

深度模型压缩与加速笔记整理

by jiapy

0.背景

总所周知，神经网络功能强大。但其巨大的存储和计算代价也使得其实用性特别是在移动设备上的应用受到了很大限制。

Krizhevsky 在 2014 年文章中提出两点观察结论：卷积层占据了大约 90-95% 计算时间和参数规模，有较大的值；全连接层占据了大约 5-10% 的计算时间，95% 的参数规模，并且值较小。这为后来的研究深度模型的压缩与加速提供了统计依据。

典型例子是具有 50 个卷积层 ResNet-50 需要超过 95MB 存储器以及 38 亿次浮点运算。在丢弃了一些冗余权重后，网络仍可照常工作，但节省了超过 75% 参数和 50% 计算时间。

1. 深度模型压缩方法分类及特点

1.1 方法分类

现有深度模型压缩方法，主要分为 4 类：

(1)参数修剪和共享（parameter pruning and sharing）

基于参数修剪和共享的方法针对模型参数的冗余性，试图去除冗余和不重要的项。

(2)低秩因子分解（low-rank factorization）

基于低秩因子分解的技术使用矩阵/张量分解来估计深度学习模型的信息参数。

(3)转移/紧凑卷积滤波器（transferred/compact convolutional filters）

基于传输/紧凑卷积滤波器的方法设计了特殊的结构卷积滤波器来降低存储和计算复杂度。

(4)知识蒸馏（knowledge distillation）

知识蒸馏方法通过学习一个蒸馏模型，训练一个更紧凑神经网络来重现一个更大网络输出。

方法名称	描述	应用场景	方法细节
剪枝和共享	删除对准确率影响不大的参数	卷积层和全连接层	对不同设置具有鲁棒性，可以达到较好效果，支持从零训练和预训练
低秩分解	使用矩阵对参数进行分解估计	卷积层和全连接层	标准化的途径，很容易实施，支持从零训练和预训练
转移、紧凑卷积核	设计特别的卷积核来保存参数	只有卷积层	算法依赖于应用程序，通常可以取得好的表现，只能从零开始训练
知识蒸馏	训练一个更紧凑的神经网络来从大的模型蒸馏知识	卷积层和全连接层	模型表现对应用程序和网络结构较为敏感，只能从零开始训练

1.2 方法特点

一般来说，参数修剪和共享，低秩分解和知识蒸馏方法可以用于全连接层和卷积层 CNN，使用转移/紧凑型卷积核的方法仅支持卷积层。

通过减少神经元之间连接或通道数量进行剪枝，在压缩加速中较为有效。但会对下一层输入造成严重影响。低秩因子分解和基于转换/紧凑型卷积核的方法提供了一个端到端流水线，可以很容易在 CPU/GPU 环境中实现。相反参数修剪和共享使用不同的方法，如向量量化，二进制编码和稀疏约束来执行任务，这导致常需要几个步骤才能达到目标。

使用迁移卷积层对 CNN 模型进行压缩受到 Cohen 等变群论（equivariant group

theory) 启发。使用紧凑卷积滤波器可以直接降低计算成本。

Inception 结构中使用了将 3×3 卷积分解成两个 1×1 的卷积;

SqueezeNet 提出用 1×1 卷积来代替 3×3 卷积, 与 AlexNet 相比, SqueezeNet 创建了一个紧凑的神经网络, 参数少了 50 倍, 准确度相当。

这些方法擅长处理广泛/平坦体系结构(如 VGGNet), 而不是狭窄/特殊(如 GoogleNet, ResidualNet)。其次, 转移假设有时过于强大, 导致某些数据集结果不稳定。结构化矩阵和迁移卷积滤波器方法必须使模型具有较强的人类先验知识, 这对模型性能和稳定性有显著影响。研究如何控制强加先验知识的影响是很重要的;

知识蒸馏, 将深度和宽度网络压缩成较浅网络, 压缩模型模拟复杂模型所学习的功能, 主要思想通过学习通过 softmax 获得类分布输出, 将知识从一个大模型转移到一个小模型。Hinton 引入了知识蒸馏压缩框架, 通过遵循“学生-教师”范式减少深度网络训练量, 通过软化“教师”输出而惩罚“学生”。为了完成这一点, 学生学要训练以预测教师输出, 即真实分类标签。方法简单, 但在各种图像分类任务中表现出较好结果。知识精炼方法有很多优势, **基于知识蒸馏方法能令更深模型变得更加浅而显著地降低计算成本。**如不需要特定硬件或实现就能直接加速模型。缺点, **只能用于具有 Softmax 损失函数分类任务, 阻碍其应用。模型假设有时太严格, 性能有时比不上其它方法。**

