DOI:10.7544/issn1000-1239.2018.20180129
55(9):1871-1888,2018

深度神经网络压缩与加速综述

纪荣嵘[1,2] 林绍辉[1,2] 晁 飞[1] 吴永坚[3] 黄飞跃[3]
[1](厦门大学信息科学与技术学院 福建厦门 361005)
[2](福建省智慧城市感知与计算重点实验室(厦门大学) 福建厦门 361005)
[3](上海腾讯科技有限公司优图实验室 上海 200233)
(rrji@xmu.edu.cn)

Deep Neural Network Compression and Acceleration: A Review

Ji Rongrong[1,2], Lin Shaohui[1,2], Chao Fei[1], Wu Yongjian[3], Huang Feiyue[3]
[1](*School of Information Science and Engineering, Xiamen University, Xiamen, Fujian*, 361005)
[2](*Fujian Key Laboratory of Sensing and Computing for Smart City (Xiamen University), Xiamen, Fujian*, 361005)
[3](*BestImage Laboratory, Tencent Technology (Shanghai) Co.,Ltd, Shanghai*, 200233)

Abstract In recent years, deep neural networks (DNNs) have achieved remarkable success in many artificial intelligence (AI) applications, including computer vision, speech recognition and natural language processing. However, such DNNs have been accompanied by significant increase in computational costs and storage services, which prohibits the usages of DNNs on resource-limited environments such as mobile or embedded devices. To this end, the studies of DNN compression and acceleration have recently become more emerging. In this paper, we provide a review on the existing representative DNN compression and acceleration methods, including parameter pruning, parameter sharing, low-rank decomposition, compact filter designed, and knowledge distillation. Specifically, this paper provides an overview of DNNs, describes the details of different DNN compression and acceleration methods, and highlights the properties, advantages and drawbacks. Furthermore, we summarize the evaluation criteria and datasets widely used in DNN compression and acceleration, and also discuss the performance of the representative methods. In the end, we discuss how to choose different compression and acceleration methods to meet the needs of different tasks, and envision future directions on this topic.

Key words DNN compression; DNN acceleration; parameter pruning; parameter sharing; low-rank decomposition; knowledge distillation

摘 要 深度神经网络在人工智能的应用中,包括计算机视觉、语音识别、自然语言处理方面,取得了巨大成功.但这些深度神经网络需要巨大的计算开销和内存存储,阻碍了在资源有限环境下的使用,如移

收稿日期:2018-02-21;**修回日期**:2018-06-08
基金项目:国家重点研发计划项目(2017YFC0113000,2016YFB10015032);国家自然科学基金项目(U1705262,61772443,61402388,61572410);国家自然科学基金优秀青年科学基金项目(61422210);福建省自然科学基金项目(2017J01125)
This work was supported by the National Key Research and Development Program (2017YFC0113000, 2016YFB1001503), the National Natural Science Foundation of China (U1705262, 61772443, 61402388, 61572410), the National Natural Science Foundation of China for Excellent Young Scientists (61422210), and the Natural Science Foundation of Fujian Province of China (2017J01125).
通信作者:林绍辉(shaohuilin007@gmail.com)

动或嵌入式设备端.为解决此问题,在近年来产生大量关于深度神经网络压缩与加速的研究工作.对现有代表性的深度神经网络压缩与加速方法进行回顾与总结,这些方法包括了参数剪枝、参数共享、低秩分解、紧性滤波设计及知识蒸馏.具体地,将概述一些经典深度神经网络模型,详细描述深度神经网络压缩与加速方法,并强调这些方法的特性及优缺点.此外,总结了深度神经网络压缩与加速的评测方法及广泛使用的数据集,同时讨论分析一些代表性方法的性能表现.最后,根据不同任务的需要,讨论了如何选择不同的压缩与加速方法,并对压缩与加速方法未来发展趋势进行展望.

关键词 深度神经网络压缩;深度神经网络加速;参数剪枝;参数共享;低秩分解;知识蒸馏

中图法分类号 TP391

近年来,随着 GPU 的快速发展及大数据时代的来临,深度神经网络(deep neural networks, DNNs)已席卷人工智能各个领域[1],包括了语音识别[2]、图像理解[3-5]、自然语言处理[6]等在内的广泛领域.不仅如此,这些深度神经网络在复杂的系统中也得到了广泛使用,如自动驾驶[7]、癌症检测[8]、复杂游戏的策略搜索[9]等.深度神经网络在很多识别任务中已大大超越了人类识别的准确率,同时突破了传统的技术方法(如手工设计的(hand-crafted)特征 SIFT[10]和 HOG[11])带来的巨大的性能提升.这些性能的提升是因为深度神经网络拥有对大数据高层(high-level)特征提取的能力,从而获得对输入空间或数据的有效表示.另外,相比传统在 CPU 平台上的计算,强大的 GPU 计算能力大大提高了深度神经网络的计算效率,使得模型的训练速度得到大幅度提升.如果能将计算平台小型化,即把深度神经网络强大的识别性能移植到移动嵌入式设备中(如手机、机器人、无人机、智能眼镜等),这样不管在军事方面的抢险救灾、敌情勘探,还是在民事方面的移动智能识别、便民出行等都起到重大的促进作用.

伴随着深度神经网络模型的性能增加,神经网络的深度越来越深,接踵而来的是深度网络模型的高存储高功耗的弊端,严重制约着深度神经网络在资源有限的应用环境和实时在线处理的应用,特别是智能化移动嵌入式设备、现场可编程门阵列(field-programmable gate array, FPGA)等在线学习和识别任务.例如 8 层的 AlexNet[3]装有 600 000 个网络节点、0.61 亿个网络参数,需要花费 240 MB 内存存储和 7.29 亿浮点型计算次数(FLOPs)来分类一副分辨率为 224×224 的彩色图像.同时,随着神经网络模型深度的加深,存储的开销将变得越大.同样来分类一副分辨率为 224×224 的彩色图像,如果采用拥有比 8 层 AlexNet 更多的 16 层的 VGGNet[12],则有 1 500 000 个网络节点、1.44 亿个网络参数,需要花费 528 MB 内存存储和 150 亿浮点型计算次数;ResNet-152[13]装有 0.57 亿个网络参数,需要花费 230 MB 内存存储和 113 亿浮点型计算次数.对于移动终端设备、FPGA 等弱存储、低计算能力特点,无法直接存储和运行上述如此庞大的深度网络.

一方面,拥有百万级以上的深度神经网络模型内部存储大量冗余信息,因此并不是所有的参数和结构都对产生深度神经网络高判别性起作用;另一方面,用浅层或简单的深度神经网络无法在性能上逼近百万级的深度神经网络.因此,通过压缩和加速原始深度网络模型,使之直接应用于移动嵌入式设备端,将成为一种有效的解决方案.

一般情况下,压缩和加速深度神经网络是 2 个不同的任务,两者之间存在区别,但又紧密联系.例如卷积神经网络(convolutional neural networks, CNNs)分 2 种类型的计算层,即卷积层和全连接层.1)卷积层,它是计算耗时最大的层,也是卷积神经网络能够获得高层语义信息重要层.在卷积层内,可以通过权值共享,减少了对权值的大量存储.2)全连接层,不同于卷积层的局部感知,在全连接层中,每一个输出单元都与所有输入单元相关,通过密集的权值进行连接,因此需要大量的参数.因为卷积层与全连接层内在的本质区别,通常把卷积层的计算加速和全连接层的内存压缩认为是 2 种不同的任务.这 2 类计算层之间又是紧密联系的,卷积层为全连接层提供分层的高层特征,全连接层通过分类指导卷积层的高判别力特征提取.在本文中,我们总结与回顾近几年来压缩和加速深度神经网络方面的相关工作,相关算法与解决方案涉及多门学科,包括机器学习、参数优化、计算架构、数据压缩、索引、硬件设计等.

主流的压缩与加速神经网络的方法可以分成 5 种:1)参数剪枝(parameter pruning);2)参数共享(parameter sharing);3)低秩分解(low-rank decom-

position);4)紧性卷积核的设计(designing compact convolutional filters);5)知识蒸馏(knowledge distillation).参数剪枝主要通过设计判断参数重要与否的准则,移除冗余的参数.参数共享主要探索模型参数的冗余性,利用 Hash 或量化等技术对权值进行压缩.低秩分解利用矩阵或张量分解技术估计并分解深度模型中的原始卷积核.紧性卷积核的设计主要通过设计特殊的结构化卷积核或紧性卷积计算单元,减少模型的存储与计算复杂度.知识蒸馏主要利用大型网络的知识,并将其知识迁移到紧性蒸馏的模型中.

如表 1 所示,简要介绍了上述主流的 5 种压缩与加速深度神经网络的方法,一般来说,除了紧性卷积核的设计只能用于卷积核外,剩余的 4 种均能应用于卷积层和全连接层.低秩分解与紧性卷积核的设计能够在 CPU/GPU 下简单实现端对端(end-to-end)训练,然而参数共享、参数剪枝需要多步或逐层完成压缩与加速任务.关于训练过程中是否需要重新开始训练还是依赖于预训练模型的问题上,参数共享、低秩分解都较为灵活有效,既能适应重新训练也能适应于预训练模型.然而紧性卷积核的设计和知识蒸馏只能支持重新训练,另外参数剪枝只能依赖于预训练模型.从上述方法能否相应组合方面,紧性卷积核的设计方法和知识蒸馏还不能结合其他方法,参数剪枝与参数共享或低秩分解方法结合甚密,通过 2 种方法的互相融合,能够在一定程度进一步压缩与加速深度网络.

在本文中,主要集中分析在图像处理中的卷积神经网络的压缩与加速,特别是对于图像分类任务.本文先回顾近年来较为经典的深度神经网络,然后详细介绍上述压缩与加速方法、相应的特性及缺点,进而分析深度模型压缩与加速的评测标准、相应数据集及性能表现,最后总结并讨论不同压缩与加速方法的选择问题,并分析了未来发展趋势.

Table 1 Summarization of Different Methods for DNN Compression and Acceleration

表 1 不同深度神经网络压缩与加速方法总结

Category	Explanation	Applications (CONV, FC)	Training from scratch or pre-trained model	Combination
Parameter Pruning	Pruning unsalient parameters by measuring the importance of parameters	Both	Pre-trained model	Support, often with parameter quantization and low-rank decomposition
Parameter Sharing	Reducing redundant parameters using hashing or quantization methods	Both	Both	Support, often with parameter pruning
Low-rank Decomposition	Decomposing generalized convolution into a sequence of matrix/tensor-based convolutions with fewer parameters	Both	Both	Support, often with parameter pruning
Designing Compact Convolutional Filters	Reducing the storage and computation cost by designing compact convolutional filters	CONV	Training from scratch	
Knowledge Distillation	Training a compact neural network with distilled knowledge of a large model	Both	Training from scratch	

1 深度神经网络相关概念与回顾

在本节中,我们主要介绍了深度神经网络的发展历史,以及深度神经网络中前馈网络(feed forward networks)代表卷积神经网络的核心组件和经典网络模型.

