ThiNet

A FILTER LEVEL PRUNING METHOD FOR DEEP NEURAL NETWORK COMPRESSION

报告人：许建荣

目　录

- 研究背景
- 文章的思想与效果
- ThiNet的框架
- 应用到现有的网络结构
- 实验效果
- 主要优点和主要贡献

研究背景

深度神经网络计算

Year	Model	Layers	Parameter	FLOPs	ImageNet Top-5 error
2012	AlexNet	5+3	60M	725M	16.4%
2013	Clarifai	5+3	60M	—	11.7%
2014	MSRA	5+3	200M	—	8.06%
2014	VGG-19	16+3	143M	19.6B	7.32%
2014	GoogLeNet	22	6.8M	1.566B	6.67%
2015	ResNet	152	19.4M	11.3B	3.57%

研究背景

从上表可以看出近年来网络层数越来越多，计算复杂度越来越高。而过高的计算复杂度通常要求我们使用GPU或者高性能的CPU对神经网络进行运算。实际上在深度学习应用过程中，我们还面临很多**诸如移动设备、嵌入式设备这样存在计算、体积、功耗等方面受限的设备（边缘智算设备）**，它们也需要应用深度学习技术。由于这些设备存在的约束，导致现有的高性能深度神经网络无法在上面**进行有效的计算和应用**。

- 模型压缩主要解决

典型的深度模型很难部署在**资源受限的设备上**，例如移动电话或嵌入式小配件。资源受限的场景意味着计算任务必须在有限的资源供应下完成，例如计算**时间，存储空间，电池电量**等。

