中国科学: 技术科学 2022 年 第 52 卷 第 5 期: 667 ~ 681
SCIENTIA SINICA Technologica techcn.scichina.com
《中国科学》杂志社 SCIENCE CHINA PRESS
论 文 紧致化视觉计算专题

基于动态稀疏和特征学习增强的模型剪枝

阮晓峰[1,2], 胡卫明[1,2,3*], 刘雨帆[1,2], 李兵[1]

1. 中国科学院自动化研究所模式识别国家重点实验室, 北京 100190;
2. 中国科学院大学人工智能学院, 北京 100049;
3. 中国科学院脑科学与智能技术卓越创新中心, 上海 200031
* E-mail: wmhu@nlpr.ia.ac.cn

收稿日期: 2021-03-15; 接受日期: 2021-08-10; 网络版发表日期: 2022-03-30
国家重点研发计划项目(编号: 2020AAA0106800)、国家自然科学基金项目(批准号: 62036011, 61721004, 61902401, 61972071, 61906052, 62036011, 61972394, U1936204, U1803119)、中国科学院前沿项目(编号: QYZDJ-SSW-JSC040)、广东省自然科学基金项目(编号: 2018B030311046)和中国科学院青年创新促进会项目(编号: 2017174)资助

摘要 为了减轻深度学习算法在实际应用中庞大的参数量和繁重的计算量, 剪枝已经被广泛地应用在模型压缩任务中. 然而, 大多数剪枝方法仅仅利用已经训练好的模型参数作为训练的初始参数, 而模型本身的特征信息没有被利用. 为此, 本文提出了一种基于模型特征学习增强的动态剪枝方法, 在训练过程中, 提出的方法不需要数据集类别标签. 一方面, 本文利用基准模型(训练好的模型)输出的预测类别信息和中间层特征作为监督信息指导压缩子模型的任务学习, 增强了压缩模型学习基准模型特征的能力; 另一方面, 本文利用不同压缩子模型的输出信息互相学习, 增强了压缩子模型之间特征学习的能力. 此外, 本文使用一种动态的结构化稀疏正则方式, 仅仅在预期稀疏的参数上施加正则, 同时使用基于泰勒级数的通道敏感性准则确定预期稀疏参数. 在优化过程中, 本文使用动态稀疏的迭代阈值收缩算法(iterative shrinkage-thresholding algorithm, ISTA)求解器解决了约束剪枝率的优化问题. 模型训练结束后, 提出的方法直接移除冗余的参数, 剪枝后的模型不需要微调. 在多个网络结构和数据集下, 本文提出的方法均获得了很好的压缩性能.

关键词 深度卷积神经网络, 模型压缩, 特征学习, 剪枝, 结构化稀疏

1 引言

近年来随着深度学习的发展, 深度卷积神经网络[1]已经取得了前所未有的巨大成功, 比如图像分类、物体检测、物体跟踪和视频理解等领域. 然而, 大量的计算资源和能耗需要影响了其在嵌入式和移动设备的部署[2]. 因此, 基于深度神经网络压缩与加速的方法不断涌现, 呈增长趋势.

结构化剪枝方法由于不需要专门的软硬件库支持, 可以直接在现有的深度学习框架上运行, 已经被广泛地研究. 在结构化剪枝方法中[3~5], 研究人员通常采用一些剪枝策略对已经训练好的模型(比如可以从Caffe Model Zoo, Torch Model Zoo等深度学习模型库中获得)进行剪枝, 然后为了恢复模型性能, 常常利用

引用格式: 阮晓峰, 胡卫明, 刘雨帆, 等. 基于动态稀疏和特征学习增强的模型剪枝. 中国科学: 技术科学, 2022, 52: 667–681
Ruan X F, Hu W M, Liu Y F, et al. Dynamic sparsity and model feature learning enhanced training for convolutional neural network-pruning (in Chinese). Sci Sin Tech, 2022, 52: 667–681, doi: 10.1360/SST-2021-0088

带有标签的数据微调(fine-tuning)剪枝后的模型(图1). 这是因为预训练+剪枝+微调的方法通常没有充分挖掘预训练模型的特征表达能力, 主要表现为多数方法仅仅利用预训练模型的参数, 而基准模型自身的特征表达能力没有被考虑(比如预训练模型的类别预测信息、中间层特征表示信息), 因此在微调过程中需要有标签的数据进行监督. 另外, 对网络进行结构化稀疏过程中, 不同的压缩子网络输出类别标签信息会有一定的差异, 这些信息对网络的训练也有一定的好处.

针对上述存在的问题, 本文提出了一种基于模型特征学习增强的动态卷积神经网络剪枝方法. 在模型训练过程中, 本文利用基准模型的特征表达信息和多个压缩子网络之间的类别信息完成整个模型压缩任务, 不需要任何数据标签信息监督, 具体框架见图2. 本文的贡献可以归纳如下.

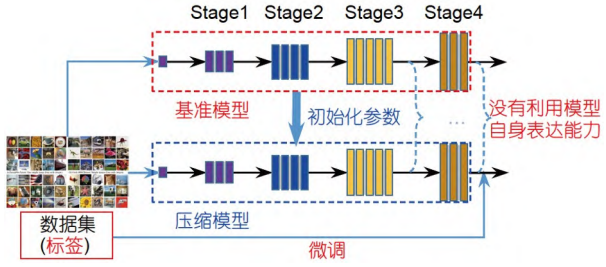


图 1 带有标签数据微调的剪枝方法

Figure 1 The pruning method with labeled data.

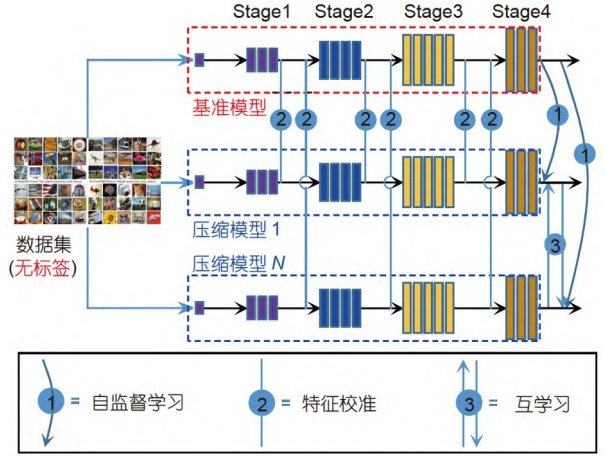


图 2 基于模型特征学习增强的剪枝方法

Figure 2 The proposed pruning method based on model feature learning enhancement.

(1) 提出一种无标签模型特征学习增强的训练方式, 并将其应用到了模型压缩任务中.

(2) 使用一种动态的结构化稀疏正则方式, 可以动态地确定不同层的稀疏分配率. 模型训练结束后, 本文提出的方法直接移除冗余的参数, 剪枝后的模型不需要微调.

(3) 在多个网络结构和数据集下, 本文提出的方法均获得了很好的性能. 在CIFAR-10数据集下, 该方法可以减少VGG-Small模型93%的参数量和70%的计算量, 精度甚至能够提升0.1%; 在精度几乎没有损失的前提下, 该方法可以减少ResNet56模型53.61%的参数量和60.33%的计算量; 对于轻量化结构设计的MobileNet v2, 该方法可以实现3倍的加速, 精度没有下降.

2 相关工作介绍

在模型剪枝方法中[6~8], 典型的剪枝框架包括三个阶段: 预训练、剪枝和微调. 首先, 模型剪枝前需要预训练好的模型; 一些方法[6,7]直接从现有的深度学习模型库中获得已经训练好的模型; 而另外一些方法[8]直接从头开始训练一个结构化稀疏的网络. 然后, 在剪枝阶段, 需要定义滤波器(通道)重要性评价方式以便移除不重要的参数, 比如滤波器权值的ℓ_1范数大小[7]、网络BN层缩放因子的大小[8]、基于权重Taylor级数展开的评价方式[9]、快速子网络的评价方式[10]. 最后, 为了获得更高的性能, 剪枝后的模型直接在带有标签的数据集下进行微调. 在上述剪枝过程中, 仅仅使用了预训练模型已经训练好的参数, 而模型本身具备的特征表达能力没有被充分挖掘. 此外, 一些方法[11,12]探讨了剪枝后模型结构的重要性, 本文提出的方法采用动态稀疏的正则方式, 可以自适应地获得结构化稀疏的网络结构.

一些模型压缩方法[13,14]利用了模型输出的类别信息和中间层特征信息, 然而仍需要带有标签的数据进行微调. Lin等人[15]提出了一种基于生成对抗学习的网络剪枝方法, 该方法实现了无类别标签端到端的模型剪枝任务, 但是有时需要通过微调技术来获得更好的效果. 本文提出了一种不需要类别标签的模型压缩框架, 结构化稀疏的网络直接被学习出来, 剪枝后的模型不需要进行微调.