多种小型平台(例如移动设备、机器人、自动驾驶汽车)硬件限制仍然是阻碍深层 CNN 发展的主要问题。相比于压缩, 可能模型加速要更为重要, 专用芯片出现固然有效, 但从数学计算上将乘加法转为逻辑和位移运算也是一种很好的思路。

2. VALSE 2017 | 神经网络模型压缩优化方法

2.0 报告简介

深度学习算法是计算密集型和存储密集型, 这使得它难以被部署到资源有限的嵌入式系统上。优化一般有以下两个方向:

- 1、**通过减少参数数量**, 达到模型压缩目的。而压缩基于一个很重要的理论, 即神经网络模型通常是过参数化的, 通常不需要那么多参数 就可以表达出模型特征。
- 2、**通过节省计算**, 降低计算量, 达到模型运算加速目的。

2017VALSE 大会上, 关于神经网络模型的压缩、加速、优化主要有如下几个报告:

- 1、孙剑, 介绍了旷视科技在网络模型加速和压缩方面的工作。
- 2、中科院自动化所程健, 作了“深度神经网络快速计算方法”的 tutorial。
- 3、新加坡国立颜水成教授作了“深度学习的三个维度: Compactness, Speed, and Accuracy”的特邀报告。

2.1 孙剑: 简化网络设计方法

介绍了模型压缩优化。他举了个例子, 比如在对图像分类的时候, 随着层级增加, 应该把图像空间分辨率慢慢缩小, 但同时也需增加每一层中 filter 数。另外**实践中发现用小 filter 是更经济, 还有用 Low-rank 分解逼近方法也比较有效。**

简化网络方面, 主要考虑**结构剪枝**。结构剪枝是指对网络结构进行修剪; 结构化剪枝是属于结构剪枝一种具体方法, 按照特定结构(相对于随机)剪枝。结构剪枝共有 3 种方法。

(1)稀疏连接, 本来一个网络里有很多连接, 其基本思想是去除不重要连接, 让连接变稀疏。虽然它可以减少网络模型大小, 但是不一定能够减少网络运行时间。

(2)张量分解, 就是把一个卷积网络参数矩阵通过张量分解, 用它的低秩特性做逼近。

(3)channel 剪枝, 就是训练好一个网络后, 简单粗暴的把一些 channel 去掉。

还有一种方法是 **Low-bit 表达**。如输入一个三维 feature map, feature map 通常是 float

表示，卷积核也是一个三维矩阵，它也是 float 表示的。**Low-bit** 表达就是用低精度表达来代替这些高精度 float，比如用 8 位或者更加极端的两位来表示。

有两篇比较著名的工作。一个是 **Binary connect**，他的思想是把这个 weight 都变成 01，这也是很夸张的一个想法。下面是更进一步的工作，它是将 feature 和 weight 全变成 01，叫 **XNOR-Net**，好处是卷积神经网络里的矩阵层，可以变成一个 bitcount 的指令，就可以完成它想要完成的计算，这个是在硬件中很有效的一个方法，也是 Low-bit 网络非常吸引人的地方。优点在于：1.内存可以降得非常多；2.潜在的加速比非常大。

Low-bit 表达除了能量化 weight 或 feature，还可以量化 gradient，因为 gradient 其实也是 float 的。他们团队使用的设置是 weight 用 01 表示，activation 用两位表示，gradient 用 4 位表示，他们将这个网络取名为 **DOReFa-Net**。该网络结构在并行训练或者 FPGA/ASIC 上训练时可以提高不少效率。

