控制与决策
Control and Decision

低碳算法的发展及压缩和加速技术的应用

赵洪科, 叶倩彤, 张志勇, 张凯, 汪珂航, 黄振亚

引用本文:
赵洪科, 叶倩彤, 张志勇, 等. 低碳算法的发展及压缩和加速技术的应用[J]. 控制与决策, 2025, 40(5): 1409-1428.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2024.0936

低碳算法的发展及压缩和加速技术的应用

赵洪科[1,2]，叶倩彤[1,2]，张志勇[1,2]，张 凯[3,4†]，汪珂航[3,4]，黄振亚[3,4]

(1. 天津大学 管理与经济学部，天津 300072；2. 天津大学 复杂管理系统实验室，天津 300072；3. 中国科学技术大学 计算机科学与技术学院，合肥 230000；4. 中国科学技术大学 认知智能全国重点实验室，合肥 230000)

摘 要: 探讨低碳神经网络算法的设计及其在工业界和大型模型中的应用. 首先, 介绍低碳算法的概念及碳足迹视角下的深度学习算法; 随后, 深入研究多种设计策略, 如剪枝、量化、低秩分解等, 这些策略能显著降低数据中心和网络设备的资源消耗, 推动绿色计算的发展. 此外, 关注了低碳算法的实际应用, 包括低精度计算、高效硬件设计和硬件加速, 展示了其在减少能源浪费和环境影响方面的潜力. 对于大语言模型 (LLMs), 讨论了训练过程中的压缩技术、模型结构优化等方法, 以降低这类高资源需求模型的环境负担. 最后, 提出了评判标准来衡量不同算法的效能, 并展望低碳算法未来的发展方向及其对可持续发展的重要意义, 旨在促进低碳算法的研究与应用, 为构建可持续的数字社会贡献力量.

关键词: 低碳算法；模型压缩；大语言模型；可持续计算；模型加速；模型压缩

中图分类号: TP183 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2024.0936

引用格式: 赵洪科, 叶倩彤, 张志勇, 等. 低碳算法的发展及压缩和加速技术的应用 [J]. 控制与决策, 2025, 40(5): 1409-1428.

Development of low carbon algorithms and application of compression and acceleration techniques

ZHAO Hong-ke[1,2]， *YE Qian-tong*[1,2]， *ZHANG Zhi-yong*[1,2]， *ZHANG Kai*[3,4†]， *WANG Ke-hang*[3,4]， *HUANG Zhen-ya*[3,4]

(1. College of Management and Economics， Tianjin University， Tianjin 300072， China； 2. Laboratory of Computation and Analytics of Complex Management Systems, Tianjin University， Tianjin 300072， China； 3. School of Computer Science and Technology， University of Science and Technology of China，Hefei 230000，China； 4. State Key Laboratory of Cognitive Intelligence，University of Science and Technology of China，Hefei 230000，China)

Abstract: This paper explores the design of low-carbon neural network algorithms and their applications in industry and large-scale models, introducing the concept of low-carbon algorithms and the perspective of carbon footprints in deep learning. The paper then discusses various design strategies such as pruning, quantization, and low-rank decomposition, which significantly reduce resource consumption in data centers and network devices, promoting green computing. It also highlights practical applications, including low-precision computation, efficient hardware design, and hardware acceleration, demonstrating their potential to reduce energy waste and environmental impact. For large language models(LLMs), the paper covers compression techniques and model structure optimization to mitigate the environmental burden of these high-resource-demand models. Additionally, it proposes evaluation criteria to measure the efficiency of different algorithms and looks ahead to the future development of low-carbon algorithms and their significance for sustainable development. This work aims to advance the research and application of low-carbon algorithms, contributing to the creation of a sustainable digital society.

Keywords: low-carbon algorithms； model compression； large language models； sustainable computing； model acceleration；model compression

收稿日期: 2024-08-06；**录用日期:** 2024-12-06.

基金项目: 国家自然科学基金项目 (72471165, 72101176).

†通信作者. E-mail: kkzhang08@ustc.edu.cn.

0 引 言

随着人类活动的不断增加, 大量温室气体的排放导致地球保温效应加强, 全球气候变暖成为当今全球面临的严重挑战之一. 为了应对这一问题, 各国在 2015 年达成巴黎协定, 旨在将全球平均气温升温幅度控制在 2℃ 以内, 并努力争取将升温幅度限制在 1.5℃ 以内[1]. 然而, 要实现这一升温限制目标, 各国都需要付出巨大的努力. 中国作为全球最大的温室气体排放国之一, 亦需采取积极措施. 2020 年 9 月 22 日, 中国国家主席习近平在第七十五届联合国大会一般性辩论上宣布: “中国将提高国家自主贡献力度, 采取更加有力的政策和措施, 二氧化碳排放力争于 2030 年前达到峰值, 努力争取 2060 年前实现碳中和.” 碳达峰、碳中和目标的提出, 在国内国际社会引发关注. 中国旨在通过如加强推广可再生能源、提高能源利用效率、优化能源结构、实现经济的结构性调整等措施减少温室气体的排放[2], 在国际合作中承担更多的责任, 共同应对全球气候变暖挑战. 在此全球背景下, 信息技术领域的碳排放也成为关注的焦点, 尤其是在大规模计算任务中, 如何减少能源消耗和碳足迹已成为关键议题.