3　方法

本文利用基准模型输出的预测类别信息和中间层特征作为监督信息指导压缩子模型的任务学习; 同时不同压缩子模型的输出信息相互学习. 此外, 本文采用一种端到端动态学习稀疏的模型压缩方法. 本文提出的方法详细步骤见算法1.

3.1　压缩模型学习基准模型特征信息

已知数据集$\mathcal{X}=\{x_i\}_{i=1}^{D}$为训练数据集, 类别标签未知. 基准模型(预训练模型)为$\mathcal{M}^{\mathcal{B}}$, 那么对于输入数据$x_i$, 输出类别预测值$P^{\mathcal{B}}(x_i)$为

$$P^{\mathcal{B}}(x_i)=\mathcal{F}^{\mathcal{B}}(x_i), \tag{1}$$

式中, $P^{\mathcal{B}}=\left\{p_1^{\mathcal{B}},p_2^{\mathcal{B}},...,p_M^{\mathcal{B}}\right\}$, M为特定任务类别数; $\mathcal{F}^{\mathcal{B}}(\cdot)$为通过网络所有层变换后, 对应输入数据的输出值. 需要指出, 在整个训练过程中, 基准模型通过前向传播获得相关信息, 不进行反向传播, 参数不更新.

本文定义压缩模型集为$\mathcal{M}^{\mathcal{P}}$, 那么对于有$N$个压缩子模型的压缩模型集合可以表示为

$$\mathcal{M}^{\mathcal{P}}=\{\mathcal{M}^{\mathcal{P}_1},\mathcal{M}^{\mathcal{P}_2},...,\mathcal{M}^{\mathcal{P}_N}\}. \tag{2}$$

对于第k个压缩子模型$\mathcal{M}^{\mathcal{P}_k}$, 如果输入数据$x_i$, 那么输出类别预测值$P^{\mathcal{P}_k}(x_i)$可以表示为

$$P^{\mathcal{P}_k}(x_i)=\mathcal{F}^{\mathcal{P}_k}(x_i), \tag{3}$$

式中, $P^{\mathcal{P}_k}=\left\{p_1^{\mathcal{P}_k},p_2^{\mathcal{P}_k},...,p_M^{\mathcal{P}_k}\right\}$, M为特定任务类别数; $\mathcal{F}^{\mathcal{P}_k}(\cdot)$为压缩子模型通过网络所有层变换后, 对应输入数据的输出值. 需要指出, 在整个训练过程中, 压缩模型同时进行前向传播和反向传播, 参数更新.

与传统的知识蒸馏手段不同, 由于数据集没有类别标签, 本文利用基准模型输出的类别信息指导压缩模型学习, 具体来说, 本文采用基准模型输出的类别概率与压缩模型输出预测的KL散度作为损失函数, 形式可以表示为

$$\mathcal{L}_{LPKD}=\sum_{k=1}^{N}KLDivLoss(Softmax(P^{\mathcal{P}_k},T),Softmax(P^{\mathcal{B}},T)), \tag{4}$$

式中, $KLDivLoss(\cdot)$为KL散度损失函数. 温度系数T为知识蒸馏[16]中的一个概念, 通过在$Softmax$转换函数中引入T, 模型输出类别概率的分布会发生改变. 在本文中, 由于所提出的方法不使用数据集的标签信息, 而是通过基准模型获得正确类别所属概率, 因此不设置太大的温度系数T, $Softmax(\cdot)$可以表示为

$$Softmax(P^{\mathcal{B}},T)=\frac{\exp\left(\frac{P^{\mathcal{B}}(x_i)}{T}\right)}{\sum_{m=1}^{M}\exp\left(\frac{p_m^{\mathcal{B}}}{T}\right)}, \tag{5}$$

$$Softmax(P^{\mathcal{P}_k},T)=\frac{\exp\left(\frac{P^{\mathcal{P}_k}(x_i)}{T}\right)}{\sum_{m=1}^{M}\exp\left(\frac{p_m^{\mathcal{P}_k}}{T}\right)}. \tag{6}$$

将式(5)和(6)代入式(4), 即可得到输出类别自监督学习的损失函数.

在特征迁移学习过程中, 中间层特征也是很重要的. 因此, 本文利用基准模型的中间层特征指导压缩

算法1　基于动态稀疏和特征学习增强的模型剪枝算法

输入: 基准模型$\mathcal{M}^{\mathcal{B}}$, 压缩率$PR_0$, 总的训练轮次$E$, 每一个轮次迭代次数$t_e$, 数据集$\mathcal{X}=\{x_i\}_{i=1}^{D}$, 学习率$\eta$.

输出: 压缩后的模型

利用基准模型, 复制出N个压缩模型, 第k个压缩子模型记为$\mathcal{M}^{\mathcal{P}_k}$, 初始化稀疏分配率;
for 1→E **do**
　for t=1, ..., t_e **do**
　　for k=1, ..., N **do**
　　前向传播: 对第k个压缩子模型, 计算损失$\mathcal{L}_{\mathcal{M}}^{k}$
　　反向传播、参数更新: 对第k个压缩子模型, 参数更新为 $\mathbf{W}_k^t\leftarrow\mathbf{W}_k^{t-1}-\eta\frac{\partial\mathcal{L}_{\mathcal{M}}^{k}}{\partial\mathbf{W}_k^{t-1}}$
　　参数稀疏:
　　for i=1, ..., L **do**
　　计算通道灵敏度: $S_{\text{channel}}^{(i)}=\left\{S_t(\mathcal{W}_1^{(i)}),S_t(\mathcal{W}_2^{(i)}),...,S_t(\mathcal{W}_{c^{(i)}}^{(i)})\right\}$
　　截取$S_{\text{channel}}^{(i)}$中最小$\lceil sr^{(i)}*c^{(i)}\rceil$个通道, 确定$\boldsymbol{\Theta}^{\mathcal{P}}$和$index$
　　利用公式(17)动态地稀疏参数
　　end for
　end for
end for
　重新计算压缩模型每一层稀疏分配率
end for
移除冗余的参数, 即可获得压缩后的模型

模型, 具体使用中间层输出特征的*MSE*损失作为约束, 可以表示为

$$\mathcal{L}_{IFMSE}=\sum_{k=1}^{N} MSELoss(Feature^{\mathcal{B}}, Feature^{\mathcal{P}_k}), \tag{7}$$

式中, $Feature^{\mathcal{B}}$为基准模型中间层特征, $Feature^{\mathcal{P}_k}$为第k个压缩子模型中间层特征.

3.2 多个压缩模型输出信息互补学习

对于压缩模型, 定义$P^{\widetilde{\mathcal{P}}_k}(x_i)=\left\{p_1^{\widetilde{\mathcal{P}}_k}, p_2^{\widetilde{\mathcal{P}}_k}, ..., p_M^{\widetilde{\mathcal{P}}_k}\right\}$为除第$k$个压缩子模型外其余$N-1$个子模型输出类别预值的平均值, 计算公式为

$$P^{\widetilde{\mathcal{P}}_k}(x_i)=\frac{1}{N-1}\sum_{l=1,l\neq k}^{N}\mathcal{F}^{\mathcal{P}_l}(x_i). \tag{8}$$

受深度互学习[17]和在线集成模型压缩[18]方法启发, 为了使压缩模型中每一个压缩子模型之间互相增强学习, 本文将压缩子模型(去除需要学习更新权重的模型)的类别输出平均值作为监督信息, 同样采用该输出平均值的*Softmax*概率与需要更新学习权重的压缩子模型输出预测值*Softmax*概率的*KL*散度作为损失函数, 具体形式可以表示为

$$\mathcal{L}_{MLKD}=\sum_{k=1}^{N} KLDivLoss(Softmax(P^{\mathcal{P}_k}, T), Softmax(P^{\widetilde{\mathcal{P}}_k}, T)). \tag{9}$$

需要指出, 在模型训练过程中, 使用$P^{\widetilde{\mathcal{P}}_k}(x_i)$值作为监督信息, 其对应的模型不参与反向传播. 类似地, $Softmax(P^{\widetilde{\mathcal{P}}_k}, T)$可以表示为

$$Softmax(P^{\widetilde{\mathcal{P}}_k}, T)=\frac{\exp\left(\frac{P^{\widetilde{\mathcal{P}}_k}(x_i)}{T}\right)}{\sum_{m=1}^{M}\exp\left(\frac{p_m^{\widetilde{\mathcal{P}}_k}}{T}\right)}. \tag{10}$$

