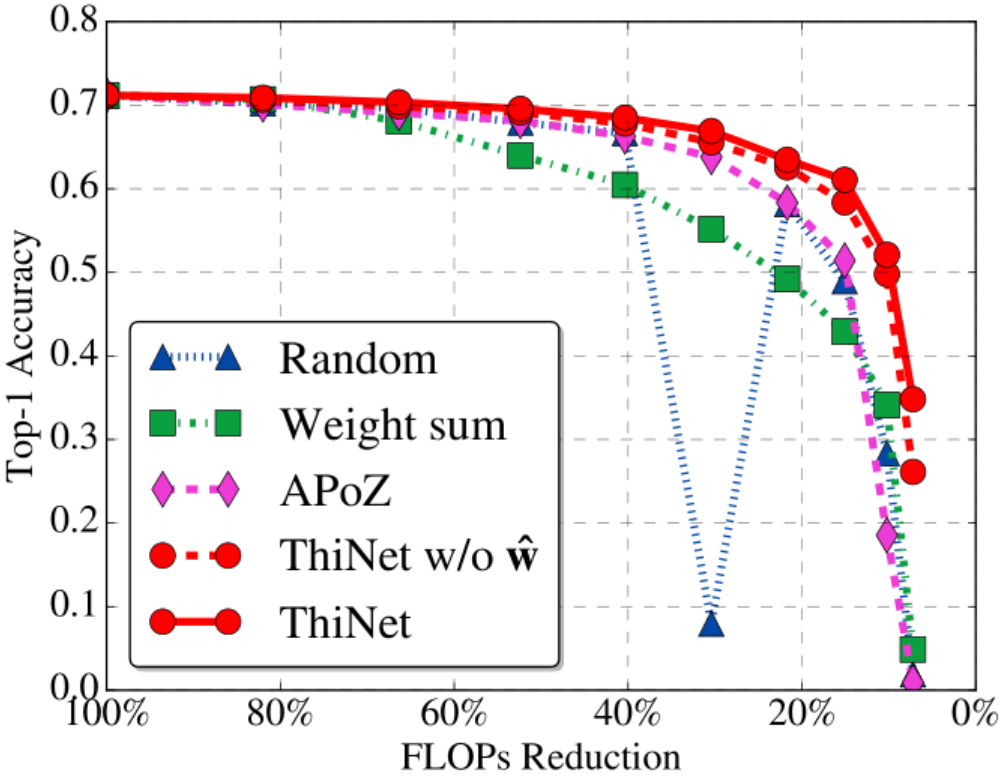

Figure 4. Performance comparison of different channel selection

- Random方法在某些情况下表现相当好，由于某些启发式的方法，但鲁棒性差，实际中不适用
- **权重和**表现效果最差，只考虑了权重和没有考虑分类的精确度，小权重也可能对损失函数产生较大影响；
- TiNet表现更好、鲁棒性更好，且最小二乘法有助于获得更好的权重初始化用于微调，尤其当压缩率较高时表现会更好

4.2 VGG-16 on ImageNet

评估ThiNet在大规模分类任务上的性能——VGG-16

- 数据集——ILSCVR-12：包含1000类别，总共100万张训练图片
  - 训练：随机从每个类别选取10张图片组成评估集，评估每个filter的重要程度，在输入一张图片时根据通道位置和空间位置不同随机采样10个实例——训练集：100000，来根据算法寻找最佳通道子集
  - 推理：最后在50k标准测试集上进行测试
- 实验设置
  - 整体微调：1 epoch——学习率10^{-3}，不同的层设置不同的学习率来阻止精确度下降过快
  - 修剪最后一层时，采用更多的epoch（12）来获得更精确的结果，学习率$10^{-3} - 10^{-5}$
  - 随机梯度下降、mini-batch size=128
    - ThiNet-Conv：只有前十个卷积层以0.5压缩率进行压缩
    - ThiNet-GAP：用全局平均池化代替所有全连接层，并用相同参数微调12个epoch
    - ThiNet-Tiny：更大的压缩率（0.25）修剪模型，得到更小的模型——准确率与AlexNet相同，ThiNet-Tiny的模型复杂度与紧凑网络SqueezeNet相同，但具有更高的准确率，结构也更加简单；SqueezeNet采用了一种特殊的结构，即Fire模块，参数高效，但依赖于人工网络结构设计。

M40 GPU with batch size 32.