在过去的十几年里, 深度学习研究一直专注于提高精确度和准确性, 在图像分类[3-5]、图像识别[6-7]、文本分类[8-9] 等任务上展现出了卓越的性能, 并在许多领域准确性已经能够超过人类. 随着计算机硬件性能的不断提升和算法技术的不断优化, 大语言模型产生并发展越来越快. 然而, 这些卓越的性能大多以增加网络复杂性、参数数量、预测延迟等为代价. 同时, 大规模数据处理和复杂计算问题的需求增长, 也会导致大量的能源消耗和碳排放. 作为人类主体社会责任的延展, 算法责任也应具有现实意义及正当性[10-11]. 因此, 如何在满足大规模模型的计算需求的同时, 减少其碳足迹和成本, 让算法在发展同时也承担起为环境和社会可持续发展的责任, 已经成为一个亟待解决的问题.

低碳算法的概念体系建立在绿色计算 (green computing) 的基础之上. 绿色计算的提出, 旨在通过优化计算资源的管理和硬件设计, 减少能源消耗和环境负担, 推动可持续计算的发展[12]. 然而, 随着全球对环境问题、碳排放及碳中和的关注不断提升, 特别是在联合国的《巴黎协定》和可持续发展目标 13 (气候行动) 的号召下[13], 以及各大科技企业在实践中的积极响应 (如谷歌和微软的低碳技术投资与碳排放承诺)[14-15], 低碳这一概念变得愈发重要. 因此, 本文提出了低碳算法, 专注于绿色计算中高效计算的碳排放部分, 旨在通过优化算法设计、硬件加速器的使用以及系统资源管理来直接减少计算任务产生的碳足迹.

低碳算法不仅继承了绿色计算在提高能效和减少资源浪费方面的核心理念, 还将重点聚焦于碳排放的量化与减少. 与绿色计算中广泛的节能目标不同, 低碳算法特别关注计算过程中碳足迹的精确量化与控制. 通过使用如 CO_2e 等碳排放标准化评估方法, 低碳算法可以精确衡量不同算法的碳排放, 提供实现碳减排的有效工具. 这种技术不仅能够响应全球应对气候变化的紧迫需求, 还通过减少冗余计算和不必要的能源消耗, 提升计算系统的整体环境效益. 低碳算法的提出与发展, 旨在帮助全球实现碳中和的目标, 成为推动低碳经济和可持续计算发展的重要技术路径.

本研究将围绕以下核心内容展开讨论:

1) 介绍深度学习与碳排放之间的关系, 着重分析深度学习算法在计算过程中产生的碳足迹, 涵盖当前深度学习算法能耗问题及其对环境的影响.

2) 详细探讨多种低碳算法设计策略, 旨在实现可持续的绿色计算实践.

3) 聚焦于工业领域的具体应用, 以展示低碳算法如何在实际生产环境中减少能源浪费并减轻环境影响.

4) 集中讨论大规模低碳算法设计, 深入研究大型语言模型 (LLMs), 以降低大型模型的资源需求, 使它们更加环保和可持续.

5) 通过对主流算法模型的综合效果比较, 展示了不同低碳算法在多个应用场景下的性能和效率.

6) 引入一套用于评估低碳算法的标准和碳排放量化工具, 以帮助研究人员和从业者更好地评估和比较算法的效果.

最后, 本研究提供了对低碳算法未来发展的详细展望, 强调其在推动可持续发展和应对全球环境挑战中的关键作用. 本文旨在推动低碳算法的研究和应用, 为实现可持续数字社会的目标贡献一份力量.

1 碳足迹视角下的深度学习

机器学习以及深度学习的快速发展重塑了每个人的日常生活, 给人们的生活带来全新范式. 但其代价使得人工智能的发展注定是一个双刃剑. 一方面, 它可以帮助减少气候危机的影响, 例如智能电网设计、开发低排放基础设施以及气候变化预测建模. 另

一方面, 深度学习模型本身就是一个重要的碳排放者. 因此, 许多研究人员开始对计算资源与技术发展的不平衡产生担忧, 也对其带来的环境问题开始重视. 例如, GPT-3 包括 1 750 亿个参数, 仅训练一次迭代就要花费数百万美元, 并需要使用约 1 000 块 A100 显卡近 1 个月的时间才能完成[16]. Strubell 等[17] 的工作首先使人们关注深度学习对环境的影响, 它量化了通过神经架构搜索训练 Transformer 模型产生的排放量, 发现它与 5 辆汽车的生命周期碳排放量相当. Patterson 等[18] 报告说, 训练 GPT-3 模型的碳排放量约为 552 t , 相当于 3 次纽约至旧金山往返航班的碳排放量. 这一发现凸显了深度学习模型对环境的巨大影响. 表 1 是大型深度学习模型的碳排放量与日常活动的排放量之间的对比, 通过表 1, 可以更好地理解深度学习模型的碳排放量.