3.3 端到端动态学习稀疏的模型压缩框架

在以前的方法中[8,19], 结构化稀疏正则通常被引入到损失函数中, 结合一些优化求解器可以进行求解. 但是, 惩罚因子无法控制剪枝率. 最后, 为了满足预设剪枝率, 剪枝后的模型需要通过微调技术来恢复精度. 为了克服上面的缺点, 不同于以前的稀疏正则策略, 在训练过程中, 本文仅仅在预期稀疏的参数上施加稀疏, 具体将参数集$\boldsymbol{\Theta}$划分为两部分: 预期稀疏的参数集$\boldsymbol{\Theta}^{\mathrm{p}}$和其补集$\overline{\boldsymbol{\Theta}^{\mathrm{p}}}$. 受组稀疏(group lasso)方法[20]启发, 本文使用参数矩阵的ℓ_{21}范数作为稀疏正则, 同时考虑卷积层输出通道对应当前层和下一层的参数[21]. 于是, 本文采用的动态稀疏正则[22]可以表示为

$$\begin{aligned}&\mathcal{R}_{DPSS}(\boldsymbol{\Theta}^{\mathrm{p}})\\&=\sum_{i=1}^{L}\sum_{j=1}^{c^{(i)}-p^{(i)}}\left\|(\mathbf{W}_{j,:}^{(i)})_{\mathrm{out}}^{2D}\right\|_2+\sum_{i=1}^{L-1}\sum_{j=1}^{c^{(i)}-p^{(i)}}\left\|(\mathbf{W}_{j,:}^{(i+1)})_{\mathrm{in}}^{2D}\right\|_2,\end{aligned} \tag{11}$$

式中, $(\mathbf{W}^{(i)})_{\mathrm{out}}^{2D}$和$(\mathbf{W}^{(i)})_{\mathrm{in}}^{2D}$分别表示第$i$个卷积层参数张量$\mathcal{W}^{(i)}$沿输出和输入通道展开的结果, $c^{(i)}$和$p^{(i)}$分别是卷积层i的输出通道和预期保留通道的个数.

使用基于泰勒级数展开的敏感性评价方式[5], 同时考虑当前层和下一层的参数, 第i个卷积层第j个通道的敏感性可以表示为

$$\begin{aligned}S_t(\mathcal{W}_j^{(i)})=&\left|\sum_{m=1}^{r^{(i)}k_1^{(i)}k_2^{(i)}}\frac{\partial\mathcal{L}(\cdot)}{\partial(\mathbf{W}_{j,m}^{(i)})_{\mathrm{out}}^{2D}}(\mathbf{W}_{j,m}^{(i)})_{\mathrm{out}}^{2D}\right.\\&\left.+\sum_{m=1}^{c^{(i+1)}k_1^{(i+1)}k_2^{(i+1)}}\frac{\partial\mathcal{L}(\cdot)}{\partial(\mathbf{W}_{j,m}^{(i+1)})_{\mathrm{in}}^{2D}}(\mathbf{W}_{j,m}^{(i+1)})_{\mathrm{in}}^{2D}\right|.\end{aligned} \tag{12}$$

对于卷积神经网络第i层, 稀疏分配率$sr^{(i)}$等于稀疏率(参数为零)和诱导稀疏率(冗余的非零参数)之和. 为了避免引入额外的超参数, 本文使用统一的比率计算不同层的诱导稀疏率. 模型训练时, 因为不同层的稀疏率在不断变化, 所以稀疏分配率也动态更新. 本文采用的方法不需要预定义剪枝结构, 能够自适应地学习结构化稀疏的网络结构.

那么对于N个压缩模型, 总的正则项$\mathcal{R}_s$可以表示为

$$\mathcal{R}_s=\sum_{k=1}^{N}\mathcal{R}_{DPSS}^{k}. \tag{13}$$

3.4 总的训练损失函数

最终, 本文所提方法总的损失函数为

$$\mathcal{L}=\lambda_1\mathcal{L}_{LPKD}+\lambda_2\mathcal{L}_{IFMSE}+\lambda_3\mathcal{L}_{MLKD}+\lambda_s\mathcal{R}_s, \tag{14}$$

式中, λ_1, λ_2, λ_3和λ_s分别用来调节不同损失函数的权重.

3.5 优化

3.5.1 基准模型与压缩模型初始化

与ThiNet[6]和GAL[15]方法一样, 压缩模型网络结构和参数来自于基准模型, 如图3所示, 基准模型使用预训练的参数和网络结构复制出N个压缩模型.

3.5.2 优化过程

本文提出的模型压缩框架包括基准模型和N个压缩模型. 其中只有压缩模型参数需要更新. 为了后续分析方便, 本文将损失函数$\mathcal{L}$前三项统一为模型特征学习的损失函数, 记为$\mathcal{L}_{\mathcal{M}}$, 那么总的损失函数可以重新表示为

$$\mathcal{L}=\mathcal{L}_{\mathcal{M}}+\lambda_s\mathcal{R}_s. \tag{15}$$

接下来, 本文通过最小化损失函数$\mathcal{L}$求解. 在损失函数$\mathcal{L}$中, $\mathcal{L}_{\mathcal{M}}$是可微分的, 而正则项由于含有绝对值项, 其形式不可以微分, 因此, 无法利用梯度下降法对上述优化问题直接求解. 在梯度下降方法中, 核心思想在某点处用近似方法得到下一个迭代点. 因此, 可以将梯度下降法的二阶近似思想推广到此优化问题中. 于是, 可以得到近似求解公式:

$$\begin{aligned}&\mathbf{W}^t=\mathrm{prox}_{\eta,\lambda_s\mathcal{R}_s(\cdot)}\left(\mathbf{W}^{t-1}-\eta\nabla\mathcal{L}_{\mathcal{M}}(\mathbf{W}^{t-1})\right),\\&t=1,2,3,\ldots,\end{aligned} \tag{16}$$

式中, prox(·)为近端算子符号, $\mathbf{W}$为压缩模型参数, η为学习率.

对于稀疏正则项$\mathcal{R}_s(\cdot)$, 利用ISTA算法[23], 可以获得近端算子的表达式, 参数$(\mathbf{W}_{j,m}^{(i)})^{2D}\in\mathbf{\Theta}^{\mathrm{p}}$通过$\left\{(\mathbf{W}_{j,m}^{(i)})^{2D}\right\}^{(n+1)}=S_\lambda\left(\left\{(\mathbf{W}_{j,m}^{(i)})^{2D}\right\}^{(n)}\right)$更新, 其中,

$$S_{\eta\lambda_s}\left((\mathbf{W}_{j,m}^{(i)})^{2D}\right)=\begin{cases}(\mathbf{W}_{j,m}^{(i)})^{2D}-\dfrac{\eta\lambda_s(\mathbf{W}_{j,m}^{(i)})^{2D}}{\left\|(\mathbf{W}_{j,:}^{(i)})^{2D}\right\|_2},\\\qquad\text{if }\left\|(\mathbf{W}_{j,:}^{(i)})^{2D}\right\|_2>\eta\lambda_s,\\0,\quad\text{otherwise.}\end{cases} \tag{17}$$

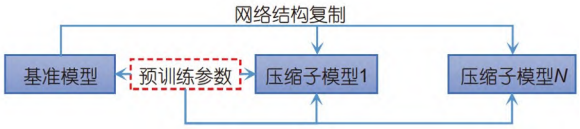


图 3 (网络版彩图)基准模型与压缩模型初始化

Figure 3 (Color online) Well-trained model and compression models initialization.

4 实验与分析

本节首先介绍了实验设置, 包括数据集、网络模型、部署详情、参数设置、评测指标和对比的主流方法; 然后, 在不同数据集和网络上对提出的方法与主流的方法进行了对比; 最后, 对提出的方法进行了分析与讨论.

4.1 实验设置

(1) 数据集和网络结构. 本文使用三个常用的数据集(即CIFAR-10, CIFAR-100和ImageNet)评估所提出来的方法, 其中CIFAR-10和CIFAR-100数据集包含50000张训练图片和10000张测试图片, ImageNet数据集包含1280000张训练图片和50000张测试图片(来自于验证集). 数据增强策略参考文献[24]. 在CIFAR-10和CIFAR-100数据集上, 对VGG-Small, ResNets, PreActResNet50和MobileNet v2网络进行压缩实验; 在更大规模的数据集ImageNet上, 通过压缩ResNet50验证本文所提方法的有效性. VGG-Small具体结构参见文献[24], ResNets和PreActResNet50结构来自于文献[1,25], MobileNet v2来自于文献[26].

(2) 部署详情. 本文使用PyTorch深度学习框架完成算法构建, 服务器环境配置多块NVIDIA RTX 2080 Ti GPU和Intel(R) Xeon(R) Gold 5118 CPU卡. 在CIFAR-10和CIFAR-100数据集上, 训练总轮次数(total epochs)设置为200, 最小批次(batch size)设置为64, 初始学习率设置为0.1, 其中学习率衰减方式为分段常数衰减, 即学习率在轮次100和150处分别降为0.01和0.001. 在训练MobileNet v2时, 学习率采用余弦衰减的方式. 在ImageNet上, 训练总轮数设置为20, 最小批次为256, 初始学习率为0.01, 在轮次10和15处学习率分别降为0.001和0.0001. 模型训练使用SGD优化器, 设置权值衰减系数为10^{-4}, 动量设置为0.9.

