·热点与综述· 文章编号:1000-3428(2021)02-0012-07 文献标志码:A 中图分类号:TP183

基于深度神经网络二阶信息的结构化剪枝算法

季繁繁,杨 鑫,袁晓彤
(南京信息工程大学 江苏省大数据分析技术重点实验室,南京 210044)

摘 要: 现有结构化剪枝算法通常运用深度神经网络(DNN)的一阶或者零阶信息对通道进行剪枝,为利用二阶信息加快DNN网络模型收敛速度,借鉴HAWQ算法思想提出一种新的结构化剪枝算法。采用幂迭代法得到经过预训练的网络参数对应Hessian矩阵的主特征向量,据此向量衡量网络通道的重要性并进行通道剪枝,同时对剪枝后的网络参数进行微调提高DNN分类性能。实验结果表明,该算法在网络参数量和每秒浮点运算次数分别减少29.9%和34.6%的情况下,在ResNet110网络上的分类准确率提升了0.74%,剪枝效果优于PF、LCCL等经典剪枝算法。

关键词: 深度神经网络;网络压缩;结构化剪枝;二阶信息;幂迭代法

开放科学(资源服务)标志码(OSID):

中文引用格式: 季繁繁,杨鑫,袁晓彤.基于深度神经网络二阶信息的结构化剪枝算法[J].计算机工程,2021,47(2):12-18.

英文引用格式: JI Fanfan,YANG Xin,YUAN Xiaotong.Structural pruning algorithm based on second-order information of deep neural network[J].Computer Engineering,2021,47(2):12-18.

Structural Pruning Algorithm Based on Second-Order Information of Deep Neural Network

JI Fanfan,YANG Xin,YUAN Xiaotong
(Jiangsu Key Laboratory of Big Data Analysis Technology,
Nanjing University of Information Science and Technology,Nanjing 210044,China)

【Abstract】 Most of the existing structural pruning algorithms are based on the first-order or zero-order information of Deep Neural Network(DNN).To use the second-order information of the networks for speeding up the convergence of DNN models,this paper proposes a new structural pruning algorithm based on the idea of the HAWQ algorithm.The proposed algorithm applies the power iteration method to get the max eigenvector of the Hessian matrix that is subject to the classification pre-trained network parameters.Then the obtained eigenvector is used to determine the importance of network channels and prune the channels.The pruned network parameters are slightly adjusted to improve the performance of DNN.Experimental results demonstrate that the proposed algorithm increases the classification accuracy by 0.74% when the number of network parameters is reduced by 29.9% and Floating-point Operations Per Second (FLOPS) by 34.6%,outperforming PF,LCCL and other classical pruning algorithms.

【Key words】 Deep Neural Network(DNN);network compression;structural pruning;second-order information;power iteration method

DOI: 10.19678/j.issn.1000-3428.0057967

0 概述

随着计算机运算速度的提升和大数据技术的发展,自从2012年AlexNet[1]在ImageNet竞赛中获得冠军后,研究人员不断设计出更深和更复杂的深度神经网络(Deep Neural Network,DNN)模型以期获得更好的分类精度。但深度神经网络复杂的结构所需的庞大计算资源使其在实际应用中受到一定的限制,主要表现为DNN模型在资源受限的无人机、智能手机、平板电脑等硬件设备上难以进行有效部署,因此网络压缩技术应运而生。目前主流的网络压缩方法有网络剪枝[2-4]、网络参数量化[5-6]与分解[7]、网络

基金项目: 国家自然科学基金(61876090,61936005);国家新一代人工智能重大项目(2018AAA0100400)。
作者简介: 季繁繁(1995—),男,硕士研究生,主研方向为机器学习、神经网络;杨 鑫,硕士研究生;袁晓彤,教授、博士。
收稿日期: 2020-04-03 **修回日期:** 2020-05-13 **E-mail:** nuistji@gmail.com

结构轻量化设计[8]和知识蒸馏[9]等，其中网络剪枝是对网络中的冗余元素进行裁剪，是最常用的网络压缩方法之一，其能同时对卷积层和全连接层进行裁剪，尤其是结构化剪枝方法可以不受硬件条件的限制而直接达到减少网络内存占用和网络加速的目的，并且网络剪枝还可以与其他网络压缩方法进行兼容，如剪枝后的网络能通过知识蒸馏提高网络性能[10]，或者利用网络结构轻量化设计进一步提升网络压缩效果[5]。