2.2 程健研究员：嵌入式平台的深度学习优化方法

列举了近年来在神经网络模型加速和压缩方面的几个有效方法：

1、剪枝与稀疏

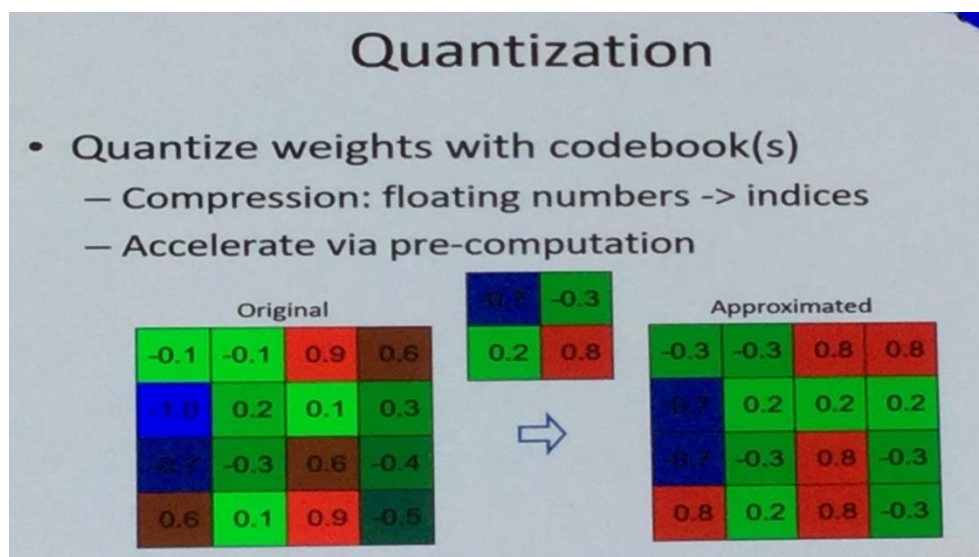
研究表明网络中很多连接都是接近 0 或冗余的，如何对这些参数进行稀疏很有意义。如 SVD 分解可从模型压缩和提高运算速度两方面优化。但并不是剪枝就一定会提高运算速度。如果剪枝过于随机，并不能有效提升运算速度。所以需要结构化剪枝（structured pruning）。剪枝一般需要三个步骤：先在原始的网络结构下训练网络，然后通过设置一个阈值，把一些权值较小的连接进行剪枝，重新训练权重，对训练好的模型再剪枝，再重新训练，直到满足设定条件为止。

2、低秩分解

有很多种分解方法，比如奇异值分解 SVD、tucker 分解、块分解。

3、权值量化

通常神经网络权值一般用单精度浮点数表示，需要占用大量存储空间。而用码书对权值进行量化可以共享权值，从而减轻存储负担。



利用哈希编码对权重进行编码，从而达到压缩权重的目的。也是一种有效的方法，如下所示。另外就是定点和二值化（fixed point / binary）的方法。量化方式有二值量化（0,1），也有三值量化。

BinaryConnect

- Forward-backward: Binary weights
- Gradient accumulation: float-point weights
- Deterministic:

$$w_b = \begin{cases} +1 & \text{if } w \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$
- Stochastic:

$$w_b = \begin{cases} +1 & \text{with probability } p = \sigma(w), \\ -1 & \text{with probability } 1 - p. \end{cases}$$

where σ is the "hard sigmoid" function:

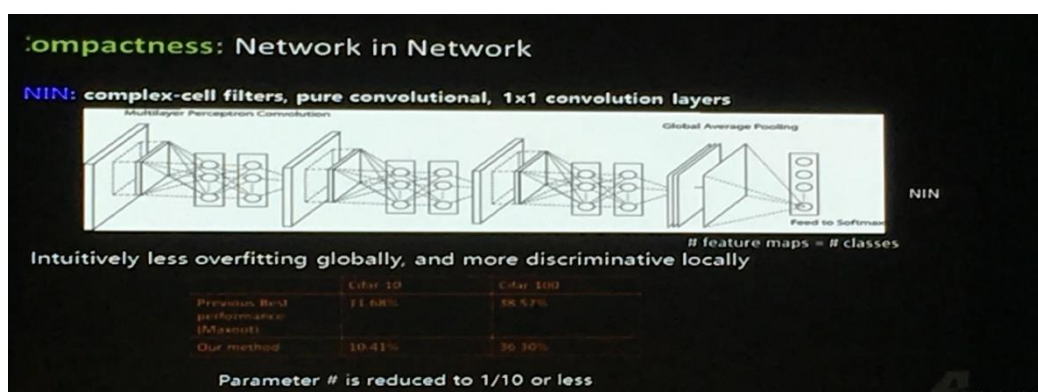
$$\sigma(x) = \text{clip}\left(\frac{x+1}{2}, 0, 1\right) = \max(0, \min(1, \frac{x+1}{2}))$$

Courbariaux et al., "BinaryConnect: Training Deep Neural Networks with binary weights during propagations".
NIPS 15

4、用定点来代替浮点数计算。

2.3 颜水成教授：深度学习的三个维度

介绍了 DL 三个维度：Compactness, Speed, and Accuracy。小模型、速度快、预测准。
小模型可以考虑 network in network。