表1　生活中的碳排放与深度学习产生的碳排放对比[17,19-20]

	活动类型	碳排放/t
生活中的碳排放	单程飞机旅行(纽约至旧金山, 单人)	0.99
	人类生活(平均每年)	5.51
	美国人生活(平均每年)	18.08
	汽车全生命周期, 包括燃料消耗(通常约为10 ～ 15年)	63.00
深度学习模型训练碳排放	Llama 2(34B)	153.90
	Llama 2(70B)	219.42
	用神经网络检索(NAS)训练Transformer	284.02
	GPT-3(175B)	502.00

此外, 研究人员还对不同深度学习模型的碳排放量进行了详细比较. 表 2 展示了几种主要模型在训练过程中的碳排放量[16-18]. 由表 2 可以看出, 不同规模的语言模型在训练过程中所产生的碳排放量存在显著差异. 例如, Transformer_base模型在 P100 GPU 上进行训练, 其训练时间为 12 h 对应的碳排放量为 11.2 kg. 相比之下, GPT-3 模型在 V100 GPU 上训练的时间长达 590.4 h, 导致的碳排放量高达 552.1 t. 这表明随着模型规模的增长, 训练所需的时间和资源也会相应增加, 从而导致的碳排放量增长是巨大的.

表2　深度学习模型训练时长和消耗对比[16-18]

模型	硬件设备	训练小时	碳排放/t
Transformer_base	P100	12.00	0.01
Transformer_big	P100	84.00	0.08
ELMo	P100	336.00	0.13
BERT_base	V100	79.00	1.36
Gshard-600B	TPU v3	74.40	4.80
T5-11B	TPU v3	480.00	46.70
Switch Transformer-1500B	TPU v3	648.00	72.20
GPT-3-175B	V100	590.40	552.10

然而, 值得注意的是, 尽管大模型如 GPT-3 在训练过程中产生了大量的碳排放, 但采用特定的硬件架构能够有效地降低碳排放量. 例如, 1 500B 参数量的 Switch Transformer 模型在 TPU v3 上训练, 虽然训练时间较长 (648 h), 但其碳排放量仅为 72.2 t. 因此, 使用高效能的硬件可以在一定程度上减少碳排放量.

2　低碳型神经网络算法设计

目前, 主流的低碳神经网络算法主要包括剪枝、量化、低秩分解和知识蒸馏等[21-22]. 这些方法通过对神经网络的结构和参数进行精细调整, 实现资源优化. 此外, 新兴方法如参数共享和动态化的网络结构[23]也在模型压缩领域中显示出潜力, 为神经网络的效率提升开辟了新途径. 不仅限于现有模型的优化, 许多研究者还设计了全新的紧凑型模型[24], 这些模型在加快训练和推理过程方面取得了显著成效, 实现了快速训练与推理过程. 接下来, 本文将深入探讨这些主流低碳神经网络算法的原理、实现及其在实际应用中的表现.

2.1　剪　枝

剪枝技术在神经网络中的应用旨在移除冗余或非必要的结构和参数, 以减小模型体积、提升效率, 同时尽量保持性能不变. 剪枝技术分为两类: 非结构化剪枝和结构化剪枝. 非结构化剪枝直接对权重参数进行操作, 不考虑网络结构, 导致稀疏模型的产生; 而结构化剪枝则删除整个通道或层, 保留网络结构的完整性, 同时减少复杂性.

剪枝技术起源于 1989 年, Yann LeCun 在《Optimal Brain Damage》中提出了基于二阶泰勒展开的剪枝方法, 通过评估参数重要性减少模型参数数量, 以提升效率. 该方法利用简化的海森矩阵忽略非对角项, 仅评估单个参数对损失函数的影响. 随后, Hassibi 等[25]提出了改进方法 (OBS), 通过完整的逆海森矩阵捕捉参数间相互作用, 实现更精准的剪枝, 无需重新训练网络. Dong 等[26] 进一步发展了逐层 OBS 算法, 根据各层误差函数的二阶导数进行剪枝研究. 尽管 OBD、OBS 及其变体为剪枝研究奠定了基础, 但其计算海森矩阵的高成本限制了实际应用, 促使研究者探索更高效的策略, 如自适应剪枝和基于规则的剪枝等方法.