网络剪枝的核心是衡量元素的重要性并对其进行裁剪，现有算法提出的剪枝标准通常只涉及网络模型的零阶信息（参数本身信息）或一阶信息（梯度信息）[4,11-12]，并没有涉及网络二阶信息。然而，网络模型的二阶信息也是十分重要的信息，在机器学习优化任务中可以加快模型收敛速度[13]。在网络量化任务中，HAWQ算法[14]利用网络结构的二阶信息将网络不同层次量化为不同精度。在网络剪枝任务中，OBD[2]和OBS[3]算法均利用网络二阶信息，但是这2种算法需要计算Hessian矩阵，所以会消耗大量计算资源，并且OBS算法的改进版本L-OBS算法[15]依然需消耗较多的计算资源，而且这3种剪枝算法都属于非结构化剪枝算法，需要特定的硬件支持才能达到网络加速效果。

为将深度神经网络二阶信息用于网络通道剪枝，本文提出一种基于HAWQ[14]算法的结构化剪枝算法。该算法利用幂迭代法[16]获取网络通道的二阶信息，将其作为衡量通道重要性的依据并进行通道剪枝，再对剪枝后的网络进行微调得到紧凑网络，最终将本文算法运用于ResNet[17]网络与VGGNet[18]网络，并在Cifar10和Cifar100数据集[19]上进行实验验证。

1　相关工作

随着深度神经网络在各行各业的深入应用，深度神经网络压缩研究也受到国内外学者的广泛关注[20]。深度神经网络剪枝算法在20世纪80年代的代表算法主要有OBD[2]和OBS[3]。HAN等人[4]于2015年对深度神经网络参数进行有效裁剪，并取得了良好的效果，该方法属于静态剪枝，即裁剪的元素不可恢复。GUO等人[21]于2016年提出动态剪枝方法，对裁剪的元素采取可恢复机制，有效提高了剪枝后的网络性能。在OBS算法的基础上，DONG等人[15]于2017年将深度神经网络的全局裁剪拓展至逐层裁剪，提出L-OBS算法。

上述剪枝算法均属于非结构化剪枝，非结构化剪枝的裁剪对象为深度神经网络参数，其虽然达到了网络稀疏效果，但需要特定的硬件支持才能达到网络加速的目的。然而，结构化剪枝可以不受硬件条件的限制，因为其剪枝对象为网络卷积核，所以可以直接达到网络加速目的。文献[11]将网络通道的L_1范数作为衡量通道重要性的依据，并据此裁剪L_1范数较小的通道。文献[22]依据网络下一层的输出，裁剪对输出影响较小的卷积核达到网络压缩的目的。文献[23]在原卷积核的基础上加入协同层来控制相应卷积核参与网络运算，使得最终网络结构变得复杂，但是仍可实现网络加速。文献[24]在网络通道剪枝过程中引入网络通道可恢复机制，提高网络的容错能力。文献[12]利用深度神经网络批量归一化（Batch Normalization，BN）层信息对深度神经网络进行有效剪枝，并将剪枝算法应用于YOLO网络剪枝任务中。针对网络剪枝的预训练问题，文献[25]分别对结构化剪枝与非结构化剪枝的经典算法进行深入研究，得出网络预训练并非网络剪枝中的必要步骤，引起了学业界的广泛讨论。另外，机器学习方法[10,26]也被运用于网络剪枝任务中对网络结构实现自动化剪枝，大幅提升了网络剪枝效率。

2　结构化剪枝算法

本文提出的结构化剪枝算法首先对网络进行预训练，然后利用幂迭代法[16]得到网络参数对应Hessian矩阵的主特征向量，将得到的向量用于衡量网络通道的重要性并进行通道剪枝，最后对剪枝后的网络进行再训练提高网络性能。

2.1　网络通道二阶信息获取

2.1.1　幂迭代法

本文算法利用幂迭代法获取网络结构的二阶信息。幂迭代法是一种用于求解矩阵按模最大特征值（主特征值）及其对应特征向量（主特征向量）的方法。设$\boldsymbol{P}=(p_{ij})\in\mathbb{R}^{n\times n}$有$n$个线性无关的特征向量$\boldsymbol{x}_i(i=1,2,\cdots,n)$及其对应的特征值$\lambda_i(i=1,2,\cdots,n)$，假设特征值满足以下关系$|\lambda_1|>|\lambda_2|>\cdots>|\lambda_n|$。对于任意非零向量$\boldsymbol{u}_0\in\mathbb{R}^n$，若使用$\boldsymbol{P}$构造向量序列等式$\boldsymbol{u}_{k+1}=\boldsymbol{P}\boldsymbol{u}_k(k=0,1,\cdots)$，则式(1)成立：

$$\lim_{k\to\infty}\frac{\boldsymbol{u}_{k,i}}{\boldsymbol{u}_{k-1,i}}=\lambda_1,i=1,2,\cdots,n \tag{1}$$