深度神经网络计算

移动设备：算不好



穿戴设备：算不了


数据中心：算不起

文章的思想与效果

文章的思想与效果

- **思想**

  根据从**下一层计算的统计信息**来评估上一层的卷积核的重要程度来修剪过滤器

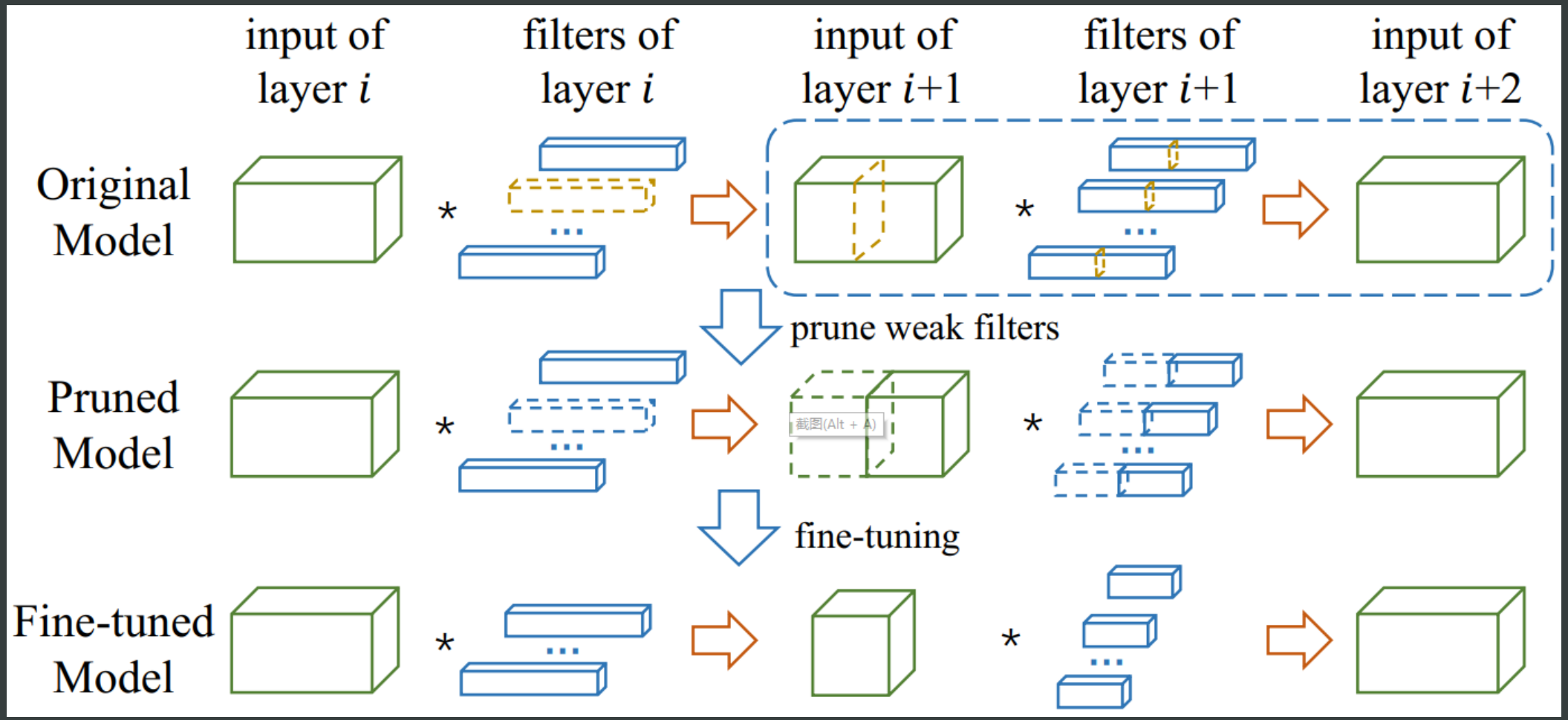
- **实验效果**

- ThiNet在ILSVRC-12（训练集包含1281167张图片，验证集包含50000张图片，测试集为100000张图片）基准测试中的性能。ThiNet在**VGG-16**上实现了**3.31×FLOP**减少和**16.63×**压缩，前5精度下降仅为0.52%。使用**ResNet-50**的类似实验表明，即使对于紧凑型网络，ThiNet也**可以减少一半以上的参数和FLOP**，但成本大约为前5精度下降1%。此外，原始的VGG-16型号可以进一步修剪成一个只有**5.05MB**型号的非常小的模型，保持AlexNet级别的准确性，但显示出更强的泛化能力。

ThiNet的框架

ThiNet的框架

1. **滤波器选择**，利用layer i+1 的input channels来选择剪枝layer i哪些部分。
2. **剪枝**，去除layer i+1的弱input channels所对应的layer i的滤波器。
3. **微调**，修复模型，为节省时间，对一层剪枝后仅训练一两个epochs，当整个网络都剪枝后，再训练足够多轮。
4. **循环**至第一步对下一层网络剪枝。


input of layer i
filters of layer i
input of layer i+1
filters of layer i+1
input of layer i+2
Original Model
prune weak filters
Pruned Model
fine-tuning
Fine-tuned Model

滤波器选择1(即卷积核)

- 一个卷积操作可以用下面这个式子表示。**W**是卷积核，**x**是输入，**b**是偏置，**c**表示输入channel。

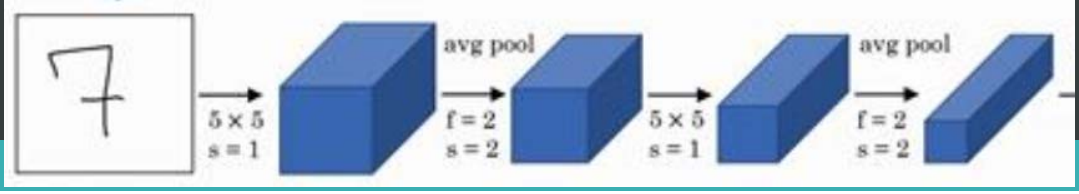
$$y = \sum_{c=1}^{C} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \widehat{W}_{c,k_1,k_2} \times x_{c,k_1,k_2} + b. \qquad (1)$$

- 如果我们定义：

$$\hat{x}_c = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \widehat{W}_{c,k_1,k_2} \times x_{c,k_1,k_2}, \qquad (2)$$

- 那么式子1就可以写成下面这个形式：

$$\hat{y} = \sum_{c=1}^{C} \hat{x}_c, \qquad (3)$$



滤波器选择2(即卷积核)

- 如果定义一个集合S（保留的集合）：

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

- **文中采用的是贪心算法来求解最优的集合S。**假设输入是：

$$\{(\hat{\mathbf{x}}_i, \hat{y}_i)\}$$

- 那么就能得到下面这个优化式子：
- **R** 表示压缩率。也就是说我们需要找到能使得这个优化式子最小的channel集合S。作者采用**贪心算法来求解这个S**，由于S包含channel较多，因此求解速度会很慢，因此**定义另一个集合T**，集合T所包含的channel要少于S

$$\arg\min_{S} \sum_{i=1}^{m} \left(\hat{y}_i - \sum_{j \in S} \hat{\mathbf{x}}_{i,j} \right)^2 \tag{5}$$
$$\text{s.t.} \quad |S| = C \times r, \quad S \subset \{1, 2, \ldots, C\}.$$

滤波器选择3(即卷积核)

- 满足：

$$S \cup T = \{1, 2, \ldots, C\} \text{ and } S \cap T = \emptyset$$

S： 保留的集合

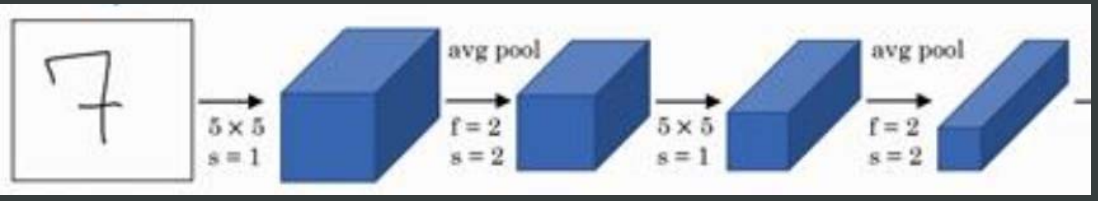
T： 移除的集合

- 这样就能将式子5转换成如下的优化式子，这样求解最优值的速度会更快。
- 即求剪掉的部分

$$\arg\min_{T} \sum_{i=1}^{m} \left(\sum_{j \in T} \hat{\mathbf{x}}_{i,j} \right)^2 \tag{6}$$
$$\text{s.t.} \quad |T| = C \times (1 - r), \quad T \subset \{1, 2, \ldots, C\}.$$