Suzuki 等[27] 根据删除单个权重连接对误差的影响来选择需要删除的连接. 在他们的研究中, 首先逐

个删除连接单元并计算删除后对误差的变化, 计算完所有连接单元后, 删除对精度影响最小的单元, 然后用反向传播 (BP) 算法重新训练网络. Han 等[28] 提出根据权重连接的重要性进行剪枝, 如图 1 所示, 可分为 3 步: 首先, 计算出权重连接的重要性值; 其次, 根据设定的阈值, 将低于阈值的权重参数全部删减, 得到比原来稀疏的网络; 然后, 重新训练稀疏网络. 在此基础上, 采用后两个步骤进行反复迭代, 结果不仅节省了内存, 还减少了计算时间, 而且在不影响计算精度的情况下, 提高了计算效率. 由图 2 可以看出, 修剪后的神经元数目显著降低.

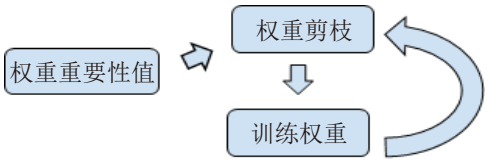

图1 权重连接重要性剪枝步骤[28]

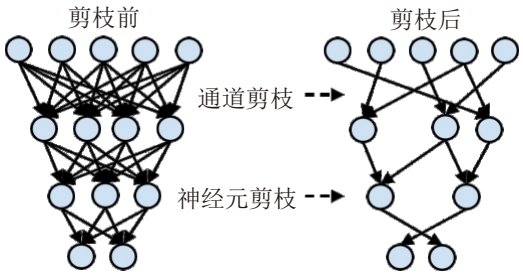

图2 权重连接重要性剪枝结果[28]

非结构化剪枝的优势在于通过对多个参数置零或裁减, 实现更高的稀疏度和精度, 并可根据硬件特性优化速度. 但其劣势在于剪枝耗时较长, 且缺乏硬件与计算库支持时难以发挥作用. 相比之下, 结构化剪枝以通道或滤波器为单位进行剪枝. 剪掉一个滤波器会同时影响其前后特征图, 但保留模型结构完整, 便于利用 GPU 等硬件加速. 结构化剪枝包括通道剪枝和滤波器剪枝两类. 在通道剪枝的研究中, Liu 等[29] 通过对批量归一化层的缩放因子施加稀疏正则化, 使剪枝更平滑且精度损失更少. He 等[30] 采用 LASSO 回归选取最具代表性的通道, 修剪冗余部分, 并利用线性最小二乘法重建剩余通道输出, 在 2 倍加速下精度损失仅为 1.4 % 和 1.0 %. 在滤波器剪枝方面, Wu 等[31] 提出 BlockDrop, 利用强化学习动态选择深度网络的层, 减少计算量而不降低精度. Wang 等[32] 设计了一种结合监督学习与强化学习的门控网络, 能根据前一层激活跳过卷积块, 解决了不可微分跳过决策的问题. Yu 等[33]通过特征排序衡量神经元重要性, 将剪枝转化为二元整数优化问题, 推导出封闭形式的最优解, 用于早期层的神经元剪枝. 随着网络深度的增加, 结构性剪枝更具优势, 可剔除整个卷积核和通路, 大幅压缩模型规模并提升效率. 但其细粒度较粗, 超过临界点会导致显著精度损失, 即使微调也难以恢复. 相比之下, 结构化剪枝配置更灵活, 而非结构化剪枝在细节优化上更精确.

2.2 量 化

在神经网络中, 权重参数通常以 32 位浮点数形式存储. 随着参数数量增加, 存储需求和能耗也随之增长. 根据霍罗威茨 2014 年的研究[34], 每次 32 位浮点加法和乘法分别消耗 0.9 pJ 和 3.7 pJ 的能量. 为减少模型尺寸、节约内存并提升运算速度, 提出了模型量化技术. 量化通过降低网络参数的位宽 (如从 32 位降至 8 位), 实现模型压缩.

对于一个范围为 $(x_{\min}, x_{\max})$ 的浮点变量, 需要将其量化到范围 $(0, N_{\text{levels}}-1)$, 其中 $N_{\text{levels}}=256$,精度为 8 位. 需要推导出两个参数: 缩放比例 (s) 和零点 (Z), 它们将浮点值映射到整数. 缩放比例指定量化器的步长和浮点零映射到零点[35-36]. 零点是一个整数, 确保零被正确量化. 这可以保证常规操作 (如零填充等) 不会导致量化的误差. 按照量化的方法可分为线性量化和非线性量化, 其中线性量化较为普遍且可实现, 是较为普遍的量化策略. 其基本原理可用下式表示:

$$Q=\text{round}\left(\frac{r}{s}\right)+Z, \tag{1}$$

$$r=s(Q-Z), \tag{2}$$

$$s=\frac{r_{\max}-r_{\min}}{Q_{\max}-Q_{\min}}, \tag{3}$$

$$Z=Q_{\max}-\text{round}\left(\frac{r_{\max}}{s}\right). \tag{4}$$

其中: Q表示量化后的整数值, r表示通过统计得到的量化前的浮点值, s表示量化的缩放比例, Z表示浮点中的 0 值经量化后对应的整数.