(3) 参数设置. 在CIFAR-10和CIFAR-100数据集上,

由于基准模型输出类别概率信息已经足够指导压缩模型的学习, 因此, 设置λ_1=1.0, λ_2=0, λ_3=1.0. 在ImageNet数据集上, 同样固定λ_3=1.0, 本文采用网格搜索的方法确定λ_1和λ_2, 首先分别部署三组实验, 即设置(λ_1, λ_2)=(1.0, 0.0), (λ_1, λ_2)=(0.5, 0.5), (λ_1, λ_2)=(0.0, 1.0), 模型压缩后对应的精度分别为59.95%, 70.41%, 70.50%, 显然将(λ_1, λ_2)=(0.0, 1.0)选择为较好的点; 然后分别设置(λ_1, λ_2)=(0.3, 0.7), (λ_1, λ_2)=(0.1, 0.9)部署两组实验, 模型压缩后对应的精度分别为70.55%, 70.53%, 显然(λ_1, λ_2)=(0.3, 0.7)对应最好的结果. 最后, λ_1设置为0.3, λ_2设置为0.7. λ_s的设置在实验部分说明; 温度系数T设置为1; 压缩子模型个数N设置为2.

(4) 评测指标与对比方法. 本文使用参数剪枝率和FLOPs剪枝率来评价提出的方法, 具体公式如下:

$$PR_{\text{Params}} = 1 - \frac{\#\text{Pruned Params}}{\#\text{Original Params}},$$

$$PR_{\text{FLOPs}} = 1 - \frac{\#\text{Pruned FLOPs}}{\#\text{Original FLOPs}},$$

式中, #Pruned Params和#Pruned FLOPs分别指模型剪枝后的参数和FLOPs; #Original Params和#Original FLOPs分别指原始网络的参数和FLOPs.

为了评估所提方法的压缩性能, 本文将其与多种主流的模型压缩方法进行了比较, 具体包括GrOWL[27], Li等人[7], ThiNet[6], Slimming[8], NRE[28], SSS[29], DCP[13], SFP[30], ASFP[31], FPGM[32], AMC[33], COP[34], KSE[35], GAL[15], HRank[36], DHP[37], SFS&DFS[38], LFPC[39]和DMC[40].

4.2 与主流方法比较

4.2.1 CIFAR-10上的结果

在CIFAR-10上, 本文对VGG-Small, ResNets(包括ResNet56和ResNet110)以及MobileNet v2网络进行了压缩, 对比方法包括GrOWL, Li等人提出的方法, Slimming, NRE, SFP, ASFP, FPGM, AMC, DCP, COP, KSE, HRank, DHP, SFS&DFS和DMC.

(1) VGG-Small模型的压缩结果. 在实验中, 设置λ_s=0.002. 从表1的比较结果可以得出, VGG-Small网络参数有大量的冗余, 比如在对比的方法中, 最低的参数压缩率达到63.90%(Li等人提出的方法), 因此本文尝试获得更大的参数压缩率. 与主流模型压缩方法相比, 本文提出的方法取得了最好的结果, 具体表现为: 当参数剪枝率和FLOPs剪枝率分别为93.02%和71.17%, 使用本文提出的方法压缩后模型的精度甚至超过了基准模型(提升0.1%), 提出的算法获得了大压缩比的无损剪枝. 进一步增大压缩率, 在精度只比基准模型下降不到0.2%的情况下, 使用提出的方法, 参数剪枝率达到了95%, FLOPs剪枝率达到了77.71%, 压缩性能远远超过了对比的方法. 这些结果说明了尽管本文不使用数据集的类别标签, 提出的算法仍然可以达到好的压缩效果.

表 1 在CIFAR-10数据集, 压缩VGG-Small网络性能[a)]

Table 1 The pruning results of VGG-Small on CIFAR-10

网络结构	对比方法	测试精度	PR_{Params}	PR_{FLOPs}
VGG-Small	Baseline	93.98%	0	0
	GrOWL	92.20%	91.26%	–
	Li等人	93.40%	63.90%	34.21%
	NRE	93.40%	92.72%	67.64%
	Slimming	93.48%	86.65%	43.50%
	COP	93.31%	92.80%	73.50%
	SFS&DFS	94.05%	90.70%	56.30%
	HRank	93.43%	82.90%	53.50%
	本文方法	**94.08%**	93.02%	71.17%
	本文方法	93.80%	**95.00%**	**77.71%**

a) 黑体为最大值

(2) ResNets模型的压缩结果. 在实验中, 设置λ_s=0.002. 与VGG-Smal网络相比, ResNets模型更加紧凑, 比如VGG-Small的参数是ResNet56的17倍之多. 尽管如此, 本文提出的方法仍然可以获得超过50%的压缩率. 如表2所示, 对于ResNet56网络, 当参数剪枝率为45.88%以及FLOPs剪枝率为53.20%时, 使用本文提出的方法压缩模型后, 精度甚至比基准模型提升了将近0.3%, 远远高于对比的方法, 例如, 比HRank高0.9%. 进一步设置更大的压缩率, 当参数剪枝率为53.61%以及FLOPs剪枝率为60.33%, 本文提出的方法压缩性能远远超过了对比的方法, 同时压缩后的模型精度仅下降了0.04%. 对于层数更深的ResNet110网络, 当获得与DHP(压缩后模型精度最高的算法)精度一样高的压缩模型时, 与DHP压缩结果比较, 使用提出的方法参数压缩率高了12%以及FLOPs压缩率高了24%, 模型理论计算量远远超过了DHP方法. 当进一步提高压缩率, 本文提出的方法可以压缩ResNet110约74%的参数、

表 2　在CIFAR-10数据集, 压缩ResNets网络性能[a)]

Table 2　The pruning results of ResNets on CIFAR-10

网络结构	对比方法	测试精度	PR_{Params}	PR_{FLOPs}
ResNet56	Baseline	93.78%	0	0
	Li等人	93.06%	13.70%	27.60%
	SFP	93.35%	–	52.60%
	ASFP	93.12%	–	52.60%
	DCP	93.49%	49.24%	49.75%
	FPGM	93.49%	–	52.60%
	AMC	91.90%	–	50.00%
	KSE	93.23%	50.00%	52.38%
	DMC	93.69%	–	50.00%
	HRank	93.17%	42.40%	50.00%
	DHP	93.58%	41.58%	49.04%
	本文方法	**94.07%**	45.88%	53.20%
	本文方法	93.74%	**53.61%**	**60.33%**
ResNet110	Baseline	93.92%	0	0
	Li等人	93.30%	–	38.60%
	SFP	93.86%	–	40.80%
	ASFP	93.10%	–	52.30%
	FPGM	93.85%	–	52.30%
	HRank	93.36%	59.20%	58.20%
	DHP	**94.63%**	36.80%	36.34%
	本文方法	**94.63%**	49.02%	60.49%
	本文方法	93.50%	**74.23%**	**80.12%**

a) 黑体为最大值

80%的FLOPs, 精度比基准模型低了近0.5%. 因此, 对于ResNet结构, 在CIFAR-10上, 提出的算法也有显著的压缩率, 同时获得了高精度的压缩模型.

(3) MobileNet v2模型的压缩结果. 在实验中, 设置λ_s=0.0005. MobileNet v2是一个轻量化的网络结构, 广泛地应用于实际场景中, 因此压缩MobileNet v2网络很有必要. 如表3所示, 当压缩超过50%的参数和计算量时, 使用本文提出的方法压缩后的模型精度甚至比基准模型高了0.4%, 与对比的方法相比, 无论在压缩率还是精度, 提出的方法均达到了最好的效果. 这些结果验证了本文提出的方法对于轻量化的网络模型压缩也是有效的.

4.2.2　CIFAR-100上的结果

在CIFAR-100上, 本文通过对VGG-Small和ResNet56网络进行压缩, 进一步评估提出的方法, 对比方法包括Li等人, Slimming, SSS, SFP, FPGM, COP, SFS&DFS, LFPC.

表 3　在CIFAR-10数据集, 压缩MobileNet v2网络性能[a)]

Table 3　The pruning results of MobileNet v2 on CIFAR-10

网络结构	对比方法	测试精度	PR_{Params}	PR_{FLOPs}
MobileNet v2	Baseline	94.36%	0	0
	DCP	94.69%	23.66%	26.47%
	DMC	94.49%	–	40.00%
	本文方法	**94.77%**	**51.72%**	**51.98%**

a) 黑体为最大值

(1) VGG-Small模型的压缩结果. 在实验中, 设置λ_s=0.002. 和CIFAR-10数据集相比, CIFAR-100更具有挑战性, 如表4所示, 本文提出的方法仍然取得了最好的效果, 在参数剪枝率和FLOPs剪枝率上远远超过了SFS&DFS, 分别超过了约14%和30%. 这些结果充分验证了本文所提方法的有效性.