1.1 深度神经网络发展历史

虽然 20 世纪 40 年代神经网络已经被提出,但一直没能得到实践应用,直到 20 世纪 90 年代,LeCun 等人提出了 LeNet 模型[14],并成功地将该模型用于手写体字符识别任务上,后续在 ATM 机上得到了广泛的应用.在此之后神经网络发展遇到瓶颈,直到 2006 年,Hinton 等人[15]提出了一种有效的训练深度神经网络的策略,不仅提升了模型的准确率,同时还极大地推动了非监督学习的发展.到了 2010 之后,深度神经网络得到了广泛的应用,特别是 2011 年微软公司的语音识别系统[2]突破了传统语音识别技术及 2012 年 AlexNet[3]的问世给整个图像识别领域带来了巨大突破,也使得深度神经网络席卷整个人工智能领域埋下伏笔.如表 2 所示,简要总结了深度学习的重要发展里程碑.

Table 2 The Development History of DNN

表 2 深度神经网络的发展历史

Time	DNN Development
1940s	Neural networks were proposed
1960s	Perceptron were proposed
1989	Neural networks for recognizing digits (LeNet-5)
2006	Deep autoencoder networks were proposed
2011	Breakthrough DNN-based speech recognition (Microsoft)
2012	DNNs for vision start supplanting hand-crafted approaches (AlexNet)
2013+	Rise of DNN compression and acceleration research

深度学习技术之所以能够在近几年来取得巨大突破,主要有 2 个原因:

1) 硬件设备计算能力的增强. 半导体设备和计算架构的提高(如 GPU),大大缩短网络计算的时间开销,包括训练(training)过程与推测过程(inference).

2) 训练数据集的不断扩充. 如图 1 所示,表示了不同时期公共数据集的数据规模. 从图 1 中可以看出,2010 年之前数据量规模停留在万级及以下,到了 2010 年之后大规模数据集的公开与使用,特别是大规模图像识别比赛(ILSVRC)[17]的成功举办,ImageNet 数据集[18]得到广泛的使用,给深度神经网络提高了数据支持,也使得拥有百万级以上参数的深度网络训练不易出现过拟合问题(overfitting).

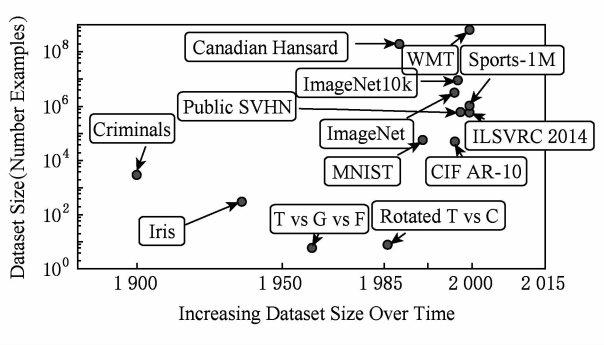


Fig. 1 The number of public training dataset over different years[16]

图 1 不同时期公共训练数据集的规模

结合以上 2 个原因,深度学习在各个人工智能领域大放异彩,特别在图像识别领域表现更加出众. 如图 2 所示,描述了在 ILSVRC 上的最好识别性能及所用的网络模型对比. 一方面,在 2012 年以前,所用的模型采用浅层网络表示(如支持向量机(SVMs)),获得 25%及以上的识别误差率;2012 年,AlexNet 利用深度学习技术将分类错误率较原来的浅层模型降低了近 10%. 自此以后,深度学习技术成为提高性能的主流与趋势. 特别是 2016 年,何凯明等人[13]提出了 ResNet,获得了 3.57%的 top-5 分类错误率($err_{\text{top-5}}$),成功超越了人类所能达到的 5%的分类错误率. 另一方面,随着分类错误率逐年降低,深度网络模型的层数越来越深,模型存储量大、计算复杂度高也越发明显,这触发了工业界及学术界对深度网络压缩与加速的关注与研究.

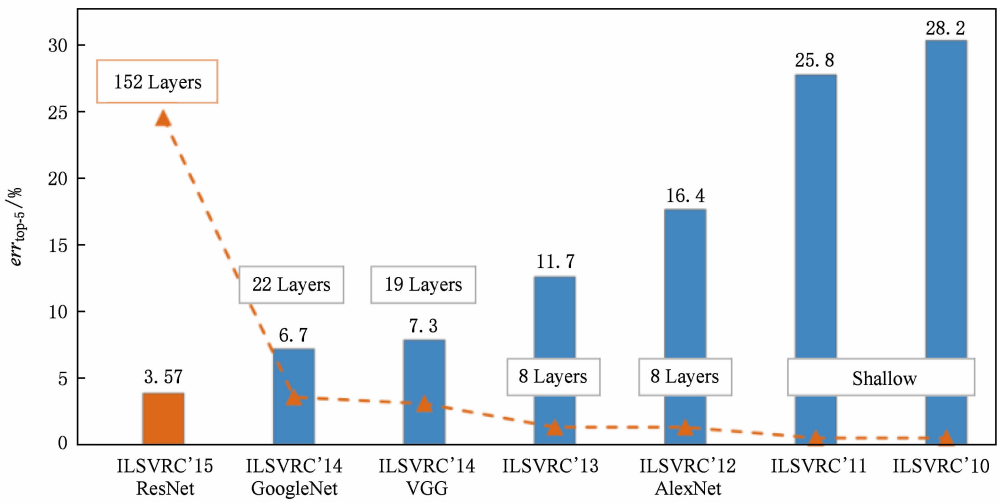


Fig. 2 Comparison of the best performance over different years using various CNNs in ILSVRC[9]

图 2 不同年份 ILSVRC 上的最好性能及所用的网络模型对比

1.2 卷积神经网络的相关术语及核心部件

卷积神经网络由多个卷积层(CONV layers)和全连接层(FC layers)组合而成. 如图 3 所示,在该网络卷积层中,每层产生了对输入图像的高层抽象

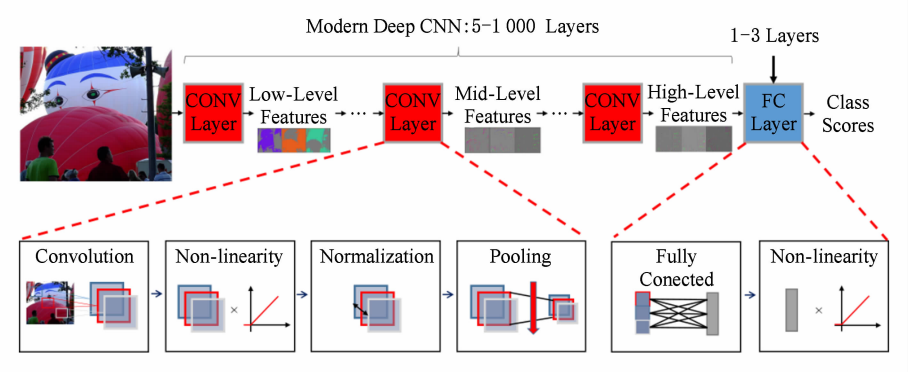
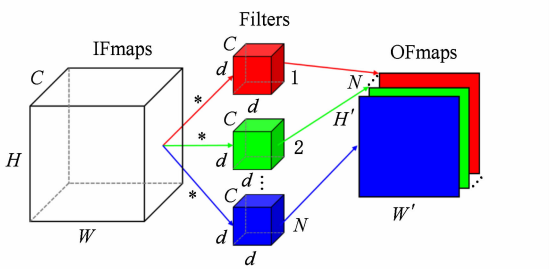


Fig. 3 Convolutional neural networks

图 3 卷积神经网络

表示的结果,称之为特征图(feature map, Fmap).底层的特征图能够获得输入图像的细节信息(如边缘、角点、颜色等),越往顶层越能获得输入图像的整体信息(如形状、轮廓等),正是卷积神经网络利用了此分层特性,取得了很好的性能提升.

如图 4 所示,显示了每个卷积层包含了复杂的高维卷积计算.输入中包含了一些列的 2-D 输入特征图(input feature maps, IFmaps),其中每一个特征图称之为通道(channel).每一个输出特征图(output feature maps, OFmap)是由所有的输入特征图与 3-D 卷积核/滤波(filter)卷积计算的结果,每一个 3-D 卷积核形成一个输出特征图.



C, H, W, D, N, H' and W' are defined in Equation (1).

Fig. 4 High-dimension convolutional operator

图 4 高维卷积计算

具体地,计算过程可以用表示为

$$\mathcal{O}_{h',w',n}=\sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{c=1}^{C}\mathcal{K}_{i,j,c,n}\mathcal{I}_{h_i,w_j,c}, \tag{1}$$

其中,$\mathcal{I}\in\mathbb{R}^{H\times W\times C}$是输入特征图,$\mathcal{O}\in\mathbb{R}^{H'\times W'\times N}$是输出特征图,$\mathcal{K}\in\mathbb{R}^{d\times d\times C\times N}$是一系列 3-D 卷积核.在卷积核 $\mathcal{K}$ 中,$d\times d$ 表示卷积核空间的维度——高和宽,C 和 N 分别表示输入和输出通道的个数.输入的高和宽可分别表示为 $h_i=h'+i-1$ 和 $w_j=w'+j-1$.为了简单表示,式(1)的卷积计算假定卷积步幅(stride)为 1,没有进行 0 填充(zero-padding),及忽略偏置(bias).

在全连接层中,不像卷积层的权值共享,全连接层的输入与输出之间存在密集连接(dense connection),需要保存大量训练参数.其中核心的计算单元是矩阵与矩阵之间的乘法,具体可由式表示:

$$\boldsymbol{Z}=\boldsymbol{W}\boldsymbol{X}, \tag{2}$$

其中,$\boldsymbol{X}\in\mathbb{R}^{d\times b}$为输入矩阵,$\boldsymbol{W}\in\mathbb{R}^{h\times d}$为权值,$\boldsymbol{Z}\in\mathbb{R}^{h\times b}$为输出矩阵.一般情况下,若将全连接层中的权值变量变换成卷积核空间 4 阶张量形式,并将空间维度设为 1×1,那么全连接的矩阵计算可以转换成卷积计算.

另外,在卷积层与全连接层中还存在许多可选层及一些其他重要操作,如非线性变换(non-linearity)、池化(pooling)、正则化(normalization)等,对于网络性能的提升起到非常重要的作用.

1.3 经典的深度神经网络模型

在过去的几十年里,已经提出了很多经典的深度神经网络模型,每个模型拥有不同的网络结构,主要体现在层数、层类型、层形状(即卷积核大小、通道及滤波个数)、层内连接等.在本节中,我们回顾近几年在 ImageNet 竞赛上使用的冠军模型以及手写体字符识别网络模型 LeNet.这些预训练模型(pre-trained model)都已公开,模型的相关参数与总结如表 3 所示.第 1 行的 top-1 分类错误率($err_{\text{top-1}}$)和第 2 行的 top-5 分类错误率 $err_{\text{top-5}}$ 均为 ImageNet 竞赛上常用的 2 个分类评价指标,所有数值均采用简单的裁剪验证集(validation dataset)图像中间部分作为输入,得到分类的结果.本文将对 LeNet[14], AlexNet[3], VGGNet[12], GoogLeNet[22] 和 ResNet[13] 进行介绍.