量化技术可按对象分为权重量化、激活值量化、梯度量化等; 按训练阶段分为感知训练量化 (QAT) 和训练后量化 (PTQ). 此外, 还可按其他维度分类, 如根据$r_{\max}$和$r_{\min}$是否关于Z对称, 分为对称量化和非对称量化; 根据量化策略分为固定量化和自适应量化.

在权重量化方面, Courbariaux 等[37] 提出了二值连接网络 (BinaryConnect), 通过符号函数随机将实值权重以 sigmoid 函数概率二值化为1或−1, 在前向和反向传播中使用二值权重进行计算, 而在参数更新阶段仍使用原权重更新, 即

$$w_b=\begin{cases}+1, & \text{probability } p=\text{sigmoid}(w_b);\\ -1, & \text{probability } 1-p.\end{cases} \tag{5}$$

Rastegari 等[38] 提出的 XNOR-Net, 对权重和激活层都进行了二值量化; 这种方法均属于感知训练量化 (ATQ), 即量化与训练同步进行, 网络使用量化值完成训练. 但这种重新训练需要数百个 epoch 来恢复准确性, 尤其是低位量化模型.

许多学者研究了基于已训练模型的量化方法, 即后量化 (PTQ), 作为昂贵的训练后量化 (ATQ) 方法的替代方案. PTQ 仅对权重进行量化和调整, 无需微调或重新训练, 但因量化在训练后进行, 其准确性和效率通常不及 ATQ. AdaRound[39] 是该方向的一项重要研究, 指出朴素的“舍入至最近值”方法会导致次优结果, 并提出一种通过约束权重变动范围 (±1) 来减小误差的改进算法. 总体而言, ATQ 在准确性和效率上表现更优, 但需更多计算资源和调试; PTQ 则灵活适用于已训练模型, 但可能无法达到最佳效果. 具体选择取决于应用场景和需求.

2.3　低秩分解

低秩分解的核心是将原始模型的参数矩阵分解为多个低秩矩阵的乘积, 从而减少参数数量并降低模型复杂度. 在低秩分解中, 一个$m\times n$维、秩为r的权重矩阵A被低秩近似方法替换为更小维度矩阵的乘积. 这样不仅减少了参数量, 还提高了模型的效率. 常见的低秩分解方法包括奇异值分解 (SVD)、主成分分析 (PCA) 和 Tucker 分解[24]. 其中, SVD 和 PCA 主要用于矩阵分解, 而 Tucker 分解则适用于张量分解.

矩阵分解中, 最广泛使用的方法就是奇异值分解 (SVD). 对于任何给定的矩阵$A\in\mathbb{R}^{m\times n}$, 可以找到一个矩阵满足

$$A=U'S'V'^{\mathrm{T}}. \tag{6}$$

其中: $U'\in\mathbb{R}^{m\times r}$, $V'^{\mathrm{T}}\in\mathbb{R}^{r\times n}$是正交矩阵, $S'\in\mathbb{R}^{r\times r}$是只包含$A$的奇异值的对角线矩阵. 在奇异值分解中, 空间复杂度可以从$O(mn)$降低到$O(r(m+n+1))$. 类似地, 可以用较小的k替换r, 称为截断 SVD. 早期就有学者[40] 在多层感知机 (MLP) 上使用奇异值分解对权重矩阵进行低秩分解, 以替换反向传播算法, 可以有效解决反向传播算法易出现的过拟合现象. 近年来, 奇异值分解被作用到卷积神经网络 CNN 中. Shim 等[41] 对 CNN 的 softmax 层权重矩阵使用 SVD 进行分解以减少 softmax 的计算成本; Zhang 等[42] 则引入基于一种 SVD 的低秩约束, 将非线性响应的重构误差最小化, 这样能够加速包括非线性层的整个卷积神经网络.

矩阵的低秩分解虽然能优化神经网络的空间和计算复杂度, 但是它的二维性质限制了极端压缩的可能. 在张量语言中, 一个二维矩阵可看作一个二阶张量, 如果张量的阶数或每个阶内的模数值超过 2(如三维张量), 则可以考虑更灵活的算法来实现极高的压缩比. 同样的, 张量也可以进行低秩分解. 首先, 张量和矩阵之间的运算称为缩约积 (mode i contracted product), 即对于d阶张量$\mathcal{A}\in\mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$和矩阵$B\in\mathbb{R}^{n_i\times m}$ $(i\in\{1,2,\ldots,d\})$, $\mathcal{A}$和B的乘积为$R=\mathcal{A}\times_i B$, 其中R也是$\mathbb{R}^{n_1\times\cdots\times n_{i-1}\times m\times n_{i+1}\times\cdots\times n_d}$的$d$阶张量. 在此前提下, d阶张量$X\in\mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$可分解为一个核张量$K\in\mathbb{R}^{r_1\times r_2\times\cdots\times r_d}$和$d$因子矩阵$F^{(i)}\in\mathbb{R}^{r_i\times n_i}(i\in\{1,2,\ldots,d\})$, 其中$r_i$称为$i$-秩, 它实际上是通过将张量重塑为$X_{(i)}\in\mathbb{R}^{n_i\times n_1\cdots n_{i-1}n_{i+1}\cdots n_d}$生成的矩阵的秩. 核张量与原始张量的阶数相同, 即d. 若继续利用模式i契约乘积, X可以改写为

$$X\approx K\times_1 F^{(1)}\times_2 F^{(2)}\times_3\ldots\times_d F^{(d)}. \tag{7}$$