极端化一点，是否可以考虑设计全部由 1x1 卷积层组成小模型？如下图所示就是使用 1x1 的卷积，它的基本结构包括以下三个部分：

经过研究统计发现，网络在经过 ReLU 后，有平均超过 40% 的输出都是 0，这些其实是无用信息，而且这些输出 0 值在经过 ReLU 前的操作也是没有意义的。可以想办法找到这些位置进行优化。

提高模型预测精度的一个有效方法是共享跨层信息，例如 ResNet 的跨层连接设计。
以上就是 VALSE2017 上关于深度学习网络模型压缩优化的部分选摘。

3. 2016ICLR 最佳论文

《Deep Compression: Compression Deep Neural Networks With Pruning, Trained Quantization And Huffman Coding》

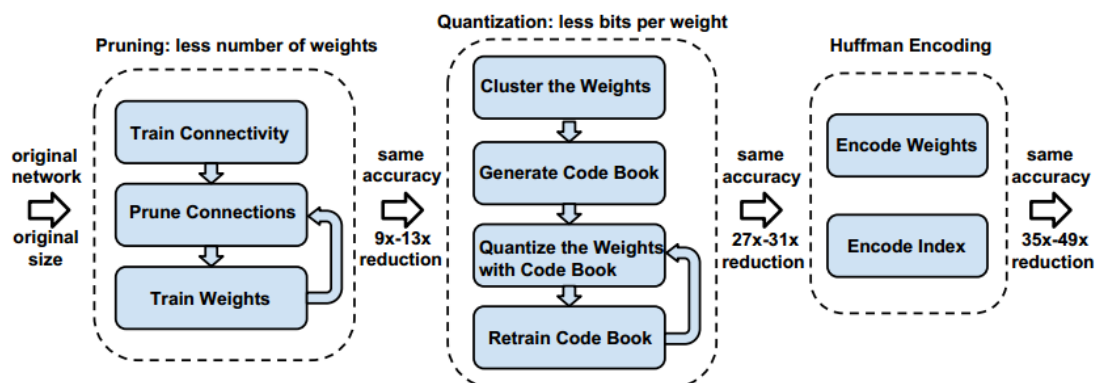
目标：降低大型神经网络存储和计算消耗，使其可在移动设备上运行，实现“深度压缩”。

3.1 论文亮点

实现过程主要有三步：

- (1) 通过移除不重要连接来对网络进行剪枝；
- (2) 对权重量化，使得许多连接共享同一权重，并且只需要存储码本(有效权重)和索引；

(3) 进行霍夫曼编码以利用有效权重有偏分布；



3.2 论文细节

“剪枝”详细点说也可以分为 3 步：（1）进行正常的网络训练；（2）删除所有权重小于一定阈值的连接；（3）对上面得到的稀疏连接网络再训练；

存储稀疏结构时采用的是稀疏压缩行 CSR 或者稀疏压缩列 CSC，假设非 0 元素个数为 a ，行或者列数为 n ，那么我们需要存储的数据量为 $2a+n+1$ 。以 CSR 为例，我们存储时采用的是 3 元组结构，即：行优先存储 a 个非零数，记为 A ； a 个非零数所在列的列号；每一行第一个元素在 A 中的位置+非零数个数。为进一步压缩，这里不存储绝对位置索引，而存储相对位置索引。如相对索引超过 8，就人为补 0。如下图：