1) LeNet[14].在 1998 年 LeNet 作为最早使用的卷积神经网络之一,被 LeCun 等人提出用于识别手写体字符.作为经典的版本——LeNet-5 包含了 2 个卷积层和 2 个全连接层.每 1 层的卷积计算都使用了 5×5 的空间维度,第 1 层使用了 20 个 3-D

卷积核,第 2 层使用 50 个卷积核. 每个卷积之后加入 Sigmoid 非线性变换作为激活函数,然后接着使用 2×2 的平均池化(average pooling)降采样特征图. 整个网络分类 1 张图像总共需要 6 万个参数、34.1 万浮点型计算次数. LeNet 是第 1 个被成功应用的卷积神经网络,它被广泛使用在 ATM 机上,应用于支票存款任务中识别手写体字符.

2) AlexNet[3]. 它是 ImageNet 竞赛上第 1 个利用深度学习获得冠军的卷积神经网络,同时该模型也为后续的深度学习的广泛应用奠定坚实的基础. 该模型由 5 个卷积层和 3 个全连接层组成. 第 1 个卷积层使用了 96 个维度为 11×11×3 的 3-D 卷积核,第 2 层使用了 256 个维度为 5×5×96 的 3-D 卷积核,后续的卷积层使用了空间维度为 3×3 的卷积核. AlexNet 引入了多种加速训练与提高模型分类准确率的技巧,如:①引入了 ReLU 非线性激活函数[20],代替原始的 Sigmoid 或 tahn 非线性激活[21]. ②在每个最大池化(max pooling)之前加入局部响应正则化(local response normalization, LRN),用于统一输出激活的数据分布. 在 AlexNet 中,LRN 应用于第 1,2 和 5 卷积层. ③利用 2 个 GPU 加速训练. ④通过扩充训练样本数和引入 Dropout 方法,防止模型训练过程过拟合. AlexNet 在 ImageNet 验证集上获得了 42.3% top-1 分类错误率和 19.1% top-5 分类错误率,且需要 0.61 亿个参数、7.29 亿浮点型计算次数处理尺寸大小为 227×227 的彩色图像.

Table 3 Summary of Popular DNNs[19]

表 3 常用深度神经网络的总结[19]

Metrics	LeNet-5	AlexNet	VGG-16	Inception-V1	ResNet-50
$err_{top\text{-}1}$/%	0.9	43.4	29.7	31.1	24.9
$err_{top\text{-}5}$/%		19.8	10.6	10.9	7.7
Input Size	28×28	227×227	224×224	224×224	224×224
# CONV Layers	2	5	13	57	53
# Depth of CONV Layers	2	5	13	21	49
Filter Sizes	5	3,5,11	3	1,3,5,7	1,3,7
# *Channels*	1,20	3-256	3-512	3-832	3-2048
# *Filters*	20,50	96-384	64-512	16-384	64-2048
Stride	1	1,4	1	1,2	1,2
# *Param*	2.6×10^3	2.8×10^6	1.47×10^7	6.0×10^6	2.35×10^7
# *FLOPs*	1.72×10^5	6.71×10^8	1.55×10^{10}	1.43×10^9	3.86×10^9
# *FC Layers*	2	3	3	1	1
Filter Sizes	1,4	1,6	1,7	1	1
# *Channels*	50,500	256-4096	512-4096	1024	2048
# *Filters*	10,500	1000-4096	1000-4096	1000	1000
# *Param*	5.8×10^4	5.86×10^7	1.23×10^8	1×10^6	2×10^6
# *FLOPs*	5.8×10^4	5.86×10^7	1.23×10^8	1×10^6	2×10^6
Total Param	6.0×10^4	6.14×10^7	1.38×10^8	7×10^6	2.55×10^7
Total FLOPs	2.3×10^6	7.29×10^8	1.56×10^{10}	1.43×10^9	3.9×10^9

3) VGGNet[12]. 它取得了 2014 年 ImageNet 竞赛的分类项目第 2 名、定位项目第 1 名. VGGNet 拥有强拓展性,体现在由很强的泛化能力、稳定的卷积特征和较好的表达能力. 具体而言,VGGNet 有 2 种形式,即 VGG-16 和 VGG-19. VGG-16 由 13 个卷积层和 3 个全连接层构成的 16 层网络. 不同于 AlexNet,VGG-16 利用多个小空间维度的卷积核(例如 3×3)代替原始大的卷积核(例如 5×5),获得相同的接受域(receptive fields). 在 VGG-16 中,所有的卷积层都采用了空间维度为 3×3 的卷积核. 为此,VGG-16 分类 1 张大小为 224×224 的彩色图像,需要 1.38 亿个参数、156 亿浮点型计算次数. 虽然 VGG-19 比 VGG-16 低 0.1% top-5 分类错误率,但需要 1.27 倍浮点型计算次数于 VGG-16.

4) GoogLeNet[22]. 它取得了 2014 年 ImageNet 竞赛的分类项目的冠军. Inception 拥有多种版本,包括 Inception-V1[22],Inception-V3[23]和 Inception-V4[24]等,其中 Inception-V1 也称为 GoogLeNet. 我们详细介绍 Inception-V1,它拥有 22 层深的网络结构,引入了核心组件——Inception module. 如图 5 所示,展示了 Inception module 主要组成部件,它由平行连接(parallel connection)构成. 在每个平行连接中,使用了不同空间维度的卷积核,即 1×1,3×3,5×5,并在每个卷积后紧跟 3×3 最大池化(其中一个 1×1 卷积核除外),所有卷积核输出结果全串在一起(concatenate)构成该模块的输出. GoogLeNet 利用 3 种不同的尺度卷积和最大池化,增加网络对不同尺度的适应性,同时可以让网络的深度和宽度高效率地扩充,提升准确率且不会过拟合. 22 层的 GoogLeNet 包括了 3 个卷积层、9 个 Inception 层(每个 Inception 层拥有 2 层深的卷积层)和 1 个全连接层. 由于采用了全局平均池化[23](global average pooling, GAP),并采用 1 个全连接层取代传统的 3 个全连接层,大大缩小了模型的存储空间. 对于分类一张 224×224 的彩色图像,GoogLeNet 需要 700 万个参数、14 亿浮点型计算次数.

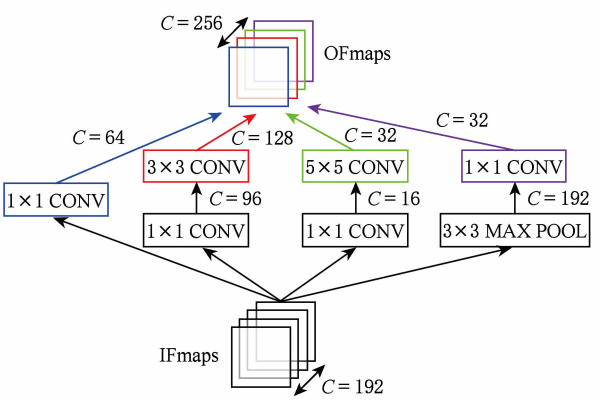


Fig. 5 Inception module in GoogLeNet

图 5 GoogLeNet 的 Inception 模块

5) ResNet[13]. 它是首次在 ImageNet 竞赛上超过人类所能达到的识别精度(即低于 5% top-5 分类错误率). ResNet 能够克服训练过程中梯度消失问题,即在反向传播(back-propagation)中梯度不断衰减,影响了网络底层的权值更新能力. 如图 6 所示,ResNet 的核心组件是残差块(residual block),每个残差块中引入了快捷模块(shortcut module),该模块包含了恒等连接或线性投影连接,并且学习残差映射($F(x)=H(x)-W_s x$),而不是直接学习权值层(weight layers)函数 $F(x)$. 同样地,ResNet 使用了 1×1 卷积核减少参数个数,并在每个卷积后加入批量正则化[22](batch normalization, BN). ResNet 有很多种结构表现形式,以 ResNet-50 为例,它包含了 1 个卷积层,紧接着 16 个残差块(每个残差块拥有 3 层深的卷积层),及 1 个全连接层. 对于分类一张 224×224 的彩色图像,ResNet-50 需要 0.25 亿个参数和 39 亿浮点型计算次数.

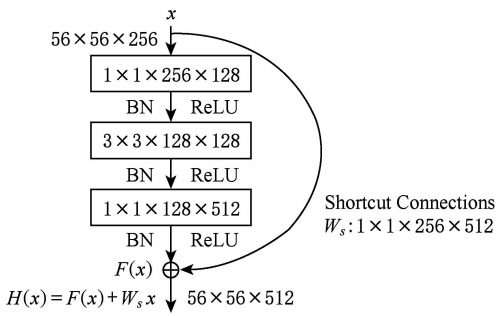


Fig. 6 Residual block in ResNet

图 6 ResNet 的残差块

如表 3 和图 2 所示,随着模型的深度和宽度的增加,深度网络取得更好的性能提升,但是卷积层的高度复杂的浮点计算及全连接层的高内存存储,严重阻碍深度模型应用于移动设备端. 因此,压缩与加速深度神经网络将愈发重要.

2 深度神经网络压缩与加速算法

在本节中,我们主要介绍了主流深度神经网络压缩与加速算法,以及相关算法的优缺点.

2.1 基于参数剪枝的深度神经网络压缩与加速

网络/参数剪枝是通过对已有的训练好的深度网络模型移除冗余的、信息量少的权值,从而减少网络模型的参数,进而加速模型的计算和压缩模型的存储空间. 不仅如此,通过剪枝网络,能防止模型过拟合. 以是否一次性删除整个节点或滤波为依据,参数剪枝工作可细分成非结构化剪枝和结构化剪枝. 非结构化剪枝考虑每个滤波的每个元素,删除滤波中元素为 0 的参数,而结构化剪枝直接考虑删除整个滤波结构化信息.