其中，$\boldsymbol{u}_{k,i}$是向量$\boldsymbol{u}_k$的第i个分量值[16]，只要选取合适的随机向量$\boldsymbol{u}_0$，当k足够大时就可以计算$\boldsymbol{P}$的主特征值λ_1及其主特征向量$\boldsymbol{u}_k$。

2.1.2　二阶信息获取

由于每层通道的参数维数很大，其对计算机存储和计算造成巨大压力，因此很难直接计算网络参数的Hessian矩阵，如对于d维的向量，得到其Hessian矩阵的复杂度为$O(d^2)$。受HAWQ算法[14]启发，本文提出一种新的结构化剪枝算法，利用幂迭代获取网络二阶信息，其需要计算Hessian矩阵与向量的乘积。

下文对深度神经网络中网络通道参数Hessian矩阵对应的主特征向量计算进行具体介绍。假设一

个网络具有L_{Lr}($L_{\mathrm{Lr}}\in\mathbb{Z}$)层网络结构,每层网络有$N_{i+1}$($0\leqslant i\leqslant L_{\mathrm{Lr}}-1$)个通道,$\boldsymbol{W}_{i,j}$表示对应通道参数,因此深度学习损失函数$L(\boldsymbol{W})$为:

$$L(\boldsymbol{W})=\frac{1}{n}\sum_{i=1}^{n}l(\boldsymbol{x}_i,\boldsymbol{y}_i,\boldsymbol{W}) \tag{2}$$

其中,$\boldsymbol{W}=\{\boldsymbol{W}_1,\boldsymbol{W}_2,\cdots,\boldsymbol{W}_{\mathrm{Lr}}\}$,$l(\boldsymbol{x}_i,\boldsymbol{y}_i,\boldsymbol{W})$是训练集$(\boldsymbol{X},\boldsymbol{Y})$中训练样本的损失函数,$n$是样本数。定义损失函数对应通道参数$\boldsymbol{W}$的梯度$\boldsymbol{g}$为:

$$\boldsymbol{g}=\frac{\partial L}{\partial \boldsymbol{W}} \tag{3}$$

对于一个与梯度$\boldsymbol{g}$维数相同的随机向量$\boldsymbol{v}$存在式(4)成立:

$$\frac{\partial(\boldsymbol{g}^{\mathrm{T}}\boldsymbol{v})}{\partial \boldsymbol{W}}=\frac{\partial \boldsymbol{g}^{\mathrm{T}}}{\partial \boldsymbol{W}}\boldsymbol{v}+\boldsymbol{g}^{\mathrm{T}}\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{W}}=\frac{\partial \boldsymbol{g}^{\mathrm{T}}}{\partial \boldsymbol{W}}\boldsymbol{v}=\boldsymbol{H}\boldsymbol{v} \tag{4}$$

其中,$\boldsymbol{H}$是损失函数L对于参数$\boldsymbol{W}$的Hessian矩阵。式(4)成立的条件之一为向量$\boldsymbol{v}$是随机向量,其与参数$\boldsymbol{W}$无关。基于此,下文给出基于幂迭代法的网络Hessian矩阵主特征向量求解算法。

算法1 基于幂迭代法的网络Hessian矩阵主特征向量求解算法

输入 网络参数$\boldsymbol{W}$、随机向量$\boldsymbol{v}$($\boldsymbol{v}$与$\boldsymbol{W}$维度相同)

输出 主特征向量$\boldsymbol{V}$($\boldsymbol{V}=\boldsymbol{v}$)

1.通过后向传播计算W的梯度:g=dL/dW;

2.正则化v:v=v/‖v‖;

3.For c=1,2,…,m

4.gv=g^{T}v;

5.Hv=d(gv)/dW;

6.正则化Hv,v=Hv/‖Hv‖;

7.End For

通过算法1可以得到深度神经网络参数所对应Hessian矩阵的主特征向量。算法1的算力损耗主要为第5步,因为第5步需要求导,假设参数$\boldsymbol{W}$维数为d,最大循环次数为m,所以求解特征向量的算法复杂度为$O(md)$,相比于直接计算Hessian矩阵的计算复杂度$O(d^2)$,算法1的计算复杂度明显降低,其能避免直接求解网络的Hessian矩阵来获得其主特征向量,该特征向量包含二阶信息。