Model	Top-1	Top-5	#Param.	#FLOPs[1]	f./b. (ms)
Original[2]	68.34%	88.44%	138.34M	30.94B	189.92/407.56
ThiNet-Conv	69.80%	89.53%	131.44M	9.58B	76.71/152.05
Train from scratch	67.00%	87.45%	131.44M	9.58B	76.71/152.05
ThiNet-GAP	67.34%	87.92%	8.32M	9.34B	71.73/145.51
ThiNet-Tiny	59.34%	81.97%	1.32M	2.01B	29.51/55.83
SqueezeNet[15]	57.67%	80.39%	1.24M	1.72B	37.30/68.62

[1] In this paper, we only consider the FLOPs of convolution operations, which is commonly used for computation complexity comparison.

[2] For a fair comparison, the accuracy of original VGG-16 model is evaluated on resized center-cropped images using pre-trained model as adopted in [10, 14]. The same strategy is also used in ResNet-50.

- Taylor：泰勒展开的方法，衡量不同的权重对损失的影响
- Thing-WS：用权重和替换论文中提到的filter选择方法

denote the approximation value.

Method	Top-1 Acc.	Top-5 Acc.	#Param. ↓	#FLOPs ↓
APoZ-1 [14]	-2.16%	-0.84%	2.04×	≈ 1×
APoZ-2 [14]	+1.81%	+1.25%	2.70×	≈ 1×
Taylor-1 [23]	–	-1.44%	≈ 1×	2.68×
Taylor-2 [23]	–	-3.94%	≈ 1×	3.86×
ThiNet-WS [21]	+1.01%	+0.69%	1.05×	3.23×
ThiNet-Conv	+1.46%	+1.09%	1.05×	3.23×
ThiNet-GAP	-1.00%	-0.52%	16.63×	3.31×

4.3 ResNet-50 on ImageNet

用论文中提到的方法压缩ResNet-50

- 实验设置
  - 迭代的修剪块，除了filter，BN层也被丢弃
  - 微调：1 epoch、学习率 10^{-4} + 9epoch、学习率10^{-3} – 10^{-5} 以提高精度
- 实验结果

speed tested on one M40 GPU with batch size 32.

Model	Top-1	Top-5	#Param.	#FLOPs	f./b. (ms)
Original	72.88%	91.14%	25.56M	7.72B	188.27/269.32
ThiNet-70	72.04%	90.67%	16.94M	4.88B	169.38/243.37
ThiNet-50	71.01%	90.02%	12.38M	3.41B	153.60/212.29
ThiNet-30	68.42%	88.30%	8.66M	2.20B	144.45/200.67

  - 与VGG-16相比，ResNet-50模型更加紧凑，冗余较少，更具挑战
  - ThiNet-50 减少了超过一般的模型参数和计算量，且精确率下降了仅1%
  - 尽管计算量减少了很多，但并没有以相同幅度减少计算时间，BN层和池化层占据了大部分时间
  - 只修改了残差块的前两层，使块输出和残差连接部分不变——修剪这些部分能够进一步的压缩

4.4 Domain adaptation ability of the pruned model

- ThiNet 的主要优点是没有改变网络结构，因此在 ImageNet 上修剪的模型可以很容易地转移到其他领域。
- 以下是ThiNet在其他小数据集上的一些表现

Table 4. Comparison of different strategies to get a small model on CUB-200 and Indoor-67. "FT" stands for "Fine Tune".

Dataset	Strategy	#Param.	#FLOPs	Top-1
CUB-200	VGG-16	135.07M	30.93B	72.30%
	FT & prune	7.91M	9.34B	66.90%
	Train from scratch	7.91M	9.34B	44.27%
	ThiNet-Conv	128.16M	9.58B	70.90%
	ThiNet-GAP	7.91M	9.34B	69.43%
	ThiNet-Tiny	1.12M	2.01B	65.45%
	AlexNet	57.68M	1.44B	57.28%
Indoor-67	VGG-16	134.52M	30.93B	72.46%
	FT & prune	7.84M	9.34B	64.70%
	Train from scratch	7.84M	9.34B	38.81%
	ThiNet-Conv	127.62M	9.57B	72.31%
	ThiNet-GAP	7.84M	9.34B	70.22%
	ThiNet-Tiny	1.08M	2.01B	62.84%
	AlexNet	57.68M	1.44B	59.55%

- ThiNet-Tiny 在 ImageNet 上的准确度与 AlexNet 相同，但它显示出更强的泛化能力。当转移到参数少 50 倍的特定领域任务时，这个微型模型可以实现比 AlexNet 高 3% ~ 8% 的分类准确度。它的模型大小足够小，可以部署在资源受限的设备上。

5 结论

本篇论文提出了一个统一的框架——ThiNet，用于CNN模型的压缩和加速任务，filter级的方法与之前的方法相比有很大的改进，在基准测试集上取得了sota的效果；论文中也提到的下一步工作将准备修剪ResNet的projection shortcuts部分或者寻找channel级的修剪方法，又或者是将剪枝网络应用在不同计算机视觉任务中（如对象检测、语义分割）