早在 20 世纪末,LeCun 等人[27]提出了最优化脑损失(optimal brain damage)算法,大大稀疏化多层网络的系数,同时保证模型预测精度依然处于零损失或最小量损失状态. 其实这种学习方式模仿了哺乳动物的生物学习过程,通过寻找最小激活的突触链接,然后在突触删减(synaptic pruning)过程中大大减少连接个数. 利用相似的思想,Hassibi 和 Stork[28]提出了最优化脑手术(optimal brain surgeon)剪枝策略,利用反向传播计算权值的二阶偏导信息

(hessian 矩阵),同时利用此矩阵构建每个权值的显著性得分,从而删除低显著性的权值.不同于最优化脑手术剪枝策略,Srinivas 等人[29]提出了不依赖于训练数据(data-free pruning)和反向传播,直接构建并排序权重的显著性矩阵,删除不显著冗余的节点.由于不依赖于训练数据及后向传播计算梯度信息,因此该网络剪枝过程较为快速.韩松等人[30-31]提出了一种基于低值连接的删除策略(low-weight connection pruning),该剪枝方法包括 3 个阶段,即训练连接、删除连接、重训练权值.第 1 阶段通过正常训练,学习重要的连接;第 2 阶段通过计算权值矩阵的范数,删除节点权重的范数值小于指定的阈值的连接,将原始的密集网络(dense network)变成稀疏网络;第 3 阶段通过重新训练稀疏网络,恢复网络的识别精度.以上剪枝方法通常引入非结构化的稀疏连接,在计算过程中会引起不规则的内存获取,相反会影响网络的计算效率.

近几年,基于结构化剪枝的深度网络的压缩方法陆续被提出,克服了非结构化稀疏连接导致的无法加速问题[27-31].其核心思想依靠滤波显著性准则(即鉴定最不重要的滤波的准则),从而直接删除显著性滤波,加速网络的计算.2016 年,Lebedev 等人[32]提出在传统的深度模型的损失函数中加入结构化的稀疏项,利用随机梯度下降法学习结构化稀疏的损失函数,并将小于给定阈值的滤波赋值为 0,从而测试阶段直接删除值为 0 的整个卷积滤波.温伟等人[33]通过对深度神经网络的滤波、通道、滤波形状、网络层数(filters, channels, filter shapes, layer depth)的正则化限制加入到损失函数中,利用结构化稀疏学习的方式,学习结构化的卷积滤波.Zhou 等人[34]将结构化稀疏的限制加入目标函数中,并利用前向后项分裂(forward-backward splitting)方法解决结构稀疏化限制的优化问题,并在训练过程中直接决定网络节点的个数与冗余的节点.另外,近年来,直接测量滤波的范数值直接判断滤波的显著性也相继被提出[35],例如:直接删除给定当前层最小 L1 范数的滤波,即移除相应的特征图(feature map),然后下一层的卷积滤波的通道数也相应地减少,最后通过重训练的方式提高删减后模型的识别精度.由于大量的 ReLU 非线性激活函数存在于主流的深度网络中,使得输出特征图高度稀疏化,Hu 等人[36]利用此特点,计算每个滤波所对应输出特征图的非零比例,作为判断滤波重要与否的标准.NVIDIA 公司 Molchanov 等人[37]提出一种基于全局搜索显著性滤波的策略,对需要删除的滤波用 0 值代替,并对目标函数进行泰勒公式展开(taylor expansion),判断使目标函数变换最小的滤波为显著滤波.通过卷积计算方式,可以建立当前层的滤波与下一层的卷积滤波的输入通道存在一一对应关系,利用此特点 Luo 等人[38]探索下一层卷积核的输入通道重要性,代替直接考虑当前层滤波,并建立一个有效的通道选择优化函数,从而删除冗余的通道以及相应的当前层的滤波.以上基于结构化剪枝的深度网络的压缩方法,通过删除卷积层的整个滤波,没有引入其他额外的数据类型存储,从而直接压缩网络的同时加速整个网络的计算.

Table 4 Comparison of Methods to Reduce Precision on AlexNet[19]

表 4 不同量化方法在 AlexNet 上的对比[19]

Reduce precision Method		Bitwidth		$err_{top\text{-}5}$ ↑/%
		Weights	Activations	
Dynamic Fixed Point	w/o fine-tuning[44]	8	10	0.4
	w/fine-tuning[45]	8	8	0.6
Reduce Weight	BC[46]	1	32(float)	19.2
	BWN[48]	1*	32(float)	0.8
	TWN[52]	2*	32(float)	3.7
	TTQ[53]	2*	32(float)	0.6
Reduce Weight and Activation	XNOR-Net[48]	1*	1*	11.0
	BNN[47]	1	1	29.8
	DoReFa-Net[49]	1*	2*	7.63
	QNN[50]	1	2*	6.5
	HWGQ-Net[51]	1*	2*	5.2

Notes: "*" denotes the method is not applied to first and/or last layers, and "↑" denotes increase.

参数剪枝的缺点在于,简单利用非结构化剪枝,无法加速稀疏化矩阵计算.虽然近年来,相关软件[39]与硬件[40]已被利用进行加速计算,但依靠软硬件的非结构化剪枝方案还无法在所有深度学习框架下使用,另外硬件的依赖性会使得模型的使用成本提高.结构化剪枝不依赖软硬件的支持,且能很好地嵌入目前主流的深度学习框架,但逐层固定的剪枝方式(layer-by-layer fixed manner)导致了网络压缩的低自适应能力、效率和效果.此外,上述的剪枝策略需要手动判断每层的敏感性,因此需要大量的精力分析及逐层微调(fine-tuning).

2.2 基于参数共享的深度神经网络压缩与加速

参数共享是通过设计一种映射将多个参数共享同一个数据.近年来,量化作为参数共享的最直接表现形式,得到广泛的应用.此外,Hash 函数和结构化线性映射也可作为参数共享的表现形式.

参数量化压缩与加速深度网络模型主要的核心思想是利用较低的位(bit)代替原始 32 b 浮点型的参数(也可记为全精度权值(full-precision weight)).龚云超等人[41]及 Wu 等人[42]利用向量量化的技术,在参数空间内对网络中的权值进行量化.近年来,利用低比特位的量化被提出用于加速与压缩深度神经网络.Gupta 等人[43]将全精度浮点型参数量化到 16 b 固定长度表示,并在训练过程中使用随机约束(stochastic rounding)技术,从而缩减网络存储和浮点计算次数.使用动态固定点(dynamic fixed point)量化,在量化 AlexNet 网络时,几乎可以做到无损压缩.例如,Ma 等人[44]将权值和激活分别量化到 8 b 和 10 b,且没有利用微调权值.随后 Gysel 等人[45]利用微调,将权值和激活全部量化到 8 b.

为了更大程度地缩减内存和浮点计算次数,对网络参数进行二值表示已被大量提出.其主要思想是在模型训练过程中直接学习二值权值或激活.BinaryConnect(BC)[46]通过直接量化权值为-1或 1,只需要加和减计算,减少了卷积计算,但因激活为全精度,无法大幅度加速网络计算.为此,通过同时量化权值和激活为-1和 1,Courbariaux 等人[47]提出了 BNN,将原始的卷积计算变成 Bitcount 和 XNOR,大幅度加速和压缩深度网络.但在压缩和加速深度网络时(如 AlexNet),分类精度大大降低.为了减少精度的丢失,Rastegari 等人[48]分别提出了 BWN 和 XNOR-Net 引入了尺度因子(scale factor),用于缩小量化误差,并保留第一层和最后一层的权值和输入为 32 b 的浮点型.同时,改变卷积和正则化的顺序,即先执行正则化、后卷积,减少了激活的动态幅度范围.伴随着这些改变,BWN 和 XNOR-Net 分别获得了相对于原始 AlexNet 0.8%和 11%的分类错误率增加.在近期的工作中[49-50],通过增加激活的位数(大于 1),并探索不同的低比特权值与激活的组合量化全精度权值和激活,提高量化后的网络在 ImageNet 数据集分类上的效果.但是在训练这些量化网络中,会出现梯度不匹配问题.Cai 等人[51]通过分析权值和激活的分布情况,设计一种新的半波高斯量化器(half-wave Gaussian quantizer)及其 BP 过程中不同的梯度近似,提出了 HWGQ-Net,有效地解决了训练过程中梯度不匹配问题.

由于权值近似分布于均值为 0 的高斯分布,即$W\sim N(0,\sigma^2)$,进一步考虑 0 作为量化后的值,可能减少量化误差.基于此思想,三元权值网络(ternary weight nets, TWN)[52]将全精度权值网络量化到三元网络(即$-w$,0 和w),其中w通过统计估计得到的量化值.通过改变对称的w,训练的三元量化[53](trained ternary quantization, TTQ)引入了不同的量化因子(即$-w_1$,0 和w_2),且通过训练得到该因子,在量化 AlexNet 时分类错误率只增加了 0.6%.如表 4 所示,列出了以上量化网络的性能比较及相应的量化比特数比较结果.

对于传统网络(如 AlexNet 和 VGG-16),全连接层的参数存储占整个网络模型的 95%以上,所以探索全连接层参数冗余性将变得异常重要.利用 Hash 函数和结构化线性映射相继提出,可用于实现全连接层的参数共享,大大减低模型的内存开销.Chen 等人[54]提出了 HashNet 模型,利用 2 个低耗的 Hash 函数对不同的网络参数映射到相同 Hash 桶中,实现参数共享.Cheng 等人[55]提出基于一种简单有效的循环投影方法,即利用存储量极小的循环矩阵代替原始矩阵,同时使用快速傅里叶变换(fast Fourier transform, FFT)加速矩阵的乘积计算.另外,Yang 等人[56]引入了新的 Adaptive Fastfood 变换,重新定义了全连接层的矩阵与向量之间的乘积计算,减少了参数量和计算量.

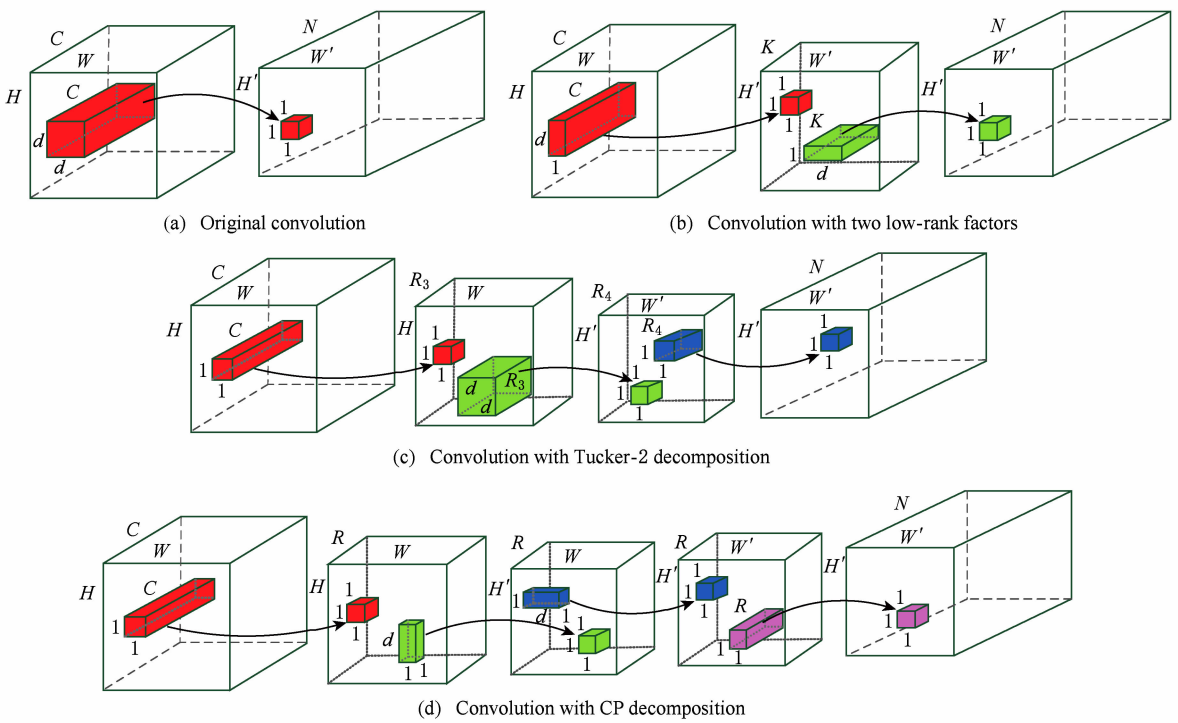
量化权值,特别是二值化网络,存在以下缺点:1)对于压缩与加速大的深度网络模型(如 GoogLeNet 和 ResNet),存在分类精度丢失严重现象;2)现有方法只是采用简单地考虑矩阵近似,忽略了二值化机制对于整个网络训练与精度损失的影响;3)对于训练大型二值网络,缺乏收敛性的理论验证,特别是对于同时量化权值和激活的二值化网络

(如 XNOR-Net,BNN 等). 对于限制于全连接层的参数共享方法,如何泛化到卷积层成为一个难题. 此外,结构化矩阵限制可能引起模型偏差,造成精度的丢失.