2.2 基于网络二阶信息的结构化剪枝算法

基于网络二阶信息的结构化剪枝算法框架(第i层)如图1所示,其中,P_i表示每一层的剪枝率,N_{i+1}表示输出通道数。

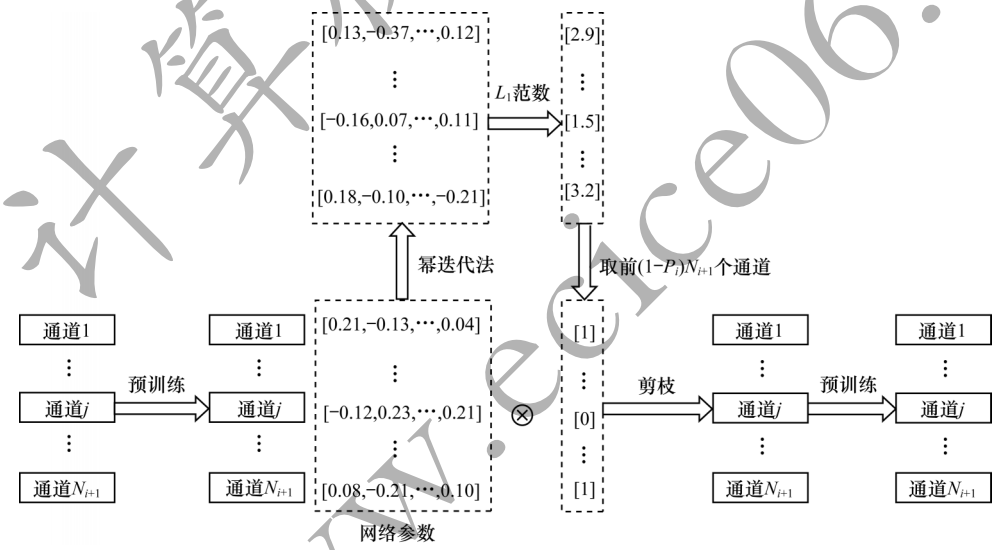

图1 基于网络二阶信息的结构化剪枝算法框架(第i层)

Fig.1 Framework of structural pruning algorithm based on second-order information of network(the i-th layer)

经过算法1可以求出网络参数对应的Hessian矩阵主特征向量$\boldsymbol{V}$,下文将具体介绍如何利用$\boldsymbol{V}$进行深度神经网络结构化剪枝,其中$\boldsymbol{V}_{i,j}$($0\leqslant i\leqslant L_{\mathrm{Lr}}-1$,$1\leqslant j\leqslant N_{i+1}$)为深度神经网络参数对应Hessian矩阵主特征向量$\boldsymbol{V}$在第i层、第j个通道的对应值。本文结构化剪枝算法属于硬剪枝[11]算法,通道重要性的衡量依据是得出的特征向量$\boldsymbol{V}$在对应通道$\boldsymbol{V}_{i,j}$的L_p范数,计算公式为:

$$\left\|\boldsymbol{V}_{i,j}\right\|_p=p\sqrt{\sum_{n=1}^{N_i}\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\left\|\boldsymbol{V}_{i,j}(n,k_1,k_2)\right\|^p} \tag{5}$$

其中:k_1、k_2分别为卷积核感受野的尺度,若为3×3卷积核,则$k_1=k_2=3$;N_i是上层卷积层的输出通道数。

硬剪枝算法需要对网络模型进行预训练,待网络模型达到较好的性能后再进行通道裁剪。因为直接裁剪后的网络性能会受到一定损耗,所以需要进行微调恢复网络性能。基于网络二阶信息的结构化剪枝算法采用预训练-剪枝-微调步骤。

算法2 基于网络二阶信息的结构化剪枝算法

输入 深度神经网络、训练数据$\boldsymbol{X}$、剪枝率$\boldsymbol{P}_i$

输出 紧凑的网络结构

1.初始化网络参数W;

2.对网络输入X进行训练,使网络损失函数收敛;

3.利用算法1计算W对应的Hessian矩阵主特征向量V;

4.For i=1,2,…,L_{Lr}-1

5.根据式(5)得到主特征向量V在对应通道$V_{i,j}$($1\leqslant j\leqslant N_{i+1}$)中的$L_1$范数$\left\| V_{i,j} \right\|_1$;

6.根据$\left\| V_{i,j} \right\|_1$进行排序选择前$(1-P_i)N_{i+1}$个$L_1$范数值最大的通道;

7.保留被选择的通道并裁剪其余通道;

8.End For

9.获得紧凑的网络结构及其对应的参数$\bar{W}$;