(2) ResNet56模型的压缩结果. 在实验中, 设置λ_s=0.002. 如表5所示, 在CIFAR-100上, ResNet56压缩更具有挑战性. 在FLOPs剪枝率约为30%的情况下, 使用本文提出的方法压缩后的模型精度超过基准模型约0.9%. 当设置更大的压缩率, 与对比方法持平时, 使用本文提出的方法压缩后的模型精度比LFPC高约0.5%, 比SFP高约2.5%. 这些结果说明, 提出的方法在困难的任务上也取得了优秀的结果.

4.2.3　ImageNet上的结果

在更具挑战性的数据集ImageNet上, 本文不使用数据标签压缩ResNet50. 在实验中, 设置λ_s=0.006. 如表6所示, 与ThiNet和SSS方法相比, 提出的方法在FLOPs剪枝率更大的情况下, 仍然获得了最好的性能, 精度分别高出0.35%和0.23%. 与GAL方法相比, 提出的方法达到了良好的性能. 这些结果进一步地说明了本文所提方法的有效性.

4.3　分析与讨论

4.3.1　消融分析

本节主要验证基准模型类别预测标签指导压缩模型的有效性和压缩子网络之间相互学习的有效性. 为此, 在CIFAR-10/100数据集上, 通过压缩VGG-Small和

表 4 在CIFAR-100数据集, 压缩VGG-Small网络性能[a)]

Table 4 The pruning results of VGG-Small on CIFAR-100

网络结构	对比方法	测试精度	PR_{Params}	PR_{FLOPs}
VGG-Small	Baseline	73.78%	0	0
	Li等人	71.98%	63.80%	34.18%
	Slimming	73.40%	68.23%	30.65%
	SSS	73.30%	66.67%	36.20%
	COP	71.77%	73.20%	43.10%
	SFS&DFS	73.69%	75.90%	45.00%
	本文方法	**74.23%**	79.55%	57.15%
	本文方法	73.91%	**90.01%**	**75.22%**

a) 黑体为最大值

表 5 在CIFAR-100数据集, 压缩ResNet56网络性能[a)]

Table 5 The pruning results of ResNet56 on CIFAR-100

网络结构	对比方法	测试精度	PR_{Params}	PR_{FLOPs}
ResNet56	Baseline	72.12%	0	0
	SFP	68.79%	–	52.60%
	FPGM	69.66%	–	52.60%
	LFPC	70.83%	–	51.60%
	本文方法	**73.01%**	15.34%	30.23%
	本文方法	71.34%	**35.13%**	**53.08%**

a) 黑体为最大值

表 6 在ImageNet数据集, 压缩ResNet50网络性能[a)]

Table 6 The pruning results of ResNet50 on ImageNet

网络结构	对比方法	测试精度	PR_{FLOPs}
ResNet50	ThiNet-70	90.67%	36.79%
	GAL-0.5	90.94%	43.00%
	SSS	90.79%	43.06%
	本文方法	**91.63%**	32.36%
	本文方法	91.02%	**44.53%**

a) 黑体为最大值

ResNet56网络来验证这两点. 在实验中, 统一设置λ_s=0.002.

一方面, 本节采用基准模型和压缩模型输出类别标签概率的*KL* loss值来表征两个网络输出结果的一致性. 如图4所示, 在模型训练早期, 由于压缩子模型被施加稀疏正则, 两个网络的*KL* loss值会变大. 之后随着压缩子模型稀疏状态逐渐稳定, loss值又开始下降. 最终, 随着学习率的下降, 两者的*KL* loss值收敛到很小的值. 对于结构复杂性不同的网络, 更紧凑的ResNet56收敛值略高于VGG-Small, 说明越紧凑的模型越难压缩. 对于不同的数据集, CIFAR-100比CIFAR-10有更大的收敛值, 说明在更有挑战性的数据集下, 越不容易压缩. 综上所述, 不需要类别标签, 本文提出的方法训练过程仍然稳定, 证明了基准模型类别预测标签指导压缩模型是有效的.

另一方面, 本节通过对比模型训练过程中的测试精度来验证压缩子网络互学习的有效性. 如图5所示, 在训练早期, 由于压缩子模型之间没有太大的差异性, 两者曲线基本重合. 之后压缩子模型会获得更高的精度, 这是由于不同稀疏状态的压缩子模型表现出差异, 图6给出了两个压缩子模型在训练过程中测试精度做差的曲线图, 从图中可以看出压缩子网络之间的差异信息. 当训练结束后, 本文提出的方法性能提升最高超过1个百分点. 因此, 本文使用压缩子网络的互学习也是有效的.

4.3.2 训练轮次数对模型性能的影响

训练总轮次数客观上可以反映压缩模型学习基准模型特征的速度, 因此本节比较了不同训练轮次数下压缩模型的性能. 本节设置总的训练轮次分别为40, 80, 120, 160和200. 为了验证不同压缩率下的性能, 在CIFAR-10下, 设置两种参数剪枝率(即93%和95%)压缩VGG-Small网络, 从图7中可以得出: (1) 随着训练总轮次数的增加, 模型的性能在逐步提升, 当轮次数增加到一定数目时, 精度提升空间有限. 比如剪枝率为93%时, 轮次数为120, 160和200获得的压缩模型精度相差不到0.1%. (2) 与Baseline相比, 在保持原始精度前提下, 本文提出的方法使用120个训练轮次数可以减少VGG-Small网络约93%的参数量和71%的FLOPs, 与两种需要微调的方法Slimming和HRank相比, 在FLOPs剪枝率指标下, 提出的方法超过约17%. 这些说明了提出的方法充分学习到了基准模型的特征, 有利于压缩模型获得更好的性能. (3) 提出的方法使用80个训练轮次数, 获得了与Slimming, HRank接近的精度, 从文献[8,36]可以得到Slimming需要160个epoch, HRank需要390个epoch. 这说明提出的方法需要少于50%训练轮次数仍然可以获得很高的性能. (4) 进一步使用更少的训练轮次数, 即仅仅使用40个轮次数, 仍然可以

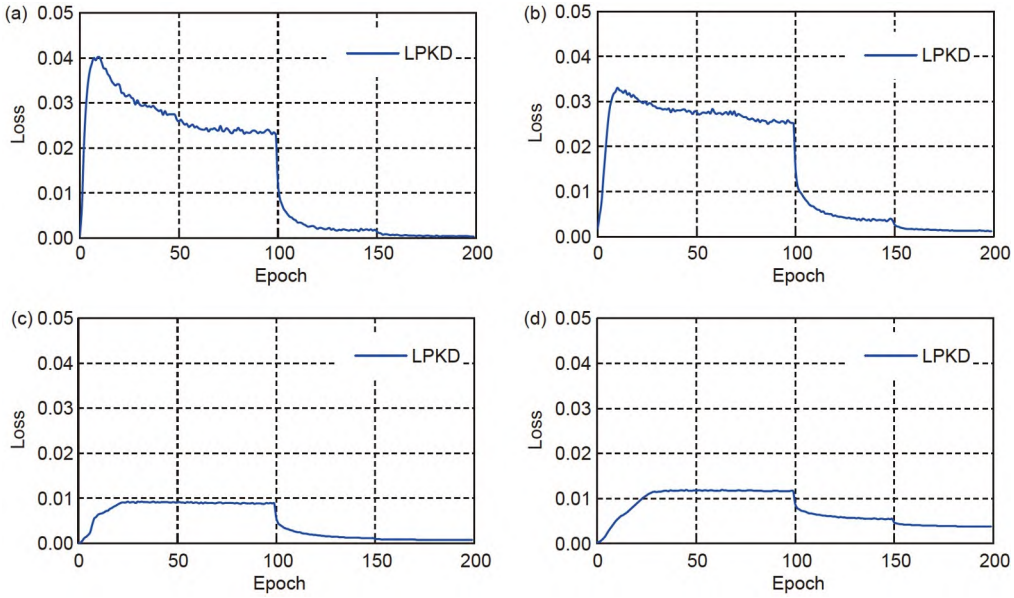


图 4　在CIFAR-10和CIFAR-100数据集下, 基准网络类别预测标签的有效性分析. (a) VGG-Small@CIFAR-10; (b) ResNet56@-CIFAR-10; (c) VGG-Small@CIFAR-100; (d) ResNet56@CIFAR-100

Figure 4　Efficacy analysis of category level information of the well-trained model on CIFAR-10/100. (a) VGG-Small@CIFAR-10; (b) ResNet56@CIFAR-10; (c) VGG-Small@CIFAR-100; (d) ResNet56@CIFAR-100.