2.3 基于低秩分解的深度神经网络压缩与加速

基于低秩分解的深度神经网络压缩与加速的核心思想是利用矩阵或张量分解技术估计并分解深度模型中的原始卷积核. 卷积计算是整个卷积神经网络中计算复杂度最高的计算操作,通过分解 4D 卷积核张量,可以有效地减少模型内部的冗余性. 此外对于 2D 的全连接层矩阵参数,同样可以利用低秩分解技术进行处理. 但由于卷积层与全连接层的分解方式不同,本文分别从卷积层和全连接层 2 个不同角度回顾与分析低秩分解技术在深度神经网络中的应用.

在 2013 年,Denil 等人[57]从理论上利用低秩分解的技术并分析了深度神经网络存在大量的冗余信息,开创了基于低秩分解的深度网络模型压缩与加速的新思路. 如图 7 所示,展示了主流的张量分解后卷积计算. Jaderberg 等人[58]利用张量的低秩分解技术,将原始的网络参数分解成 2 个小的卷积核. 利用相同简单的策略,Denton 等人[59]先寻找对卷积层参数的低秩近似,然后通过微调的方式恢复模型的识别精度. 此外,利用经典的 CP 分解[60],将原始的张量参数分解成 3 个秩为 1 的小矩阵. 相似地,利用 Tucker 分解[61],将原始的张量分解成 3 个小的张量的乘积. Tai 等人[62]提出了新的低秩分解张量算法,同时也提出了引入批量正则化,从头开始训练有低秩限制的卷积神经网络. Ioannou 等人[63]利用卷积核的低秩表示,代替分解预训练的卷积核参数,并设计了一种有效的权值初始化方法,从头开始训练计算有效的卷积神经网络. 同样地,代替直接分解预训练的模型参数,温伟等人[64]从训练的角度探讨如何更有效地聚集更多参数于低秩空间上,提出了新的强力正则化项(force regularization),迫使更多的卷积核分布于更为低秩空间中. 以上低秩分解卷积核的方法,虽然减少了卷积核的冗余性,即考虑了卷积神经网络内部结构的冗余性,但全盘接受了视觉输入的全部,极大地影响了模型加速比. 为此,林绍辉等人[65]提出了 ESPACE 卷积计算加速框架,考虑了视觉输入的冗余性,即从输入计算空间和通道冗余性 2 方面移除低判别性和显著性的信息.



K, (R_3, R_4) and R are the corresponding rank of low-rank decomposition, Tucker-2 decomposition and CP decomposition respectively.

Fig. 7 Convolution with several low-rank factors

图 7 拥有若干个低秩因子的卷积计算

对于全连接层特定的 2D 矩阵形式,虽然可以通过转变 2D 矩阵计算为 1×1 的卷积计算,从而利用上述低秩分解技术进行应用,但对于特定的全连接层也存在相关低秩分解方法. Denil 等人[57]利用

了低秩分解方法减少了深度神经网络中的动态参数个数.林绍辉等人[66]分析了直接对层内参数低秩分解压缩无法获得高精度分类效果的缺点,提出考虑层间的各种非线性关系,参数层间的联合优化,代替单层的优化,构建全局误差最小化优化方案.

基于低秩分解的深度网络模型压缩算法,在特定场景下取得良好的效果,但增加了模型原有的层数,极易在训练过程中造成梯度消失的问题,从而影响压缩后网络的精度恢复.另外,逐层低秩分解优化参数,无法从全局进行压缩,延长了离线分解时间开销.

2.4 基于紧性卷积核的深度神经网络压缩与加速

对深度网络模型的卷积核使用紧性的滤波直接替代,将有效地压缩深度网络.基于该思想,直接将原始较大的滤波大小(如 5×5,3×3)分解成 2 个 1×1 卷积滤波,大大加速了网络的计算同时获得了较高的目标识别性能.

2016 年,SqueezeNet[67]的提出是将原始的卷积结构替换成为 Fire Module,即包括 Squeeze 层和 Expand 层.在 Squeeze 层将 3×3 的卷积滤波替换成 1×1 的卷积滤波,并在 Expand 层中加入 1×1 和 3×3 的卷积滤波,同时减少 3×3 的卷积滤波个数,减少池化(pooling),从而简化网络复杂度,降低卷积网络模型参数的数量,同时也达到 AlexNet 识别精度.最终的模型参数降低了 50 倍,大大压缩了深度网络模型.另外,Google 公司 Howard 等人[68]提出了 MobileNets,利用计算和存储更小的深度分割卷积(depthwise separable convolution)替代原始的标准卷积计算.Zhang 等人[69]提出 ShuffleNet,利用组卷积(group convolution)和通道重排(channel shuffle)两个操作设计卷积神经网络模型,从而减少模型使用的参数.一般情况下,使用组卷积会导致信息流通不当,通过通道重排改变通道的序列,得到与原始卷积相似的计算结果.Chollet[70]提出 Xception 网络结构,在原始 Inception-V3 的基础上引入深度分割卷积计算,不同于原始 MobileNet 中的深度分割卷积,Xception 先进行 1×1 的点卷积计算(point convolution),然后再逐通道卷积,提高了模型的计算效率.在同参数量情况下,分类效果优于 Inception-V3.

基于紧性卷积核的深度神经网络压缩与加速采用了特定的卷积核的设计或新卷积计算方式,大大压缩神经网络模型或加速了卷积计算.但压缩与加速方法的扩展性和结合性较弱,即较难在紧性卷积核的深度神经网络中利用不同压缩或加速技术进一步提高模型使用效率.另外,跟原始模型相比,基于紧性卷积核设计的深度神经网络得到的特征普适性及泛化性较弱.

2.5 基于知识蒸馏的深度神经网络压缩与加速

知识蒸馏(KD)的基本思想是通过软 Softmax 变换学习教师输出的类别分布,并将大型教师模型(teacher model)的知识精炼为较小的模型.如图 8 所示,展示了简单的知识蒸馏的流程.2006 年,Buciluǎ 等人[71]首先提出利用知识迁移(knowledge transfer, KT)来压缩模型.他们通过集成强分类器标注的伪数据(pseudo-data)训练了一个压缩模型,并重现了原大型网络的输出结果,然而他们的工作仅限于浅层网络.近年来,知识蒸馏[72]提出了可以将深度和宽度的网络压缩为浅层模型,该压缩模型模仿了复杂模型所能实现的功能.

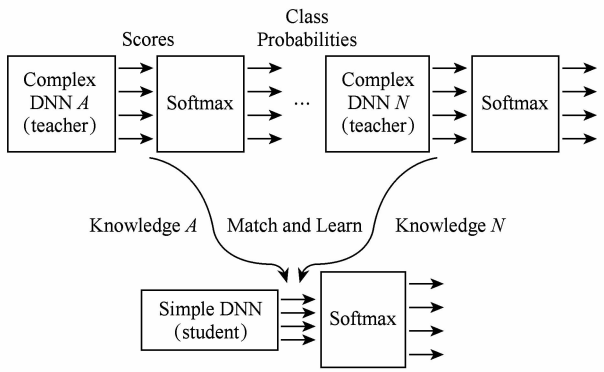


Fig. 8 Knowledge distillation

图 8 知识蒸馏

Hinton 等人[73]提出了知识蒸馏的压缩框架,通过软化教师网络输出指导和惩罚学生网络(student network).该框架将集成的深度网络压缩成为相同深度的学生网络.为此,利用教师软输出的结果作为标签,训练压缩学生网络.Romero 等人[74]提出了基于知识蒸馏的 FitNet,通过训练窄且深网络(学生网络),压缩宽且浅网络(教师网络).

近几年,知识蒸馏也得到了改进和拓展,例如:Balan 等人[75]通过在线训练的方式学习带有参数的学生网络近似蒙特卡洛(Monte Carlo)教师网络.不同于原来的方法,该方法使用软标签作为教师网络知识的表达,代替原来的教师网络的软输出.Luo 等人[76]利用高层隐含层神经元的输出作为知识,它比使用标签概率作为知识能保留更多的知识.Zagoruyko 等人[77]提出了注意力迁移(attention transfer, AT),通过迁移注意力图(attention maps),松弛了 FitNet 的假设条件.

虽然基于知识蒸馏的深度神经网络压缩与加速方法能使深层模型细小化,同时大大减少了计算开销,但是依然存在2个缺点:1)只能用于具有Softmax损失函数分类任务,这阻碍了其应用;2)模型的假设较为严格,以至于其性能可能比不上其他压缩与加速方法.

2.6 其他类型的深度神经网络压缩与加速

在1.3节介绍GoogLeNet时,为了减少全连接层的参数个数,全局均匀池化代替传统的3层全连接层,减少了全连接层的参数.在近几年提出的最新的网络结构中,全局均匀池化方法广泛其中,例如Network in Network(NIN),GoogLeNet,ResNet,ResNeXt[78]等,在很多基准(benchmark)任务中取得了最优的(state-of-the-art)性能.但该类型结构在ImageNet数据集上学习到的特征很难直接迁移到其他任务上.为了解决此问题,Szegedy等人[22]在原始结构的基础上加入了线性层.

此外,基于卷积的快速傅里叶变化[79]和使用Winograd算法[80]的快速卷积计算,大大减少了卷积计算的开销.Zhai等人[81]提出了随机空间采样池化(stochastic spatial sampling pooling),用于加速原始网络中的池化操作.但是这些工作仅仅为了加速深度网络计算,无法达到压缩网络的目的.