10.微调网络参数$\bar{W}$得到最终的网络参数W^*

算法2的第1步是对神经网络参数进行随机初始化;第2步是训练一个性能良好的深度神经网络用于剪枝,该步骤属于网络预训练过程;第3步可以通过算法1实现;第5步是计算特征向量的L_1范数,即向量元素的绝对值之和;第6步和第7步是对网络参数进行裁剪;第4步~第9步为一个内循环过程,对深度神经网络一层通道进行裁剪;第10步是微调过程,网络微调实际上是对剪枝后的深度神经网络进行再训练,由于剪枝后深度神经网络的一些网络通道会缺失,因此网络精度会降低,需要对剪枝后的网络参数$\bar{\boldsymbol{W}}$进行微调得到性能良好的网络参数$\boldsymbol{W}^*$,计算公式为:

$$\boldsymbol{W}^*=\underset{\bar{\boldsymbol{W}}}{\operatorname{argmin}}\ L(\bar{\boldsymbol{W}}) \tag{6}$$

在微调过程中的学习率和迭代次数均小于预训练时的学习率和迭代次数。

3 实验结果与分析

本节主要介绍本文结构化剪枝算法在ResNet[17]和VGGNet[18]网络模型上的压缩效果,所有实验均基于PyTorch框架[27]。

3.1 实验数据集

Cifar10数据集[19]是深度学习领域较常用的图片数据集,其常被应用于深度学习的分类任务。Cifar10数据集共分为10类,每类有6 000张图片,包括10种不同物体,每张图片为32像素×32像素的三通道RGB图片。Cifar10数据集共有50 000张训练集图片和10 000张测试集图片。

Cifar100数据集[19]是Cifar10数据集的一个衍生数据集,其所有图片均与Cifar10数据集相同。Cifar100数据集将Cifar10数据集的每类物体数据进行再划分,使原有的10类样本划分为100类样本,每类样本包含600张图片,所以Cifar100数据集可以看成Cifar10数据集的精细版本。

3.2 对比算法

在实验中,将本文结构化剪枝算法与以下算法进行对比:

1)PF算法[11]。PF算法是经典的硬剪枝算法,该算法利用已训练的网络参数L_1范数作为衡量通道重要性的标准并裁剪L_1范数较小的通道,并对裁剪后的网络进行微调。

2)LCCL算法[23]。LCCL算法在卷积层后面加入一个低损耗的协同层来控制卷积层中相应的卷积核参与运算,其虽然达到网络加速目的,但是网络结构由此变得更加复杂。

3)Rethinking算法[25]。Rethinking算法将经典的剪枝算法进行复现,使用特殊的再训练方式对裁剪后的网络参数进行赋值,从而达到良好的网络性能。

4)SFP算法[24]。SFP算法是经典的软剪枝算法,在剪枝过程中对通道采用可恢复机制,提高了算法剪枝效果,但是其相比硬剪枝算法整个剪枝过程更长。

3.3 实验设置与性能指标

本文算法在剪枝前网络模型需要进行预训练160步,初始学习率为0.1,在第80步~第120步内对学习率进行逐步衰减。微调步数为40步或80步,初始学习率设为0.001,在50%的训练步数后实施学习率衰减。算法1的最大迭代次数m取10。在VGGNet网络、ResNet56网络及ResNet110网络上,本文算法采用PF算法[11]的剪枝率。PF算法通过实验找出每个网络模型的敏感层,即对最终输出结果有很大影响的层,该算法在裁剪过程中跳过敏感层,即敏感层的剪枝率为0,其中:ResNet56网络A剪枝方式的剪枝率为10%,其跳跃层为第18层、第20层、第38层和第54层;ResNet56网络B剪枝方式的剪枝率为60%、30%和10%,其跳跃层为第16层、第18层、第20层、第34层、第38层和第54层;ResNet110网络A剪枝方式只在前期采用50%的剪枝率,其跳跃层为第36层;ResNet110网络B剪枝方式的剪枝率为50%、40%和30%,其跳跃层为第36层、第38层和第74层;VGG16网络的A剪枝方式对特定的多个卷积层采用50%的剪枝率。

网络剪枝算法的性能评价指标主要有分类准确率、参数量以及每秒浮点运算次数(Floating-point Operations Per Second,FLOPS),其中:分类准确率表征DNN模型分类结果;参数量衡量DNN所占用的计算机内存大小,参数量减少比例是剪枝后的网络与未剪枝的原始网络在网络参数量上相比减少的百分比;FLOPS是衡量网络运行速度的指标,FLOPS减少说明网络在实际运算中可以取得加速的效果,减少量越大,说明加速效果越明显。