算法实现

- 综上，求最优集合T的算法如下：



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

应用到现有的网络结构

应用到现有的网络结构

- **VGG-16**

由于前面10层卷积占据了90%的计算量，而全连接层又占据了86%的参数，**因此作者采用对前面10层卷积层进行prune，达到加速目的，另外将所有全连接层用一个global average pooling层代替。**

- **ResNet-50**

只对一个block的前两层卷积做prune，而不动最后一个卷积层，如图Figure3。

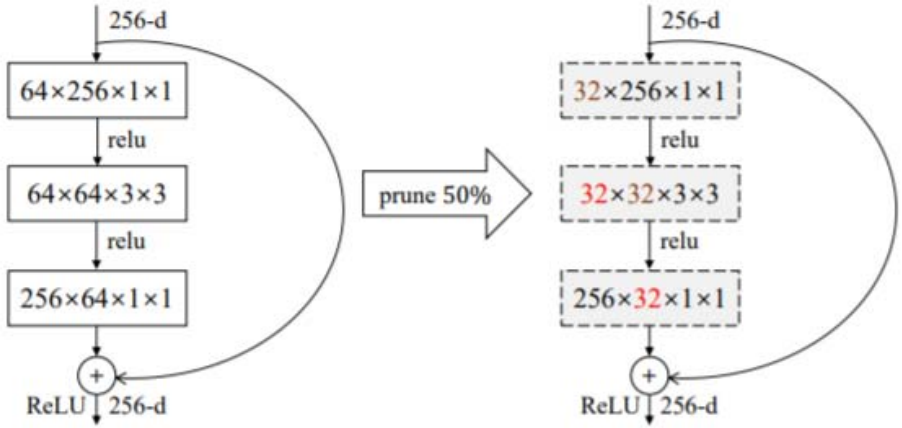


Figure 3. Illustration of the ResNet pruning strategy. For each residual block, we only prune the first two convolutional layers, keeping the block output dimension unchanged.

实验效果

实验效果1

- Table1是在VGG-16网络上应用ThiNet压缩。其中**ThiNet-Conv**表示只对前面10个卷积层进行压缩，**ThiNet-GAP**表示在对卷积层压缩的基础上还将所有的全连接层用global average pooling代替。**ThiNet-Tiny是压缩率较高的网络，主要是和SqueezeNet做对比。**

Table 1. Pruning results of VGG-16 on ImageNet using ThiNet. Here, M/B means million/billion (10^6/10^9), respectively; f./b. denotes the forward/backward timing in milliseconds tested on one 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

实验效果2

- **Table2也是VGG-16，这是和其他基于filter做prune的算法的对比。**
  APoZ-1和APoZ-2的差别在于后者prune的卷积层数量更少。Taylor算法对filter做prune的依据在于如果一个filter的存在对减少loss没有帮助，则可以去掉

Table 2. Comparison among several state-of-the-art pruning methods on the VGG-16 network. Some exact values are not reported in the original paper and cannot be computed, thus we use ≈ to 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×

实验效果3

- **Table3**是在ResNet-50上应用ThiNet进行压缩和加速的结果。ResNet-50在网络结构设计上本身存在的冗余就比较少，所以可压缩的空间比VGG-16要少很多。另外由于在对ResNet-50压缩时候只是压缩大部分卷积层，而没有压缩像BN，pooling层等，但是这些层在测试的时候会占用约40%的GPU计算资源，因此测试时候的加速不会太明显。

Table 3. Overall performance of pruning ResNet-50 on ImageNet via ThiNet with different compression rate. Here, M/B means million/billion respectively, f./b. denotes the forward/backward 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

主要优点和主要贡献。

主要优点和主要贡献。

- 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在许多任务上显示出对现有方法的显着改进。该方法不会改变原有的网络结构，因此任何现成的深度学习库都可以完美支持它。（**与非结构化对比**）
- 正式将过滤修剪作为优化问题，并揭示我们需要修剪过滤器，**从下一层计算的统计信息**，而不是当前层，这将ThiNet与现有方法区分开来。
- 在实验中，**VGG-16模型可以修剪成5.05MB**，显示出有希望的转移学习泛化能力。使用ThiNet可以使用更精确的模型保留更高的精度。

谢谢