3 数据集与已有方法性能

3.1 数据集

在深度神经网络压缩与加速中常见并具有代表性的数据集主要包括MNIST[14],CIFAR-10/100[82],ImageNet[18],如表5所示,罗列了这些数据集的基本统计信息.

Table 5 Comparison of the Widely Used Dataset for DNN Compression and Acceleration

表5 常见深度神经网络压缩与加速数据集统计信息

Dataset	#*Class*	#*Train*	#*Validation*	#*Test*	*Size*
MNIST	10	6×10^4		1×10^4	28×28
CIFAR-10	10	5×10^4		1×10^4	32×32
CIFAR-100	10	5×10^4		1×10^4	32×32
ImageNet	1 000	1.2×10^6	5×10^4	1×10^5	

MNIST是用于手写字符分类被广泛使用的数据集.在1998年,该数据集被公开作为字符识别算法评测的公共数据集.在该数据集中,包含了来自10个类别(即手写体数字0～9),像素为28×28的手写体字符灰度图,总计有6万张训练图像和1万张测试图像.LeNet-5作为经典模型在MNIST上分类错误率达到0.9%.MNIST已作为简单且公平的数据集用于评测深度神经网络压缩与加速性能.

CIFAR是用于分类小图像的数据集,它是8千万张Tiny Image数据集的子集.在2009年,该数据集被公开作为分类小型彩色图像算法评测的公共数据集.CIFAR数据集有2个版本,分别为CIFAR-10和CIFAR-100,包含了像素均为32×32的自然彩色图像.在CIFAR-10数据集中,包含了来自10个互不交叉的类别,总计有5万张训练图像(每类5千张)和1万张测试图像(每类1千张).在CIFAR-100数据集中,包括了100个类别不同的图像,其中5万张训练图像(每类500张)和1万张测试图像(每类100张).在不引入任何数据扩展方法时,经典模型NIN在CIFAR-10和CIFAR-100分类错误率达到10.41%和35.68%.

ImageNet是一个大尺度图像数据集.2010年首次被提出,并在2012年得到稳定使用.该数据集包含1 000个类别彩色图像,且一般先缩放至像素大小为256×256.不同于以上2种数据集,该数据集的类别标签可用字网络(WordNet)表示,即由主类别标签词表示并包含相同目标的近义词,相当于词汇的分层结构.该数据集总计约有0.13亿张训练图像(每个类别数量在732～1 300之间),10万张测试图像(每类100张)和5万张验证图像(每类50张).由于测试数据集标签未公开,一般情况下,测试该数据集性能时通过验证集上性能作为测试依据.另外,top-1分类错误率和top-5分类错误率是判断ImageNet分类准确率指标.top-1指的是得分最高的类别刚好是标签类别时,分类正确;top-5指的是得分前5的类别中包含正确标签时,分类正确.AlexNet,VGG-16,ResNet-50作为ImageNet数据集上经典的网络模型,分别获得了42.24% top-1分类错误率和19.11% top-5分类错误率、31.66% top-1分类错误率和11.55% top-5分类错误率以及24.64% top-1分类错误率和7.76% top-5分类错误率.

综上所述,MNIST是较为简单的小型数据集,而ImageNet是一个大尺度类别多样的复杂数据集,充满着挑战.另外,对于同一个网络,在不同数据集下表现出不一样的性能,因此判断网络性能时一般需要考虑所采用的数据集.

3.2 评价准则

率失真(rate-distortion)作为评价深度神经网

络压缩与加速性能标准,既考虑了模型的压缩与加速比,同时需要计算压缩与加速后的模型分类误差率.为了计算模型的压缩与加速比,我们假设 λ 和 λ^* 分别为原始模型 M 和压缩后模型 M^* 所有参数所占的内存开销,那么模型的压缩比 C_r 可计算为

$$C_r(M,M^*)=\frac{\lambda}{\lambda^*}. \tag{3}$$

相似地,假设 γ 和 γ^* 分别为原始模型 M 和压缩后模型 M^* 推测(inference)整个网络的时间,那么模型的加速比 S_r 可定义为

$$S_r(M,M^*)=\frac{\gamma}{\gamma^*}. \tag{4}$$

关于分类误差率需要根据不同的数据集,如上述所述的数据集中,MNIST 和 CIFAR 只存在 top-1 分类错误率,而 ImageNet 有 top-1 和 top-5 两种分类错误率.根据数据集不同,选择相应模型性能评价标准.一般情况下,好的深度神经网络压缩与加速方法表现出:压缩与加速后的模型具有相似于原始模型的分类错误率,同时伴随着少量的模型参数和计算复杂度.因此模型的错误率和压缩比(或加速比)需要统一考虑.

目前深度学习框架层出不穷,包括 Caffe[83],Tensorflow[84],Torch 等.为了公平评测各压缩模型的实际速度,对于深度神经网络压缩与加速领域,采用 Caffe 作为主流的深度学习框架.

3.3 代表性的深度神经网络压缩与加速方法性能

目前已有工作的实验所用数据集一般是 3 个数据集:MNIST,Cifar,ImageNet,使用模型包括 LeNet-5,AlexNet,VGG-16,ResNet 等.下面我们比较代表性深度神经网络压缩与加速方法在不同模型上的压缩与加速效果.

如表 6 所示,列出了包括剪枝与参数共享在内的压缩与加速方法在 LeNet-5 上的比较结果.#*FLOPs* 表示浮点型计算次数个数,#*Param* 表示模型中参数的数量,$err_{\text{top-1}}\uparrow$ 表示相比较于原始模型,压缩后的模型 top-1 分类错误率增加值.从整体上看,在该模型上压缩与加速取得了较好的结果.从压缩与加速 2 方面比较,非结构化剪枝[31]在压缩模型参数上能取得 12×更好的结果,而结构化剪枝 SSL[33]在加速模型上取得 3.62×的结果,基本保证模型不丢失精度.

Table 6 Results of Different DNN Compression and Acceleration on LeNet-5

表 6 不同压缩与加速方法在 LeNet-5 上的结果

Method	#*FLOPs*	#*Param*(C_r)	S_r	$err_{\text{top-1}}\uparrow$/%
LeNet-5	2.3×10^6	0.43×10^6(1×)	1×	0
Pruning[31]		3.44×10^4(12×)		−0.1
SSL[33]	1.62×10^5	4.5×10^5(9.5×)	3.62×	0.05
Circulant[55]		7.4×10^4(5.8×)	1.43×	0.03
AFT[56]		5.2×10^4(8.3×)		−0.16

Note: ↑ denotes increase.

如表 7 和表 4 所示,主要比较了不同压缩与加速方法在 AlexNet 上的结果.如表 7 所示,主要比较了不同低秩分解技术在 AlexNet 上加速效果(除 SSL[33]外).从比较结果可以看出,LRA[64]不仅在准确率和加速比高于传统的 CPD[60]和 TD[61],而且在加速比(4.05×vs. 3.13×)上也高于 SSL,top-5 分类错误率增加几乎一样(1.71% vs. 1.66%).如表 4 所示,主要列出了关于量化网络在 AlexNet 上的性能比较.原始的权值和激活的保存形式为 32 b 浮点型,通过用低比特位数量化网络,减少模型存储和计算开销.增加比特位的位数,模型的性能得到提升,量化到 8 b 几乎无损压缩模型,如利用动态固定点量化[44-45].若只对模型的权值进行量化,保持激活位数不变,BWN[48],TWN[52],TTQ[53]能够较好地控制模型误差增加(除了 BC[46]外),但加速效果有限.通过进一步量化激活,特别是量化激活为 2 b 时,HWGQ-Net[51]取得了 top-5 分类错误率增加仅为 5.2%.

Table 7 Comparison of Different CNN Acceleration Methods on AlexNet[64]

表 7 不同卷积神经网络加速方法在 AlexNet 上的对比

Method	$err_{\text{top-5}}\uparrow$/%	*Conv*1	*Conv*2	*Conv*3	*Conv*4	*Conv*5	Average Speedup
AlexNet	0	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×
CPD[60]	1.00		4.00×				1.27×
TD[61]	1.70	1.48×	2.30×	3.84×	3.53×	3.13×	2.52×
SSL[33]	−0.39	1.00×	1.27×	1.64×	1.68×	1.32×	1.35×
	1.66	1.05×	3.37×	6.27×	9.73×	4.93×	3.13×
LRA[64]	0.17	2.61×	6.06×	2.48×	2.20×	1.58×	2.69×
	1.71	2.65×	6.22×	4.81×	4.00×	2.92×	4.05×

Note: ↑ denotes increase.

在压缩与加速 VGG-16 上,如表 8 所示,展示了部分比较有代表性的剪枝算法的评测结果. 为了进一步压缩 VGG-16,借鉴 NIN[25] 全局均匀池化,代替原始模型中的 3 层全连接层. 我们把压缩后的模型记为"X-GAP",指的是利用"X"方法剪枝完所有的卷积层后,加入 GAP 进行微调整个压缩后的网络. 特别地,VGG-GAP 不删除任何卷积层的滤波,反而取得了较高的分类错误率,即 top-1 分类错误率和 top-5 分类错误率分别为 37.97%和 15.65%. 另外,ThiNet[38] 较其他方法,取得了较低的分类错误率增加(即 top-1 分类错误率和 top-5 分类错误率分别增加为 1.00%和 0.52%),但加速和压缩比相对较低于 L1[35],APoZ[36]. 通过进一步增加剪枝卷积核的数量(记为"ThiNet-T"),取得 106.5×压缩比和 4.35×加速比,但压缩后模型的 top-1 分类错误率和 top-5 分类错误率分别增加了 9%和 6.47%.

Table 8 Comparison of Parameter Pruning Methods for Compressing and Accelerating VGG-16

表 8 参数剪枝方法在压缩与加速 VGG-16 上的对比

Method	$10^{-9}\times$ #*FLOPs*	$10^{-6}\times$ #*Param*	S_r (GPU)	$err_{\text{top-1}}\uparrow$ /%	$err_{\text{top-5}}\uparrow$ /%
VGG-16	15.5	138.4	1×	0	0
VGG-GAP	15.4	15.2	≈1×	1.50	0.56
L1-GAP[35]	4.4	9.2	2.5×	4.62	3.10
APoZ-GAP[36]	4.4	9.2	2.5×	3.72	2.65
TE[37]	4.2	135.7	2.7×		3.94
ThiNet[38]	4.9	9.5	2.3×	1.00	0.52
ThiNet-T[38]	2.0	1.3	4.35×	9.00	6.47

Note: ↑ denotes increase.

如表 9 所示,展示了主流的二值量化方法在 ImageNet 数据集上压缩与加速 ResNet-18 的比较结果. 全精度的 ResNet-18 能达到 30.7% top-1 分类错误率及10.8%top-5分类错误率. 虽然BWN[48] 和 DeReFa-Net[49] 取得较好的分类性能,但它们分别使用了全精度的激活和 4 b 的激活. 若同时二值化权值和激活,计算卷积时,只需要 bitcount 和 XNOR 计算,大大提高卷积计算效率,但降低了模型的分类效果,例如 BNN[47] 和 XNOR-Net[48] 分别增加了 27.1%和 18.1%的 top-1 分类错误率.