3.4 结果分析

3.4.1 Cifar10数据集上的实验结果

表1为本文算法与PF、LCCL等剪枝算法在ResNet与VGGNet网络模型上的实验结果对比,其中直接使用文献[11,23-25]中已有对比算法的实验结果,加粗数据表示最优结果,“—”表示文献[23-24]缺少相应数

据。可以看出,经过本文算法剪枝,网络结构的参数量与FLOPS均明显减少,但是网络性能未受到影响,而且在ResNet110、VGG16等网络模型上,网络性能还得到了一定的提升,具体表现为本文算法准确率在ResNet110网络模型和VGG16网络模型上分别提升了0.74%和0.45%。

表1 5种剪枝算法在Cifar10数据集上的对比结果

Table 1 Comparison results of five pruning algorithms on Cifar10 dataset

网络模型	剪枝算法	剪枝前分类准确率/%	剪枝率/%	剪枝后分类准确率/%	减少比例/%		
					分类准确率	参数量	FLOPS
ResNet20	LCCL算法	91.53	—	91.43	0.10	—	20.30
	SFP算法	91.20	20	90.20	1.00	—	29.30
	PF算法	91.60	10	91.65	−0.05	11.24	11.81
		91.60	30	90.73	0.87	31.07	30.79
	本文算法	91.60	10	**91.69**	**−0.09**	11.24	11.81
		91.60	30	91.65	−0.05	31.07	30.79
ResNet32	LCCL算法	92.33	—	90.74	1.59	—	31.20
	PF算法	92.56	10	92.13	0.43	11.26	11.87
		92.56	30	91.53	1.03	31.09	30.94
	本文算法	92.56	10	**92.60**	**−0.04**	11.26	11.87
		92.56	30	92.53	0.03	31.09	30.94
ResNet56	SFP算法	93.59	20	**93.47**	0.12	—	28.84
	PF算法	93.04	10	93.10	−0.06	9.40	10.40
		93.04	30	93.06	−0.02	13.70	27.60
	Rethinking算法	93.14	10	93.09	−0.05	9.40	10.40
		93.14	30	93.05	0.09	13.70	27.60
	本文算法	93.18	10	93.33	**−0.15**	9.40	10.40
		93.18	30	93.10	0.08	13.70	27.60
ResNet110	LCCL算法	93.63	—	93.44	0.19	—	34.20
	SFP算法	93.68	20	93.93	−0.25	—	28.20
	PF算法	93.53	10	93.55	−0.02	2.30	15.90
		93.53	30	93.30	0.23	32.40	38.60
	Rethinking算法	93.14	10	93.25	−0.11	2.30	15.90
		93.14	30	93.60	−0.46	32.40	38.60
	本文算法	93.32	10	93.48	−0.16	2.30	15.90
		93.32	30	**94.06**	**−0.74**	29.90	34.60
VGG16	Rethinking算法	93.63	10	93.78	−0.15	64.00	34.20
	PF算法	93.63	10	93.41	0.22	64.00	34.20
	本文算法	93.41	10	**93.86**	**−0.45**	64.00	34.20

在相同的剪枝率下,与PF算法和Rethinking算法相比,本文算法在所有网络模型上均取得了较好的结果,说明本文算法的剪枝性能最优。在ResNet110网络模型上,本文算法相比LCCL算法在FLOPS减少比例相当的情况下,可以获得更高的分类准确率,且经过本文算法裁剪后的网络占用内存更小。在ResNet56网络模型上,在FLOPS减少比例相近的情况下,本文算法相比SFP算法分类准确率减少比例更具优势,说明其可以与软剪枝算法取得相当的剪枝性能且实现更简单。另外,从表1可以看到,ResNet网络随着层数的加深,取得的剪枝效果也越来越好,VGG16网络经过剪枝后分类准确率也得到了一定的提升,说明随着网络层数的加深,网络中可能存在更多的冗余元素,裁剪这些冗余元素会提升分类准确率,从而论证越复杂的深度神经网络越容易裁剪这一观点。

3.4.2 Cifar100数据集上的实验结果

表2为本文剪枝算法与PF算法在Cifar100数据集上的实验结果,其中加粗数据表示最优结果。由于文献[11]没有使用Cifar100数据集进行实验,因此该实验所有结果均是笔者复现的结果。根据表2可以看出,本文算法在更复杂的分类任务中同样可以对深度神经网络进行有效裁剪,在对分类准确率没有较大影响的情况下网络模型的参数量与FLOPS都得到明显减少,但当网络模型的剪枝率较小时,其