这种分解形式称为 Tucker 分解[43-44]. 还有一种常用的张量分解方法, 称为 CP 分解[45], 表示将张量因式分解为一系列秩一张量的总和, 因此可以看作 Tucker 分解的特殊形式. 综合来看, 在张量低秩分解中, Tucker 分解能精确捕捉高维数据的结构, 但计算成本高且参数量大[43]. 而 CP 分解实现简单、适于降维, 但结果可能不唯一且易受局部最优的影响[45]. 因此, 在真实的应用场景中, 若简单数据可以选用 CP 分解方法, 而复杂数据建议用 Tucker, 则要留意计算和过拟合问题.

近年来, 随着深度学习技术的快速发展, 低秩分解方法逐渐被应用于更复杂的神经网络结构中. 例如, Swaminathan 等[46] 提出了稀疏低秩 (SLR) 方法, 通过对 SVD 矩阵稀疏化并降低不重要神经元的秩, 实现了更优的模型压缩. 总体而言, 低秩分解为深度学习提供了一种有效策略, 不仅降低了模型复杂度, 还能在某些情况下提升性能, 实现效率与性能的平衡.

2.4　知识蒸馏

知识蒸馏通过将大型神经网络模型 (教师模型) 学习到的知识传递给更小的神经网络模型 (学生模型), 在保持学生模型轻量化的同时尽量保留其泛化能力. 教师模型可为单一模型或多个模型的集合. 知识蒸馏在低碳计算中尤为重要, 因轻量级学生模型能显著减少推理时间和能耗, 特别适合边缘和移动设备. 此外, 训练轻量模型需更少资源, 有助于降低碳排放, 符合低碳算法目标.

2006 年, Buciluǎ提出了早期类似知识蒸馏的思路[47]: 利用大模型对未标记数据进行预测, 并以此训练较小模型, 从而近似复杂模型的学习函数. Ba 等[48]扩展了这一思路, 通过最小化教师模型和学生模型的logits(softmax 前的输出) 间的平方差, 实现学生模型对教师内部表示的学习. Hinton 等[49]进一步系统化了这一方法, 提出使用温度参数T生成软目标 (Soft Targets), 并结合教师软目标、硬目标与学生模型的预测结果优化损失函数, 增强学生模型的泛化能力. 假设p_i表示教师模型对第i个样本的预测概率, q_i表示学生模型对该样本的预测概率, 则损失函数为

$$L_{\text{distill}} = \alpha T^2 \sum_{i=1}^{N} \text{KL}\left(\frac{p_i}{T}, \frac{q_i}{T}\right). \tag{8}$$

其中: N表示样本总数, T表示温度参数, α表示知识损失的权重因子. 如果$T=1$, 则它就是正常的softmax 函数; 更高的T值代表更多的信息, 相当于增加了蒸馏的“软性”程度. KL(Kullback-Leibler) 表示散度, 定义为

$$\text{KL}(p||q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}. \tag{9}$$

需要注意的是, 知识蒸馏不同于迁移学习. 知识蒸馏的目标是提供更小的模型来解决与更大的模型相同的任务[49]; 而迁移学习的目标是减少求解与其他模型求解任务相似模型的训练时间[50]. 知识蒸馏通过改变被训练模型 (学生模型) 的损失函数来考虑预训练模型 (教师模型) 隐含层的输出来实现其目标, 而迁移学习通过学习预训练模型的参数来初始化模型实现其目标.

2.5 参数共享

神经网络模型参数过多是一个常见问题, 随着模型规模增大, 不必要的参数导致资源浪费. 对于特定任务, 模型往往只需少量参数即可达到与使用全部参数相同的推断效果[51]. 参数共享是一种有效解决过参数化问题的策略[52-53], 其核心思想是通过映射网络参数到少量数据上, 显著减少存储需求并提高计算效率. 此外, 参数共享还能缓解过拟合问题, 提高模型泛化能力. 常见方法包括循环矩阵和聚类共享[52-54]. 循环矩阵用一个向量生成权重矩阵, 将全连接层参数从$O(n^2)$降低到$O(n)$, 广泛用于参数共享. 聚类共享则通过聚类算法对参数分组, 在每组内共享一个代表性参数, 显著减少参数数量, 同时较好地保持性能. 参数共享技术可有效降低模型复杂性, 减少计算资源需求, 实现减少能耗和碳排放的目标, 符合低碳算法研究方向. 具体来说, 神经网络模型各网络层之间使用的是非线性变换, 即