Table 9 Results of Binary Network on ImageNet for Compressing ResNet-18

表 9 二值网络在 ImageNet 上压缩 ResNet-18 的结果

Model	W-bit	A-bit	$err_{\text{top-1}}\uparrow$ /%	$err_{\text{top-5}}\uparrow$ /%
ResNet-18	32	32	0	0
BWN[48]	1	32	8.5	6.2
DoReFa-Net[49]	1	4	10.1	7.7
XNOR-Net[48]	1	1	18.1	16.0
BNN[47]	1	1	27.1	22.1

Note: ↑ denotes increase.

如表 10 所示,比较 MobileNet[68] 和 ShuffleNet[69] 在 ImageNet 上性能比较结果. *a* MobileNet-224 指的是在基准模型框架下(即 1 MobileNet-224),以 *a* 倍缩放基准结构的每层卷积核滤波个数;ShuffleNet *b*×,*g*=*c* 指的是在组的个数为 *c* 的情况下,以 *b* 倍缩放基准结构(即 ShuffleNet 1×)的每层卷积核滤波个数. 通过不同层次的浮点型计算复杂度对比结果,可以明显得出 ShuffleNet 的性能优于 MobileNet. 特别值得注意的是,在小的浮点型计算复杂度情况下(如 0.4 亿浮点型计算次数),ShuffleNet 取得了更好的分类结果,即高于相同复杂度等级的 MobileNet 6.7%分类错误率.

Table 10 Comparison of MobileNet and ShuffleNet on ImageNet Classification[69]

表 10 MobileNet 和 ShuffleNet 在 ImageNet 分类上比较[69]

Model	$10^{-6}\times$ #*FLOPs*	$err_{\text{top-1}}$/%	$err_{\text{top-1}}\downarrow$/%
1.0 MobileNet-224	569	29.4	0
ShuffleNet 2×,*g*=3	524	29.1	0.3
0.75 MobileNet-224	325	31.6	0
ShuffleNet 1.5×,*g*=3	292	31.0	0.6
0.5 MobileNet-224	149	36.3	0
ShuffleNet 1×,*g*=3	140	34.1	2.2
0.25 MobileNet-224	41	49.4	0
ShuffleNet 0.5×,*g*=8	40	42.7	6.7
ShuffleNet 0.5×,*g*=3	40	45.2	4.2

Note: ↓ denotes decrease.

4 讨论:压缩与加速方法选择

在第 2 节中,我们回顾与总结深度神经网络压缩与加速算法,但如何选择不同的压缩与加速算法成为一个难题. 为此,在本节中,我们将进行详细讨论选择深度模型压缩与加速策略.

事实上,上述介绍的 5 种主流的深度神经网络压缩与加速方法各有优缺点,也没有固定和可量化的准则来判断哪种方法是最优的. 所以压缩与加速方法的选择依赖于不同的任务和需要. 总结出压缩与加速方法选择的 6 条意见:

1) 对于在线计算内存存储有限的应用场景或设备,可以选择参数共享和参数剪枝方法,特别是二值量化权值和激活、结构化剪枝.其他方法虽然能够有效的压缩模型中的权值参数,但无法减小计算中隐藏的内存大小(如特征图).

2) 如果在应用中用到的紧性模型需要利用预训练模型,那么参数剪枝、参数共享以及低秩分解将成为首要考虑的方法.相反地,若不需要借助预训练模型,则可以考虑紧性滤波设计及知识蒸馏方法.

3) 若需要一次性端对端训练得到压缩与加速后模型,可以利用基于紧性滤波设计的深度神经网络压缩与加速方法.

4) 一般情况下,参数剪枝,特别是非结构化剪枝,能大大压缩模型大小,且不容易丢失分类精度.对于需要稳定的模型分类的应用,非结构化剪枝成为首要选择.

5) 若采用的数据集较小时,可以考虑知识蒸馏方法.对于小样本的数据集,学生网络能够很好地迁移教师模型的知识,提高学生网络的判别性.

6) 主流的5个深度神经网络压缩与加速算法相互之间是正交的,可以结合不同技术进行进一步的压缩与加速.如:韩松等人[30]结合了参数剪枝和参数共享;温伟等人[64]以及Alvarez等人[85]结合了参数剪枝和低秩分解.此外对于特定的应用场景,如目标检测,可以对卷积层和全连接层使用不同的压缩与加速技术分别处理.

5 未来发展趋势

就目前研究成果而言,深度神经网络压缩与加速还处于早期阶段,压缩与加速方法本身性能还有待提高,合理压缩与加速评价标准还需完善,为深入地研究提供帮助.

合理的性能评价标准的建立.虽然率失真率能同时考虑压缩或加速、分类性能分布情况,但不同压缩与加速方法无法统一到同一指标下,即同一压缩与加速率下判断分类性能的优劣,或同一分类性能下判断压缩与加速比.在后续的评测标准中,特别对不同压缩与加速方法的合理评价将得到补充和完善.

压缩与加速的模型多样性得到推广.目前深度神经网络压缩与加速的模型多使用卷积神经网络,这肯定了卷积神经网络强大的特征表示能力及应用覆盖面.除卷积神经网络外,还存在大量其他不同结构的网络,如递归神经网络(recurrent neural network, RNN)、长短期记忆网络(long short-term memory, LSTM),广泛应用于人工智能领域.在未来一段时间内,对于递归神经网络、长短期记忆网络等不同于卷积神经网络结构的网络,是否可以使用上述介绍的压缩与加速方法进行处理,这将有待于研究.

更多深度神经网络压缩与加速技术嵌入终端设备,实现实际应用落地.随着深度神经网络压缩与加速方法的快速推进与发展,对于分类任务的压缩与加速方面已取得较大进步,但对于其他视觉任务较少涉及.设计基于视觉任务(如目标检测、目标跟踪、图像分割等)为一体的深度神经网络压缩与加速,将成为深度神经网络压缩与加速方法真正植入智能终端设备,实现实际应用落地的研究热点.

6 总 结

本文首先描述了深度神经网络压缩与加速技术的研究背景;然后对深度神经网络压缩与加速相关代表方法进行详细梳理与总结;其次回顾了目前主流压缩与加速算法所使用的数据集、评价准则及性能评估;最后讨论了相关深度神经网络压缩与加速算法的选择问题,并分析了未来发展趋势.随着深度神经网络压缩研究的不断深入,希望本文能给当前及未来的研究提供一些帮助.

参 考 文 献

[1] LeCun Y, Bengio Y, Hinton G. Deep learning [J]. Nature, 2015, 521(7553): 436-444

[2] Deng Li, Li Jinyu, Huang Juiting, et al. Recent advances in deep learning for speech research at Microsoft [C] //Proc of ICASSP 2013. Piscataway, NJ: IEEE, 2013: 8604-8608

[3] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks [C] // Proc of NIPS 2012. Cambridge, MA: MIT Press, 2012: 1097-1105

[4] Zhou Ye, Zhang Junping. Multi-scale deep learning for product image search [J]. Journal of Computer Research and Development, 2017, 54(8): 1824-1832 (in Chinese)
(周晔,张军平. 基于多尺度深度学习的商品图像检索 [J]. 计算机研究与发展, 2017, 54(8): 1824--1832)

[5] Liang Bin, Liu Quan, Xu Jin, et al. Aspect-based sentiment analysis based on multi-attention CNN [J]. Journal of Computer Research and Development, 2017, 54(8): 1724-1735 (in Chinese)

(梁斌, 刘全, 徐进, 等. 基于多注意力卷积神经网络的特定目标情感分析[J]. 计算机研究与发展, 2017, 54(8): 1724-1735)

[6] Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch [J]. Journal of Machine Learning Research, 2011, 12(8): 2493-2537

[7] Chen Chenyi, Seff A, Kornhauser A, et al. Deepdriving: Learning affordance for direct perception in autonomous driving [C] //Proc of ICCV 2015. Piscataway, NJ: IEEE, 2015: 2722-2730

[8] Esteva A, Kuprel B, Novoa R A, et al. Dermatologist-level classification of skin cancer with deep neural networks [J]. Nature, 2017, 542(7639): 115-118

[9] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search [J]. Nature, 2016, 529(7587): 484-489

[10] Lowe D G. Object recognition from local scale-invariant features [C] //Proc of ICCV 1999. Piscataway, NJ: IEEE, 1999: 1150-1157

[11] Dalal N, Triggs B. Histograms of oriented gradients for human detection [C] //Proc of CVPR 2005. Piscataway, NJ: IEEE, 2005: 886-893

[12] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition [J]. arXiv preprint, arXiv:1409.1556, 2014

[13] He Kaiming, Zhang Xiangyu, Ren Shaoqing, et al. Deep residual learning for image recognition [C] //Proc of CVPR 2016. Piscataway, NJ: IEEE, 2016: 770-778

[14] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324

[15] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks [J]. Science, 2006, 313(5786): 504-507

[16] Goodfellow I, Bengio Y, Courville A. Deep Learning [M]. Cambridge, MA: MIT Press, 2016

[17] Russakovsky O, Deng Jia, Su Hao, et al. ImageNet large scale visual recognition challenge [J]. International Journal of Computer Vision, 2015, 115(3): 211-252

[18] Deng Jia, Dong Wei, Socher R, et al. ImageNet: A large-scale hierarchical image database [C] //Proc of CVPR 2009. Piscataway, NJ: IEEE, 2009: 248-255

[19] Sze V, Chen Yuhsin, Yang Tienju, et al. Efficient processing of deep neural networks: A tutorial and survey [J]. Proceedings of the IEEE, 2017, 105(12): 2295-2329

[20] Nair V, Hinton G E. Rectified linear units improve restricted boltzmann machines [C] //Proc of ICML 2010. New York: ACM, 2010: 807-814

[21] LeCun Y, Bottou L, Orr G B, et al. Efficient Backprop [M]. Berlin: Springer, 1998

[22] Szegedy C, Liu Wei, Jia Yangqing, et al. Going deeper with convolutions [C] //Proc of CVPR 2015. Piscataway, NJ: IEEE, 2015: 1-9

[23] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision [C] //Proc of CVPR 2016. Piscataway, NJ: IEEE, 2016: 2818-2826

[24] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resNet and the impact of residual connections on learning [C] //Proc of AAAI 2017. Menlo Park, CA: AAAI, 2017: 4278-4284

[25] Lin Min, Chen Qiang, Yan Shuicheng. Network in network [J]. arXiv preprint, arXiv:1312.4400, 2013

[26] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift [C] //Proc of ICML 2015. New York: ACM, 2015: 448-456

[27] LeCun Y, Denker J, Solla S, et al. Optimal brain damage [C] //Proc of NIPS 1989. Cambridge, MA: MIT Press, 1989: 598-605