分类准确率反而得到了提升。与PF算法相比，本文算法在3个网络模型上均更具优势，说明其在更复杂的分类任务中也具有更好的剪枝性能。同时，对比网络模型在两个数据集上的实验结果，可以看出深度神经网络模型在Cifar100数据集上剪枝后的分类准确率减少比例更大，说明其对于不同复杂度的分类任务而言具有不同的冗余度，具体表现为在越复杂的分类任务中网络结构的冗余度越低。

表2　两种硬剪枝算法在Cifar100数据集上的对比结果

Table 2　Comparison results of two hard pruning algorithms on Cifar100 dataset

网络模型	剪枝算法	剪枝前分类准确率/%	剪枝方式	剪枝后分类准确率/%	减少比例/%		
					分类准确率	参数量	FLOPS
ResNet56	PF算法	70.38	A	70.42	−0.04	9.4	10.4
		70.38	B	69.95	0.43	13.7	27.6
	本文算法	70.38	A	**70.86**	**−0.48**	9.4	10.4
		70.38	B	70.25	0.07	32.4	38.6
ResNet110	PF算法	72.29	A	72.49	−0.20	2.3	15.9
		72.29	B	70.88	1.41	13.7	27.6
	本文算法	72.29	A	**72.53**	**−0.24**	2.3	15.9
		72.29	B	70.99	1.30	29.9	34.6
VGG16	PF算法	73.01	A	71.11	1.90	64.0	34.2
	本文算法	73.01	A	**71.62**	**1.39**	64.0	34.2

3.4.3　不同剪枝率时的实验结果

根据上述深度神经网络模型实验结果可以看出，剪枝率会对网络模型裁剪效果产生影响。为对此影响进行分析，以ResNet20网络为例对不同剪枝率下的网络模型剪枝效果进行统计，具体结果如图2所示。由于不同网络存在不同的冗余度，对于ResNet20而言，当剪枝率为10%和30%时，有助于提升深度神经网络剪枝性能，当剪枝率大于40%时，剪枝性能会下降比较快。由此可以看出，合适的网络模型剪枝率不仅不会损耗模型性能，而且还有助于提升网络性能，然而如果网络模型剪枝率设置过大，虽然网络结构变得紧凑，但是网络性能会受到较大影响，具体表现为网络模型分类准确率大幅降低，因此如何平衡剪枝算法的剪枝率与剪枝后模型的实现效果，对于深度神经网络剪枝算法而言十分重要。

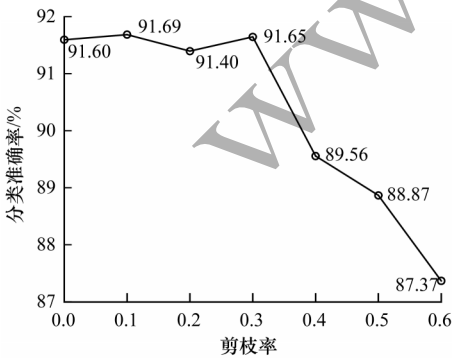

图2　不同剪枝率下的DNN模型剪枝效果

Fig.2　Pruning effect of DNN model under different pruning rates

4　结束语

本文提出一种基于深度神经网络二阶信息的结构化剪枝算法，利用幂迭代法得到经过预训练的网络模型参数Hessian矩阵的主特征向量，然后依据此向量衡量网络通道的重要性并进行通道剪枝。实验结果表明，该算法可在对原网络分类准确率影响较小的情况下实现网络参数量和FLOPS的有效裁剪。但由于本文算法目前仅在Cifar数据集上进行实验，后续可尝试使用更复杂的数据集，同时考虑与其他网络压缩方法进行融合，进一步提高深度神经网络模型的压缩效率。

参考文献

[1] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[2] LECUN Y, DENKER J S, SOLLA S A. Optimal single-class classification strategies [C]//Proceedings of the 19th International Conference on Neural Information Processing Systems. Cambridge, USA: MIT Press, 2007: 598-605.

[3] HASSIBI B, STORK D G. Second order derivatives for network pruning: optimal brain surgeon[C]//Proceedings of the 2nd order derivatives for network pruning: optimal brain surgeon. Berlin, Germany: Springer, 1993: 164-171.

[4] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural networks[EB/OL]. [2020-02-13]. https://arxiv. org/abs/1506. 02626.

[5] HAN S, MAO H, DALLY W J. Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding[EB/OL]. [2020-02-13]. https://arxiv. org/abs/1510. 00149.

[6] CHOI Y, ELKHAMY M, LEE J, et al. Towards the limit of network quantization [C]//Proceedings of International Conference on Learning Representations. Sydney, Australia: [s. n.], 2017: 1-8.

[7] ZHANG Xiangyu, ZOU Jianhua, MING Xiang, et al. Efficient and accurate approximations of nonlinear convolutional networks [C]//Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Berlin, Germany: Springer, 2015: 1984-1992.