$$f(x) = \sigma(Mx). \tag{10}$$

其中: x是输入张量, $\sigma(\cdot)$是非线性变换, M是大小为$m \times n$的可学习矩阵. 此时一个直观的模型压缩方法是使用一个大小为$m \times n$但是参数量小于mn的矩阵来替代传统矩阵M. 作为结构矩阵的一种, 循环矩阵是具有循环对称性质的特殊矩阵, 对于张量$\mathcal{H} = \{h_0, h_1, \ldots, h_{d-1}\}$, 循环矩阵的每一行都由上一行向右平移一格得到, 即

$$\mathcal{M} = \begin{bmatrix} h_0 & h_1 & \ldots & h_{d-1} \\ h_{d-1} & h_0 & \ldots & h_{d-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \ldots & h_0 \end{bmatrix}. \tag{11}$$

Cheng 等[53] 将卷积神经网络中的全连接映射优化为循环映射, 时间复杂度从$O(d^2)$降至$O(d\log d)$, 空间复杂度从$O(d^2)$降至$O(d)$. Wang 等[54] 设计了一种在傅里叶域中为对角矩阵的循环矩阵, 从而显著降低了训练复杂度和存储成本. 这种方法适用于资源受限环境 (如移动设备), 但在模型压缩后如何平衡性能和效率仍是研究重点.

另一种参数共享方法是聚类共享. Chen 等[52]利用哈希函数将权重随机分组到哈希桶中, 同一桶内的权重共享参数, 并在训练中调整这些参数. Wu 等[55] 则通过 k 均值聚类, 使相似权重共享一个中心值, 从而压缩模型. 然而, 尽管参数共享方法简单高效, 其泛化性和可解释性不足, 仍需进一步优化以提升模型性能和理论支撑.

2.6 其他 —— 轻量化神经网络模型设计

轻量化神经网络通过从头设计轻量级模型, 实现模型压缩和高效计算. 这种设计倾向于减少卷积核数量、缩小卷积核尺寸、降低网络深度, 同时采用深度可分离卷积、组卷积等特殊结构来进一步降低计算成本. 尽管两者在实现方式和应用场景上不同, 但目标均是构建高效、低资源消耗的神经网络模型.

在紧凑神经网络模型中, 深度可分离卷积核[56]技术十分常见, 主要思想是将标准卷积操作分为两个步骤: 深度卷积和逐点卷积, 这种结构可以显著减少参数量和计算量, 从而实现更轻量级的模型. 而Szegedy 等设计的 inception-v3[57] 通过将一个$n \times n$的卷积核分解成$n \times 1$和$1 \times n$的非对称卷积核, 构建了不到 2 500 万个参数的 CNN 模型; Xception[58] 网络扩展了深度可分离卷积核, 效果超越 Inception-V3 网络; MobileNets[59] 除了使用深度可分离卷积构建之外, 还引入了两个超参数, 宽度乘数α和应用于

输入图像的分辨率参数 p, 将计算成本降低了 p^2 个数量级.

由 Iandola 等提出的 SqueezeNet[60] 是一个基于 CNN 的轻量化神经网络模型, 使用瓶颈方法来设计一个非常小的网络. 它的参数比 AlexNet 少 50 倍, 但在 ImageNet 数据集上与 AlexNet 达到了相同的精度. 此外, 作者还使用深度压缩[61] 技术将 SqueezeNet 压缩至 0.5 MB, 大小比原始 AlexNet 小 510 倍. 在另一项研究中, Iandola 等[62] 引入了 DenseNet, 其包含多个紧凑的密集结构, 网络的每一层都连接到每个交替层, 因此可以减少参数数量、特征重用和梯度消失问题. 在进一步的研究中, Huang 等[63] 又提出了 CondenseNet, 它只使用 1/10 的计算量却达到了与 DenseNet 相似的精度. 综上所述, 研究者们针对深度学习的轻量化模型设计取得了较为显著的进展. 在追求低碳算法的大背景下, 这些轻量化模型不仅优化了计算资源的使用, 更为在有限能源环境中部署深度神经网络模型提供了强大支持, 有助于推动算法的可持续发展.

2.7　低碳算法发展趋势探讨

本章系统性地介绍了几种常见的模型压缩方法, 包括剪枝、量化、低秩分解、知识蒸馏、参数共享以及从头设计的轻量化模型, 并在表 3 中对这些方法进行了详细的对比分析. 值得强调的是, 随着计算资源日益紧张和技术发展的不断推进, 低碳算法已成为提升系统性能和资源利用效率的核心手段之一. 这类算法不仅极大地提高了计算效率, 还通过合理分配和调度计算资源, 有效降低了能耗, 进一步推动了系统的可持续发展. 尤其在当前绿色计算和低碳经济的大趋势下, 低碳算法在减少碳排放方面发挥了至关重要的作用. 因此, 以模型压缩技术为核心的低碳算法不仅是应对大规模模型训练和部署成本的关键方案, 更是推动绿色技术创新、实现碳中和目标的有效路径.