[28] Hassibi B, Stork D G. Second order derivatives for network pruning: Optimal brain surgeon [C] //Proc of NIPS 1993. Cambridge, MA: MIT Press, 1993: 164-171

[29] Srinivas S, Babu R V. Data-free parameter pruning for deep neural networks [J]. arXiv preprint, arXiv: 1507.06149, 2015

[30] Han Song, Mao Huizi, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding [J]. arXiv preprint, arXiv: 1510.00149, 2015

[31] Han Song, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network [C] //Proc of NIPS 2015. Cambridge, MA: MIT Press, 2015: 1135-1143

[32] Lebedev V, Lempitsky V. Fast convnets using group-wise brain damage [C] //Proc of CVPR 2016. Piscataway, NJ: IEEE, 2016: 2554-2564

[33] Wen Wei, Wu Chunpeng, Wang Yandan, et al. Learning structured sparsity in deep neural networks [C] //Proc of NIPS 2016. Cambridge, MA: MIT Press, 2016: 2074-2082

[34] Zhou Hao, Alvarez J M, Porikli F. Less is more: Towards compact CNNs [C] //Proc of ECCV 2016. Berlin: Springer, 2016: 662-677

[35] Li Hao, Kadav A, Durdanovic I, et al. Pruning filters for efficient ConvNets [J]. arXiv preprint, arXiv:1608.08710, 2016

[36] Hu Hengyuan, Peng Rui, Tai Y W, et al. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures [J]. arXiv preprint, arXiv:1607.03250, 2016

[37] Molchanov P, Tyree S, Karras T, et al. Pruning convolutional neural networks for resource efficient transfer learning [J]. arXiv preprint, arXiv:1611.06440, 2016

[38] Luo Jianhao, Wu Jianxin, Lin Weiyao. Thinet: A filter level pruning method for deep neural network compression [C] //Proc of ICCV 2017. Piscataway, NJ: IEEE, 2017: 5058-5066

[39] Liu Baoyuan, Wang Min, Foroosh H, et al. Sparse convolutional neural networks [C] //Proc of CVPR 2015. Piscataway, NJ: IEEE, 2015: 806-814

[40] Han Song, Liu Xingyu, Mao Huizi, et al. EIE: Efficient inference engine on compressed deep neural network [C] // Proc of the 43rd Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 243-254

[41] Gong Yunchao, Liu Liu, Yang Ming, et al. Compressing deep convolutional networks using vector quantization [J]. arXiv preprint, arXiv:1412.6115, 2014

[42] Wu Jiaxiang, Leng Cong, Wang Yuhang, et al. Quantized convolutional neural networks for mobile devices [C] //Proc of CVPR 2016. Piscataway, NJ: IEEE, 2016: 4820-4828

[43] Gupta S, Agrawal A, Gopalakrishnan K, et al. Deep learning with limited numerical precision [C] //Proc of ICML 2015. New York: ACM, 2015: 1737-1746

[44] Ma Yufei, Suda N, Cao Yu, et al. Scalable and modularized RTL compilation of convolutional neural networks onto FPGA [C] //Proc of the 26th Int Conf on Field Programmable Logic and Applications. Piscataway, NJ: IEEE, 2016: 1-8

[45] Gysel P, Motamedi M, Ghiasi S. Hardware-oriented approximation of convolutional neural networks [J]. arXiv preprint, arXiv:1604.03168, 2016

[46] Courbariaux M, Bengio Y, David J P. Binaryconnect: Training deep neural networks with binary weights during propagations [C] //Proc of NIPS 2015. Cambridge, MA: MIT Press, 2015: 3123-3131

[47] Courbariaux M, Hubara I, Soudry D, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1 [J]. arXiv preprint, arXiv:1602.02830, 2016

[48] Rastegari M, Ordonez V, Redmon J, et al. Xnor-net: ImageNet classification using binary convolutional neural networks [C] //Proc of ECCV 2016. Berlin: Springer, 2016: 525-542

[49] Zhou Shuchang, Wu Yuxin, Ni Zekun, et al. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients [J]. arXiv preprint, arXiv: 1606.06160, 2016

[50] Hubara I, Courbariaux M, Soudry D, et al. Quantized neural networks: Training neural networks with low precision weights and activations [J]. arXiv preprint, arXiv: 1609.07061, 2016

[51] Cai Zhaowei, He Xiaodong, Sun Jian, et al. Deep learning with low precision by half-wave Gaussian quantization [C] // Proc of CVPR 2017. Piscataway, NJ: IEEE, 2017: 5918-5926

[52] Li Fengfu, Zhang Bo, Liu Bin. Ternary weight networks [J]. arXiv preprint, arXiv:1605.04711, 2016

[53] Zhu Chenzhuo, Han Song, Mao Huizi, et al. Trained ternary quantization [J]. arXiv preprint, arXiv:1612.01064, 2016

[54] Chen Wenlin, Wilson J, Tyree S, et al. Compressing neural networks with the hashing trick [C] //Proc of ICML 2015. New York: ACM, 2015: 2285-2294

[55] Cheng Yu, Yu F, Feris R, et al. An exploration of parameter redundancy in deep networks with circulant projections [C] //Proc of ICCV 2015. Piscataway, NJ: IEEE, 2015: 2857-2865

[56] Yang Zichao, Moczulski M, Denil M, et al. Deep fried convnets [C] //Proc of ICCV 2015. Piscataway, NJ: IEEE, 2015: 1476-1483

[57] Denil M, Shakibi B, Dinh L, et al. Predicting parameters in deep learning [C] //Proc of NIPS 2013. Cambridge, MA: MIT Press, 2013: 2148-2156

[58] Jaderberg M, Vedaldi A, Zisserman A. Speeding up convolutional neural networks with low rank expansions [J]. arXiv preprint, arXiv:1405.3866, 2014

[59] Denton E, Zaremba W, Bruna J, et al. Exploiting linear structure within convolutional networks for efficient evaluation [C] //Proc of NIPS 2014. Cambridge, MA: MIT Press, 2014: 1269-1277

[60] Lebedev V, Ganin Y, Rakhuba M, et al. Speeding-up convolutional neural networks using fine-tuned cp-decomposition [J]. arXiv preprint, arXiv:1412.6553, 2014

[61] Kim Y, Park E, Yoo S, et al. Compression of deep convolutional neural networks for fast and low power mobile applications [J]. arXiv preprint, arXiv:1511.06530, 2015

[62] Tai Cheng, Xiao Tong, Zhang Yi, et al. Convolutional neural networks with low-rank regularization [J]. arXiv preprint, arXiv:1511.06067, 2015

[63] Ioannou Y, Robertson D, Shotton J, et al. Training cnns with low-rank filters for efficient image classification [J]. arXiv preprint, arXiv:1511.06744, 2015

[64] Wen Wei, Xu Cong, Wu Chunpeng, et al. Coordinating filters for faster deep neural networks [C] //Proc of ICCV 2017. Piscataway, NJ: IEEE, 2017: 658-666

[65] Lin Shaohui, Ji Rongrong, Chen Chao, et al. ESPACE: Accelerating convolutional neural networks via eliminating spatial and channel redundancy [C] //Proc of AAAI 2017. Menlo Park, CA: AAAI, 2017: 1424-1430

[66] Lin Shaohui, Ji Rongrong, Guo Xiaowei, et al. Towards convolutional neural networks compression via global error reconstruction [C] //Proc of IJCAI 2017. San Francisco, CA: Morgan Kaufmann, 2016: 1753-1759

[67] Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size [J]. arXiv preprint, arXiv:1602.07360, 2016

[68] Howard A, Zhu Menglong, Chen Bo, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications [J]. arXiv preprint, arXiv: 1704.04861, 2017

[69] Zhang Xiangyu, Zhou Xinyu, Lin Mengxiao, et al. ShufflNnet: An extremely efficient convolutional neural network for mobile devices [J]. arXiv preprint, arXiv:1707.01083, 2017

[70] Chollet F. Xception: Deep learning with depthwise separable convolutions [C] //Proc of CVPR 2017. Piscataway, NJ: IEEE, 2017: 1251-1258

[71] Buciluǎ C, Caruana R, Niculescu-Mizil A. Model compression [C] //Proc of KDD 2006. New York: ACM, 2006: 535-541

[72] Ba J, Caruana R. Do deep nets really need to be deep? [C] //Proc of NIPS 2014. Cambridge, MA: MIT Press, 2014: 2654-2662

[73] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network [J]. arXiv preprint, arXiv: 1503. 02531, 2015

[74] Romero A, Ballas N, Kahou S E, et al. FitNets: Hints for thin deep nets [J]. arXiv preprint, arXiv:1412. 6550, 2014

[75] Balan A K, Rathod V, Murphy K P, et al. Bayesian dark knowledge [C] //Proc of NIPS 2015. Cambridge, MA: MIT Press, 2015: 3438-3446

[76] Luo Ping, Zhu Zhenyao, Liu Ziwei, et al. Face model compression by distilling knowledge from neurons [C] //Proc of AAAI 2016. Menlo Park, CA: AAAI, 2016: 3560-3566

[77] Zagoruyko S, Komodakis N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer [J]. arXiv preprint, arXiv: 1612. 03928, 2016

[78] Xie Saining, Girshick R, Dollár P, et al. Aggregated residual transformations for deep neural networks [C] //Proc of CVPR 2017. Piscataway, NJ: IEEE, 2017: 5987-5995

[79] Mathieu M, Henaff M, LeCun Y. Fast training of convolutional networks through FFTs [J]. arXiv preprint, arXiv:1312. 5851, 2013

[80] Lavin A, Gray S. Fast algorithms for convolutional neural networks [C] //Proc of CVPR 2016. Piscataway, NJ: IEEE, 2016: 4013-4021

[81] Zhai Shuangfei, Wu Hui, Kumar A, et al. S3Pool: Pooling with stochastic spatial sampling [J]. arXiv preprint, arXiv: 1611. 05138, 2016

[82] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images [D]. Toronto: Department of Computer Science, University of Toronto, 2009

[83] Jia Yangqing, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding [C] //Proc of the 22nd ACM Int Conf on Multimedia. New York: ACM, 2014: 675-678

[84] Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems [J]. arXiv preprint, arXiv:1603. 04467, 2016

[85] Alvarez J M, Salzmann M. Compression-aware training of deep networks [C] //Proc of NIPS 2017. Cambridge, MA: MIT Press, 2017: 856-867

Ji Rongrong, born in 1983. PhD, professor and PhD supervisor. His main research interests include computer vision, multimedia and machine learning.

Lin Shaohui, born in 1989. PhD candidate. His main research interests include machine learning and computer vision.

Chao Fei, born in 1979. PhD, associate professor and master supervisor. His main research interests include developmental robotics, machine learning, and optimization algorithms.

Wu Yongjian, born in 1982. Master. His main research interests include face recognition, image understanding, and large scale data processing.

Huang Feiyue, born in 1979. PhD. His main research interests include image understanding and face recognition.