[8] ZHANG Xiangyu, ZHOU Xinyu, LIN Mengxiao, et al. ShuffleNet: an extremely efficient convolutional neural network for mobile devices [C]//Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Berlin, Germany: Springer, 2018: 6848-6856.

[9] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network[EB/OL]. [2020-02-13]. https://arxiv. org/abs/1503. 02531.

[10] DONG Xuanyi, YANG Yi. Network pruning via transformable architecture search [C]//Proceedings of NIPS'19. Vancouver, Canada: [s. n.], 2019: 759-770.

[11] LI H, KADAV A, DURDANOVIC I, et al. Pruning filters for efficient ConvNets [C]//Proceedings of International Conference on Learning Representation. Toulon, France: [s. n.], 2016: 1-10.

[12] YANG Minjie, LIANG Yaling, DU Minghui. YOLO pruning algorithm based on parameter subspace and scaling factor [J/OL]. Computer Engineering: 1-10 [2020-02-13]. https://doi. org/10. 19678/j. issn. 1000-3428. 0056932. (in Chinese)
杨民杰, 梁亚玲, 杜明辉. 基于参数子空间和缩放因子的YOLO剪枝算法[J/OL]. 计算机工程: 1-10[2020-02-13]. https://doi. org/10. 19678/j. issn. 1000-34 28. 0056932.

[13] BOYD S, VANDENBERGHE L. Convex optimization[M]. Cambridge, UK: Cambridge University Press, 2004.

[14] DONG Z, YAO Z, GHOLAMI A, et al. HAWQ: hessian aware quantization of neural networks with mixed-precision[C]//Proceedings of 2019 IEEE/CVF International Conference on Computer Vision. Berlin, Germany: Springer, 2019: 293-302.

[15] DONG X, CHEN S, PAN S. Learning to prune deep neural networks via layer-wise optimal brain surgeon[EB/OL]. [2020-02-13]. https://arxiv. org/abs/1705. 07565v1.

[16] BOOTH T E. Power iteration method for the several largest eigenvalues and eigenfunctions [J]. Nuclear Science and Engineering, 2006, 154(1): 48-62.

[17] HE Kaiming, ZHANG Xiangyu, REN Shaoqing, et al. Deep residual learning for image recognition [C]// Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition. Berlin, Germany: Springer, 2016: 770-778.

[18] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[EB/OL]. [2020-02-13]. https://arxiv. org/abs/1409. 1556.

[19] KRIZHEVSKY A, HINTON G. Learning multiple layers of features from tiny images [D]. Toronto, Canada: University of Toronto, 2009.

[20] WANG Peisong. Acceleration and compression of deep neural networks [D]. Beijing: University of Chinese Academy of Sciences, 2018. (in Chinese)
王培松. 深度神经网络加速与压缩方法研究[D]. 北京: 中国科学院大学, 2018.

[21] GUO Yiwen, YAO Anbang, CHEN Yurong. Dynamic network surgery for efficient DNNs[C]//Proceedings of Advances in Neural Information Processing Systems. Barcelona, Spain: [s. n.], 2016: 1379-1387.

[22] LUO Jianhao, WU Jianxin, LIN Weiyao. ThiNet: a filter level pruning method for deep neural network compression [C]//Proceedings of 2017 IEEE International Conference on Computer Vision. Washington D. C., USA: IEEE Press, 2017: 5068-5076.

[23] DONG Xuanyi, HUANG Junshi, YANG Yi, et al. More is less: a more complicated network with less inference complexity[C]//Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition. Washington D. C. , USA: IEEE Press, 2017: 1895-1903.

[24] YANG He, KANG Guoliang, DONG Xuanyi, et al. Soft filter pruning for accelerating deep convolutional neural networks[EB/OL]. [2020-02-13]. https://arxiv. org/abs/1808. 06866.

[25] LIU Zhuang, SUN Mingjie, ZHOU Tinghui, et al. Rethinking the value of network pruning [EB/OL]. [2020-02-13]. https://arxiv. org/abs/1810. 05270.

[26] HE Yihui, LIN Ji, LIU Zhijian, et al. AMC: AutoML for model compression and acceleration on mobile devices [C]//Proceedings of European Conference on Computer Vision. Berlin, Germany: Springer, 2018: 815-832.

[27] PASZKE A, GROSS S, MASSA F, et al. PyTorch: an imperative style, high-performance deep learning library [EB/OL]. [2020-02-13]. https://arxiv. org/abs/1912. 01703? context=cs. LG.

编辑 陆燕菲