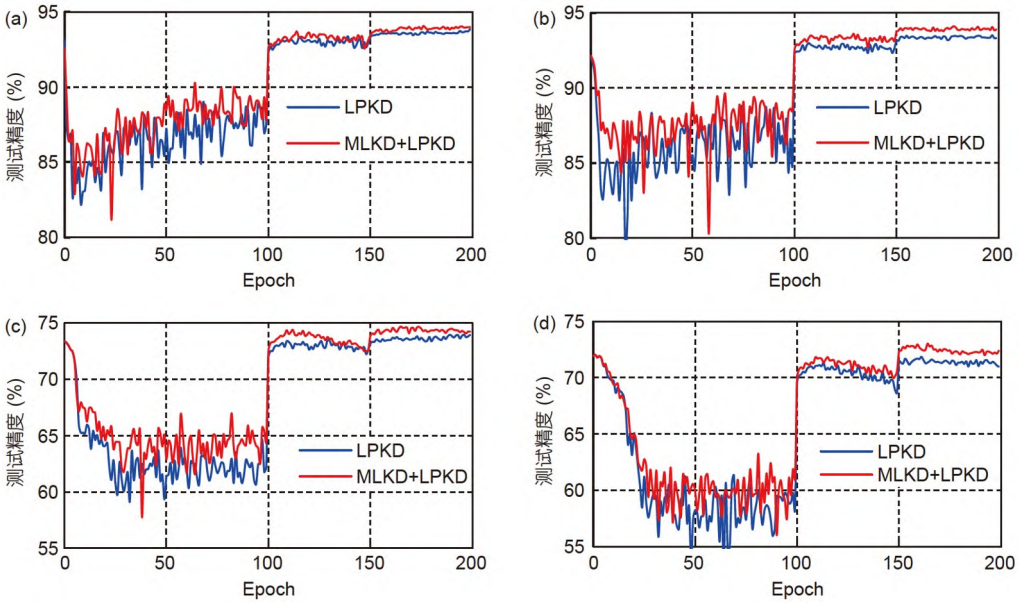


图 5　在CIFAR-10和CIFAR-100数据集下, 压缩子网络互学习的有效性分析. (a) VGG-Small@CIFAR-10; (b) ResNet56@CI-FAR-10; (c) VGG-Small@CIFAR-100; (d) ResNet56@CIFAR-100

Figure 5　Efficacy analysis of learning from each other between compression models on CIFAR-10/100. (a) VGG-Small@CIFAR-10; (b) ResNet56@CIFAR-10; (c) VGG-Small@CIFAR-100; (d) ResNet56@CIFAR-100.

获得有意义的性能, 提出的方法在参数和FLOPs剪枝率远远超过Slimming和HRank的同时, 精度接近持平(低0.1%). 总之, 通过分析模型总的训练轮次数, 可以得出本文提出的方法能够更快地学习到基准模型的特

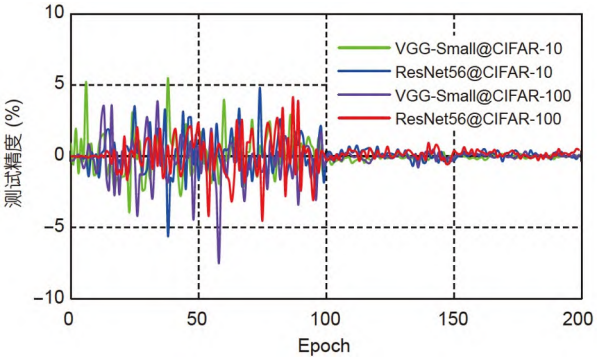


图 6 不同压缩子模型性能差异曲线

Figure 6 Accuracy differences between the compression models.

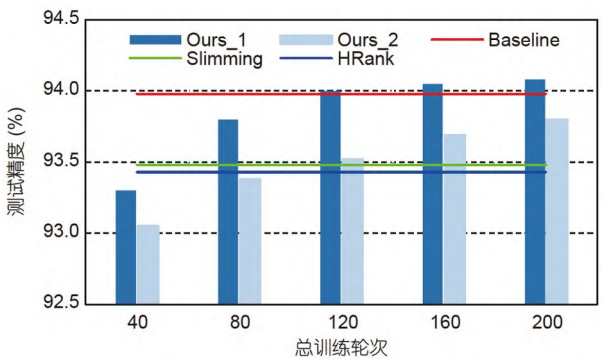


图 7 不同训练总轮次的测试精度

Figure 7 Accuracy using the different epochs.

征, 同时快速地完成模型压缩任务.

4.3.3 通道可视化分析

从压缩后模型层通道数来分析提出来的压缩方法. 在CIFAR-10下, 通过压缩VGG-Small和MobileNet v2获得压缩后的模型, 在CIFAR-100下, 本节仅仅压缩VGG-Small, 如图8所示, 可视化了模型压缩前后不同层的通道数. 在实验中, 分别设置两种压缩率(不同的网络和数据集上)完成6组实验, 具体设置见表7. 从图中可以得出以下结论. (1) 对于VGG-Small结构: 模型最后几个层的通道被大量移除, 比如CIFAR-10数据集下, 第10, 11, 12和13层移除了大约95%的通道, 这说明VGG-Small网络对于CIFAR10/100数据集来说, 在最后几个层有大量的冗余. 对于MobileNet v2结构: 提出的方法在降采样操作(卷积操作步长为2)发生时, 会保留更多的通道提取特征. 这些可视化结果证明提出的压缩方法可以自适应地学习出压缩后的结构. (2) 在同一个数据集不同压缩率下, 剩余通道曲线走势相似, 比如在CIFAR-10数据集下最大的通道对应层为第5层. 因此, 在获得最优压缩结构后, 可以通过等比例缩放的方式快速获得不同剪枝率的模型结构. (3) 在不同的数据集下, 更加简单的任务能够获得更大的压缩性能, 比

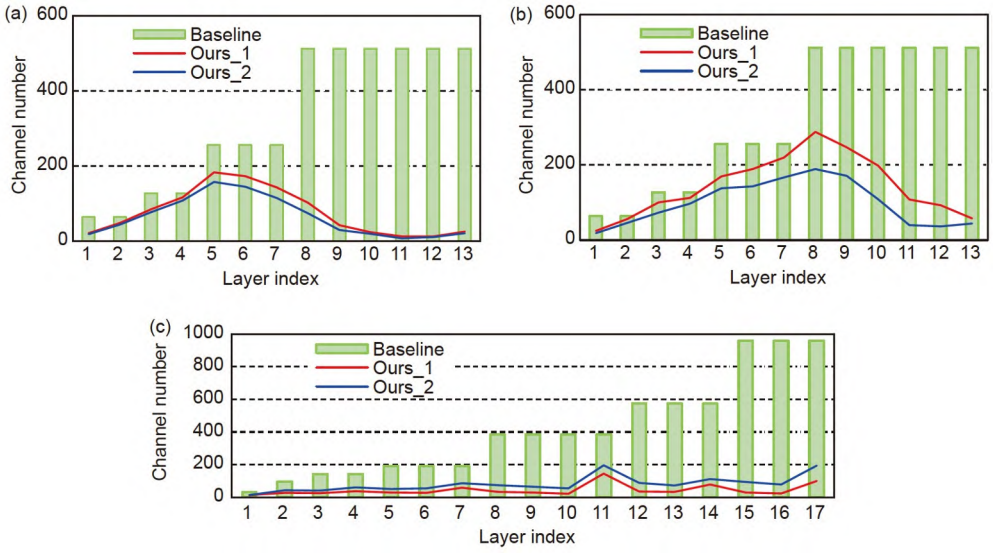


图 8 在CIFAR-10和CIFAR-100数据集下, 压缩模型通道数可视化. (a) VGG-Small@CIFAR-10; (b) VGG-Small@CIFAR-100; (c) MobileNet v2@CIFAR-10

Figure 8 The channel number of pruned models on CIFAR-10 and CIFAR-100. (a) VGG-Small@CIFAR-10; (b) VGG-Small@CIFAR-100; (c) MobileNet v2@CIFAR-10.

表 7　在CIFAR-10/100数据集下, 不同模型压缩率和精度

Table 7　The pruning ratio and accuracy on CIFAR-10/100

数据集	对比模型	PR_{Params}	PR_{FLOPs}	测试精度
CIFAR-10	VGG-Small (Ours_1)	93.02%	71.17%	94.08%
	VGG-Small (Ours_2)	95.00%	77.71%	93.80%
CIFAR-100	VGG-Small (Ours_1)	79.55%	57.15%	74.23%
	VGG-Small (Ours_2)	90.01%	75.22%	73.91%
CIFAR-10	MobileNet v2 (Ours_1)	67.25%	70.07%	94.36%
	MobileNet v2 (Ours_2)	73.84%	79.32%	93.18%

如在提升性能接近的情况下, VGG-Small在CIFAR-10上可以压缩93%的参数, 在CIFAR-100下, 只能压缩90%. 因此, 模型压缩是一个针对实际任务的问题. 总之, 在没有类别标签时, 本文所提出的压缩方法可以自动地学习到最佳的网络模型结构.