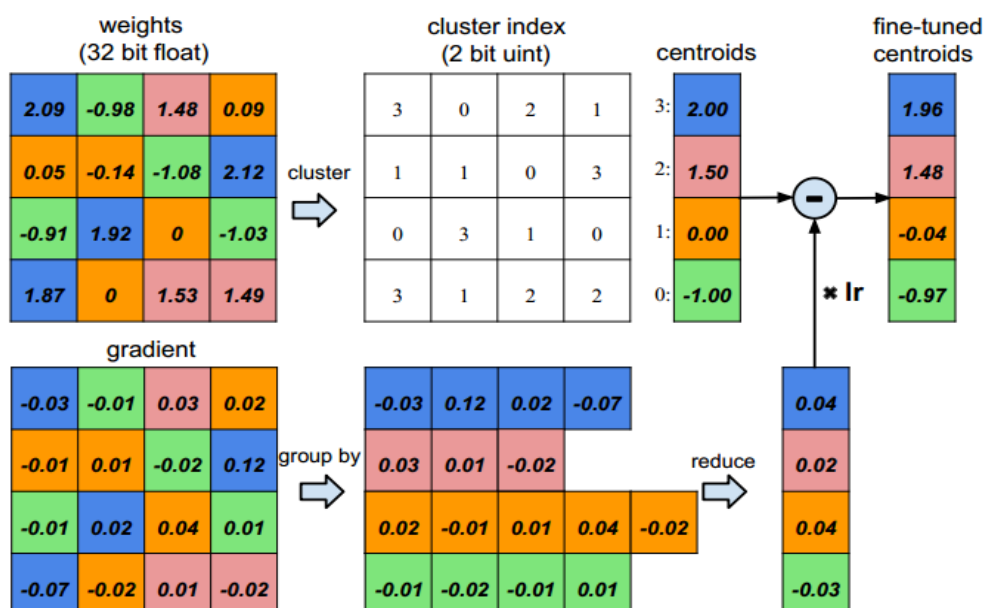
Span Exceeds $8=2^3$

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
diff		1			3								8			3
value		3.4			0.9								0			1.7

Filler Zero

Trained Quantization And Weight Sharing

这一部分通过对权重进行量化来进一步压缩网络（量化可以降低表示数据所用位数）。通过下图，我们来说明如何对权重进行量化，以及后续又如何对量化网络进行再训练。



首先，假设输入神经元 4 个，输出神经元 4 个，那么 weight 是 4x4，同样梯度也是。量化权重为 4 阶，即上图用 4 种不同颜色表示。就只需要存储 4 个码字以及 16 个 2bit 的索引。最后，应该如何再训练，即如何对量化值进行更新。事实上，我们仅对码字进行更新。具体如上图的下部分：计算相同索引处梯度之和，然后乘以学习率，对相应码字进行更新。实际中，AlexNet 作者对卷积层使用了 8bit 量化(即 256 个码字)，对全连接层使用了 5bit 量化(即 32 个码字)，但却不损失性能。

4. 结论：

核的稀疏化可能需要一些稀疏计算库支持，加速效果可能受到带宽、稀疏度等因素制约；其中基于核稀疏化方法，主要是在参数更新时增加额外惩罚项，来诱导核稀疏化，然后就可利用裁剪或者稀疏矩阵相关操作来实现模型压缩；

模型裁剪简单明了，直接在原有模型上剔除掉不重要 filter，虽然压缩方式比较粗糙，但神经网络自适应能力很强，加上大模型往往冗余较多，将一些参数剔除后，通过一些 retraining 手段可将由剔除参数而降低的性能恢复回来，因此只需挑选一种合适裁剪手段以及 retraining 方式，就能够有效在已有模型基础上很大程度压缩，是目前使用最普遍方法。基于模型裁剪方法，主要是对已训练好网络进行压缩，往往就是**寻找一种更加有效的评价方式，将不重要参数剔除**，以达到模型压缩目的，这种压缩方法实现简单，尤其是 regular 的方式，裁剪效率最高，但是如何寻找一个最有效评价方式是最重要的。

基于教师——学生网络，基于迁移学习的方法，利用一个性能好的教师网络来监督学生网络进行学习，大大降低了简单网络学习到不重要信息比例，提高了参数利用效率，也是目前用的较多的方法。







基于精细模型设计的方法，模型本身体积小，运行速度快，性能也不错，目前小的高效模型也开始广泛运用在各种嵌入式平台中。

总的来说，以上几种方法可结合使用，如说先对参数进行结构化限制，使得参数裁剪起来更加容易，然后再选择合适裁剪方法，考虑不同评价标准以及裁剪策略，并在裁剪过程中充分考虑参数量、计算量、带宽等，以及不同硬件平台特性，在模型性能、压缩、以及平台上的加速很好的进行权衡，达到更好效果。

目前使用最为广泛的方法为：**基于模型裁剪的方法，原理简单，且裁剪效率相对较高**。建议是否可以从模型裁剪这个领域入手，先做一下尝试？看看具体效果

5. 重要 Paper

详见论文附件压缩包。

 基于核的稀疏化	2019/1/2 22:09	文件夹
 基于教师-学生	2019/1/2 22:24	文件夹
 基于精细化设计	2019/1/2 22:28	文件夹
 基于模型裁剪	2019/1/2 22:19	文件夹
 250-optimal-brain-damage-LeCun-Yan.pdf	2019/1/2 22:05	PDF 文件
 Deep Compression-Song Han.pdf	2019/1/2 22:04	PDF 文件

Reference

[1]Valse 2017 | 神经网络模型压缩优化方法：<https://blog.csdn.net/electech6/article/details/72822009>

[2]深度学习模型压缩方法综述：<https://blog.csdn.net/wspba/article/details/75671573>

[3]大部分参考自 CSDN、知乎等技术平台