4.3.4　不同剪枝率下的实际加速和内存消耗

为了验证本文提出的方法在实际中的性能, 本节给出了不同压缩率下模型的参数、FLOPs、推理时间、消耗内存和模型存储大小. 具体来说, 在CIFAR-10和CIFAR-100下, 压缩PreActResNet50和MobileNet v2网络. 在实验部署中, 对于PreActResNet50, 分别设置FLOPs剪枝率为70%, 75%, 80%和85%, 对于MobileNet v2, 分别设置FLOPs剪枝率为50%, 60%和70%. 测试时, 在CPU卡下仅使用一个线程, 同时设置测试批次大小为1, 表8给出了不同剪枝率下的各项指标值. 对于PreActResNet50模型: 当剪枝率约为70%时, 相比于Baseline, 在CIFAR-10下, 压缩后的模型推理时间减少了41 ms, 内存下降了75.3 MB, 模型大小减少了56.62 MB, 精度提升0.32%; 在CIFAR-100下, 压缩后的模型推理时间减少了43 ms, 内存下降了68.2 MB, 模型大小减少了53.9 MB, 精度提升0.86%. 当剪枝率达到约85%, 在CIFAR-10数据集, 压缩后的模型精度比基准模型低0.56%, 推理时间减少到55 ms, 消耗内存占用77.3 MB, 模型大小不到17 MB; 在CIFAR-100数据集下, 压缩后的模型精度下降约2%, 推理时间减少到56 ms, 消耗内存占用78.3 MB, 模型大小不到18 MB. 对于轻量化模型MobileNet v2: 当剪枝率约为50%时, 相比于Baseline, 在CIFAR-10下, 压缩后的模型推理时间减少了16 ms, 内存下降了9.1 MB, 模型大小减少了4.39 MB, 精度提升0.64%; 在CIFAR-100下, 压缩后的模型推理时间减少了17 ms, 内存下降了9.2 MB, 模型大小减少了3.61 MB, 精度提升1.21%. 当剪枝率达到约70%, 对于CIFAR-10数据集, 压缩后的模型精度与基准模型持平, 推理时间减少到36 ms, 消耗内存占用28.7 MB, 模型大小不到3 MB; 对于CIFAR-100数据集, 压缩后的模型精度下降约0.71%, 推理时间减少到36 ms, 消耗内存占用28.9 MB, 模型大小为4 MB. 这些结果说明本文提出的方法在不同剪枝率下均取得了好的压缩效果, 有的精度甚至超过了基准模型.

4.3.5　其他分析

具体比较了实际加速率、消耗内存降低率和模型大小减少率, 从图9可以得出: (1) 对于PreActResNet50, 压缩后的模型在加速约2倍时, 性能降低0.02%(CIFAR-10)和0.32%(CIFAR-100); 对于MobileNet v2, 由于采用轻量化的网络结构设计, 加速倍率略低于PreActResNet50, 当压缩后的模型加速约1.7倍, 在CIFAR-10下性能与基准模型持平, 在CIFAR-100下性能比基准模型低0.71%; (2) 对于消耗内存降低率和模型大小减少率, 当压缩PreActResNet50时, 压缩后的模型在精度下降不到0.5%的情况下, 消耗内存降低率超过了50%, 模型大小减少率超过了70%; 当压缩MobileNet v2时, 压缩后的模型在精度下降不到0.8%的情况下, 消耗内存降低率超过了30%, 模型大小减少率超过了55%. 这些结果验证了本文所提出的方法在实际推理时也表现出良好的性能.

最后, 比较了不同批次大小下模型的实际推理性能. 在实验中, 设置批次大小分别为1, 16, 32, 64和128. 如图10所示, 随着测试批次数的增加, 实际加速倍率增大, 当批次大小达到64, 实际加速倍率基本不变. 与基准模型相比, 在精度保持持平的前提下, 本文提出

表 8 不同剪枝率下, 模型实际运行指标表

Table 8 The real-world performance in different pruning ratios

数据集	模型	剪枝率 (%)	参数 ($\times10^6$)	FLOPs ($\times10^8$)	推理时间 (ms)	消耗内存 (MB)	模型大小 (MB)	精度 (%)
CIFAR-10	PreActResNet50	0	23.51	25.96	116	187.1	89.02	95.07
		~70	8.44	7.65	75	111.8	32.40	95.39
		~75	7.09	6.40	67	100.7	27.26	95.23
		~80	5.62	5.00	60	88.2	21.63	95.05
		~85	4.36	3.92	55	77.3	16.81	94.51
	MobileNet v2	0	2.24	1.76	60	42.7	8.72	94.36
		~50	1.09	0.88	44	33.6	4.33	95.00
		~60	0.91	0.70	40	29.9	3.63	94.56
		~70	0.73	0.53	36	28.7	2.93	94.36
CIFAR-100	PreActResNet50	0	23.69	25.96	118	188.8	90.63	78.15
		~70	9.57	7.59	75	120.6	36.73	79.01
		~75	8.15	6.48	69	108.4	31.29	78.60
		~80	6.35	5.17	62	92.9	24.44	77.83
		~85	4.66	3.89	56	78.3	17.98	76.18
	MobileNet v2	0	2.35	1.76	60	44.0	9.12	76.04
		~50	1.40	0.88	43	34.8	5.51	77.25
		~60	1.21	0.70	40	32.5	4.76	76.60
		~70	1.01	0.53	36	28.9	4.00	75.33

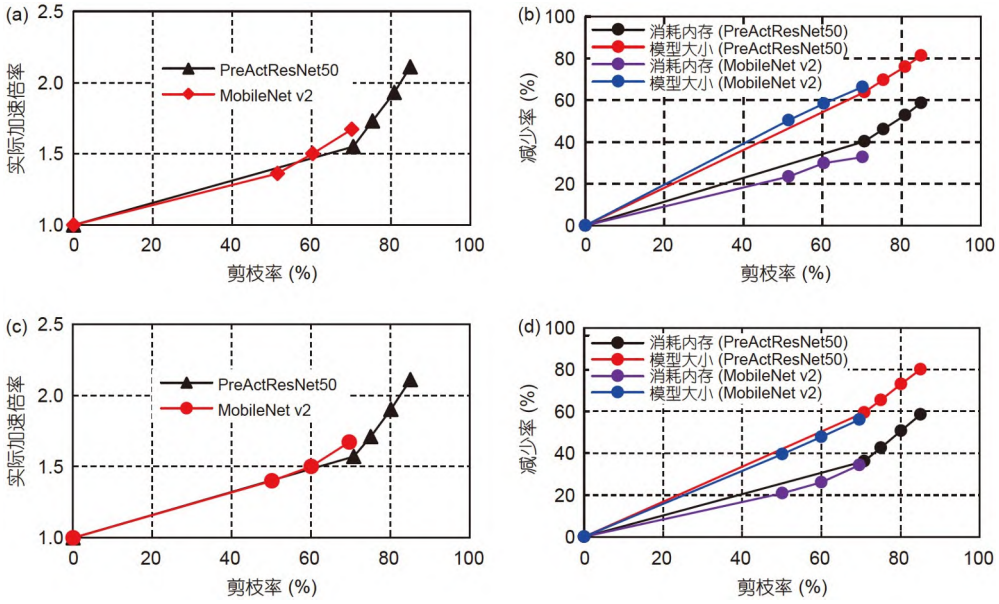
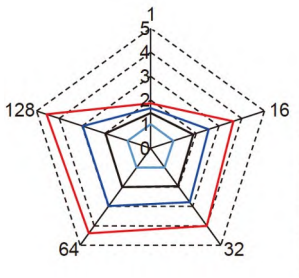


图 9 不同剪枝率下, 模型对应的实际加速率、消耗内存减少率和模型大小减少率. (a) CIFAR-10数据集; (b) CIFAR-10数据集; (c) CIFAR-100数据集; (d) CIFAR-100数据集

Figure 9 The comparison of speedup, run-time memory, and model size in different pruning ratios. (a) CIFAR-10 data; (b) CIFAR-10 data; (c) CIFAR-100 data; (d) CIFAR-100 data.



模型	精度	PR_{FLOPs}
Baseline	94.36%	0
WM-0.5	92.96%	71.21%
Ours_1	94.36%	70.07%
Ours_2	93.18%	79.32%

图 10　在不同测试批次下, MobileNet v2实际加速性能比较
Figure 10　The comparison of speedup in the different test batch size (MobileNet v2).

的方法实际加速接近3倍; 当实际加速约4.6倍时, 压缩后模型的精度只降了1.18%, 而WM[26]方法获得了约2倍的加速, 性能比本文提出的方法低0.22%. 接着, 比较了实际加速与理论FLOPs减少率, 将理论FLOPs减少率换算成理论加速倍率, 压缩后的模型1(Ours_1)实际加速比理论加速低0.37倍(2.97 vs. 3.34), 压缩后的模型2(Ours_2)实际加速比理论加速低0.27倍(4.57 vs. 4.84). 这是因为在实际推理时, 受硬件平台、数据IO方式的影响. 对于WM算法来说, 压缩后的模型实际加速与理论加速相差1.47倍(2 vs. 3.47), 超过了本文提出的算法, 可能是由于WM直接使用等比缩放的方式获得压缩模型, 没有对不同层通道做出优化. 总之, 本文所提出的方法在实际应用中也可以获得大的加速倍率, 超过了对比的方法.

5　结论

本文提出了一种基于模型特征学习增强的动态剪枝方法, 在训练过程中, 提出的方法不需要数据集类别标签. 为了充分利用基准模型的特征, 本文使用基准模型的输出信息和中间层特征信息指导压缩模型的学习, 同时使用压缩模型之间输出类别信息相互学习. 此外, 本文使用一种动态的结构化稀疏正则方式, 可以自适应地获得结构化稀疏的网络结构. 模型训练结束后, 提出的方法直接移除冗余的参数, 剪枝后的模型不需要微调. 大量的实验结果验证了本文所提出的方法可以显著地提升模型压缩率, 并取得了很好的精度.

参考文献

1 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, 2016. 770–778
2 Ji R R, Lin S H, Chao F, et al. Deep neural network compression and acceleration: A review (in Chinese). J Comput Res Develop, 2018, 55: 1871–1888 [纪荣嵘, 林绍辉, 晁飞, 等. 深度神经网络压缩与加速综述. 计算机研究与发展, 2018, 55: 1871–1888]
3 He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. Venice, 2017. 1389–1397
4 Luo J H, Zhang H, Zhou H Y, et al. Thinet: Pruning CNN filters for a thinner net. IEEE Trans Pattern Anal Mach Intell, 2019, 41: 2525–2538
5 Lin S, Ji R, Li Y, et al. Accelerating convolutional networks via global & dynamic filter pruning. In: Proceedings of International Joint Conference on Artificial Intelligence, 2018. 2425–2432
6 Luo J H, Wu J, Lin W. Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE International Conference on Computer Vision, 2017. 5058–5066
7 Li H, Kadav A, Durdanovic I, et al. Pruning filters for efficient ConvNets. In: Proceedings of International Conference on Learning Representations (ICLR). 2017
8 Liu Z, Li J, Shen Z, et al. Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE International Conference on Computer Vision, 2017. 2736–2744
9 Molchanov P, Mallya A, Tyree S, et al. Importance estimation for neural network pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Long Beach, 2019. 11264–11272
10 Li B, Wu B, Su J, et al. EagleEye: Fast sub-net evaluation for efficient neural network pruning. In: Proceedings of the European Conference on Computer Vision, 2020. 639–654
11 Liu Z, Sun M, Zhou T, et al. Rethinking the value of network pruning. In: Proceedings of International Conference on Learning Representations (ICLR). 2019

12 Frankle J, Carbin M. The Lottery Ticket Hypothesis: Finding sparse, trainable neural networks. In: Proceedings of International Conference on Learning Representations (ICLR). 2019
13 Zhuang Z, Tan M, Zhuang B, et al. Discrimination-aware channel pruning for deep neural networks. In: Advances in Neural Information Processing Systems, 2018. 875–886
14 Lin S, Ji R, Chen C, et al. Holistic CNN compression via low-rank decomposition with knowledge transfer. IEEE Trans Pattern Anal Mach Intell, 2018, 41: 2889–2905
15 Lin S, Ji R, Yan C, et al. Towards optimal structured CNN pruning via generative adversarial learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019. 2790–2799
16 Geoffrey H, Oriol V, Dean J. Distilling the knowledge in a neural network. arXiv: 1503.02531
17 Zhang Y, Xiang T, Hospedales T M, et al. Deep mutual learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 4320–4328
18 Walawalkar D, Shen Z, Savvides M. Online ensemble model compression using knowledge distillation. In: Proceedings of the European Conference on Computer Vision, 2020. 18–35
19 Wen W, Wu C, Wang Y, et al. Learning structured sparsity in deep neural networks. In: Advances in Neural Information Processing Systems, 2016. 2074–2082
20 Yuan M, Lin Y. Model selection and estimation in regression with grouped variables. J R Stat Soc B, 2006, 68: 49–67
21 Li J, Qi Q, Wang J, et al. OICSR: Out-in-channel sparsity regularization for compact deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Long Beach, 2019. 7046–7055
22 Ruan X, Liu Y, Li B, et al. DPFPS: Dynamic and progressive filter pruning for compressing convolutional neural networks from scratch. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2021. 2495–2503
23 Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J Imag Sci, 2009, 2: 183–202
24 Ruan X, Liu Y, Yuan C, et al. EDP: An efficient decomposition and pruning scheme for convolutional neural network compression. IEEE Trans Neur Netw Learning Syst, 2021, 32: 4499–4513
25 He K, Zhang X, Ren S, et al. Identity mappings in deep residual networks. In: Proceedings of the European Conference on Computer Vision, 2016. 630–645
26 Sandler M, Howard A, Zhu M, et al. Mobilenet v2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 4510–4520
27 Zhang D, Wang H, Figueiredo M, et al. Learning to share: Simultaneous parameter tying and sparsification in deep learning. In: Proceedings of International Conference on Learning Representations (ICLR). 2018
28 Jiang C, Li G, Qian C, et al. Efficient DNN neuron pruning by minimizing layer-wise nonlinear reconstruction error. In: Proceedings of International Joint Conference on Artificial Intelligence, 2018. 2298–2304
29 Huang Z, Wang N. Data-driven sparse structure selection for deep neural networks. In: Proceedings of European Conference on Computer Vision, 2018. 317–334
30 He Y, Kang G, Dong X, et al. Soft filter pruning for accelerating deep convolutional neural networks. In: Proceedings of International Joint Conference on Artificial Intelligence, 2018. 2234–2240
31 He Y, Dong X, Kang G, et al. Asymptotic soft filter pruning for deep convolutional neural networks. IEEE Trans Cybern, 2020, 50: 3594–3604
32 He Y, Liu P, Wang Z, et al. Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019. 4340–4349
33 He Y, Lin J, Liu Z, et al. AMC: AutoML for model compression and acceleration on mobile devices. In: Proceedings of European Conference on Computer Vision, 2018. 784–800
34 Wang W, Fu C, Guo J, et al. COP: Customized deep model compression via regularized correlation-based filter-level pruning. In: Proceedings of International Joint Conference on Artificial Intelligence, 2019. 3785–3791
35 Li Y, Lin S, Zhang B, et al. Exploiting kernel sparsity and entropy for interpretable CNN compression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019. 2800–2809
36 Lin M, Ji R, Wang Y, et al. HRank: Filter pruning using high-rank feature map. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020. 1529–1538

37 Li Y, Gu S, Zhang K, et al. DHP: Differentiable meta pruning via HyperNetworks. In: Proceedings of the European Conference on Computer Vision, 2020. 608–624
38 Li H, Ma C, Xu W, et al. Feature statistics guided efficient filter pruning. In: Proceedings of International Joint Conference on Artificial Intelligence, 2020. 2619–2625
39 He Y, Ding Y, Liu P, et al. Learning filter pruning criteria for deep convolutional neural networks acceleration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020. 2009–2018
40 Gao S, Huang F, Pei J, et al. Discrete model compression with resource constraint for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020. 1899–1908

Dynamic sparsity and model feature learning enhanced training for convolutional neural network-pruning

RUAN XiaoFeng[1,2], HU WeiMing[1,2,3], LIU YuFan[1,2] & LI Bing[1]

[1] *National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China;*
[2] *School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China;*
[3] *CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai 200031, China*

Recently, model-pruning approaches have become popular in reducing the high burden of deep neural networks in real-world applications. However, several existing pruning methods simply use a well-trained model to initialize parameters without considering its feature representation. Thus, we propose a label-free and dynamic pruning method based on model feature learning enhanced training. Furthermore, we use the category-level information and features of intermediate layers (well-trained model) to guide the task learning of the compression models, which enhances their ability to learn the features of the well-trained model. Additionally, we use different submodels (compression models) output information to learn from one another, promoting the feature learning ability between different submodels. Moreover, we use a structured sparsity-inducing regularization in a dynamic sparsity manner. The expected pruning parameters are identified using Taylor series-based channel sensitivity criteria. The proposed method solves the optimization problem using an iterative shrinkage-thresholding algorithm with dynamic sparsity. After the training is complete, the proposed method only eliminates redundant parameters without fine-tuning. Extensive experimental results show that the proposed method achieves good compression performance on multiple datasets and networks.

deep convolutional neural network, model compression, feature learning, pruning, structured sparsity

doi: 10.1360/SST-2021-0088