表3　不同的模型压缩方法对比

压缩方法	压缩原理	优点	缺点	使用场景	分类
剪枝	去除神经网络中冗余连接和参数, 减小模型大小和计算量	显著降低模型计算和存储需求; 可保留主要连接和参数, 减小精度损失; 模型仍可保持较高的性能	剪枝过程需要额外的计算和优化步骤; 对模型结构的依赖性较强, 不适用于所有模型和任务	深度神经网络	结构化剪枝、非结构化剪枝
量化	减少模型中的参数表示的比特数, 降低模型存储和计算开销	可显著减小模型的存储空间和计算量; 可以提高模型的推理速度	量化精度降低可能导致模型性能下降; 在较高压缩比下, 会出现显著性能损失	深度神经网络	对称量化、非对称量化
低秩分解	将权重矩阵分解为较低秩的近似矩阵, 减小模型参数量和计算复杂度	具有较好泛化能力	分解过程需要额外计算和优化步骤; 选择适当的秩数和分解方法是一个挑战	深度神经网络	CP分解、Tucker分解
知识蒸馏	使用大型模型的知识指导训练小型模型, 提高模型的性能	通过大型模型的知识指导训练小型模型提高性能; 蒸馏后模型更易部署	需要大型模型作为知识源, 模型间依赖性强; 性能的提升有一定限制	大型预训练模型和小模型	基于模型、样本的知识蒸馏
参数共享	将全部参数映射到一小部分参数上, 降低模型计算复杂度	操作简单, 易于实现, 且能够与其他方式进行组合	泛化性能差, 可解释性差, 理论依据不够充分	深度神经网络	循环矩阵、聚类共享

为了深入探究国内模型压缩领域的发展态势, 本研究通过在知网中以关键“模型压缩”“压缩模型”“轻量化神经网络”进行检索, 并从结果中筛选出 211 篇与主题相关的文献进行了详细分析.

通过关键词共现图 (图 3) 可见, 在模型压缩的关键词下, 与模型压缩相关的技术受到学者们的关注, 如知识蒸馏、网络剪枝、模型剪枝和参数量化等. 此外, 下游任务中的“目标检测”“人脸识别”“图像识别”也引起了相当大的关注. 从聚类结果来看, 模型压缩话题下, 更多的研究集中在各种方法的研究, 如量化、师生框架、张量分解、级联融合等. 然而, 一些学者已经开始将网络剪枝和知识蒸馏等方向应用于下游任务, 例如场景分类、图像分类和人脸识别. 另外, 在与深度学习相关的关键词中, 涉及到压缩感知和差值编码等对深度学习模型的更新迭代方向, 可看出学者们对深度学习模型轻量化发展的展望.

由图 4 还可以观察到, 模型压缩的研究已经开始向下游任务的应用方向转变. 例如, 在目标检测、人脸识别和图像识别等领域, 学者们将压缩技术应用于实际场景中, 以提高模型在特定任务上的性能和效率. 然而, 尽管模型压缩领域取得了显著的进展, 但仍面临一些挑战. 总的来说, 模型压缩作为深度学习领域的重要研究方向, 在提高模型效率和资源利用方面具有巨大潜力. 未来的研究可以进一步探索

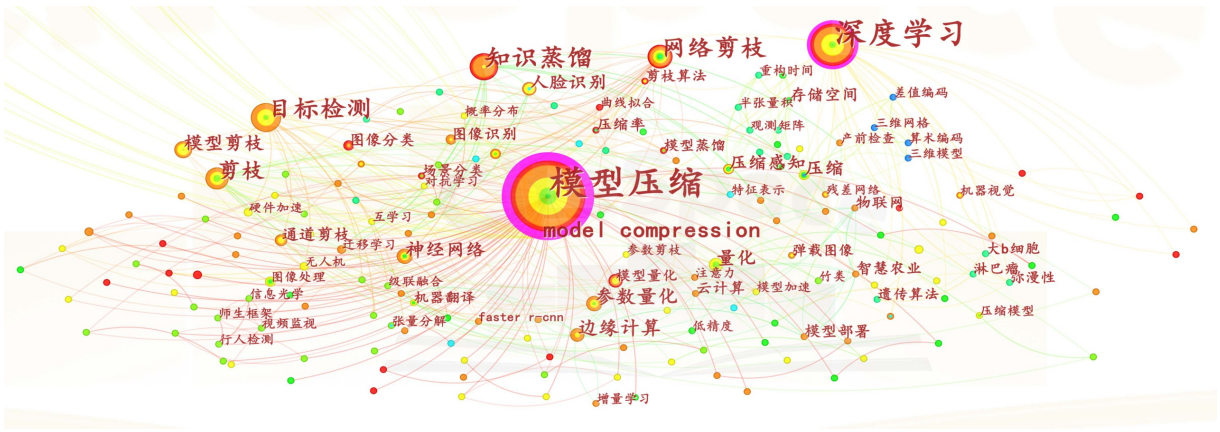

图3 关键词共现图

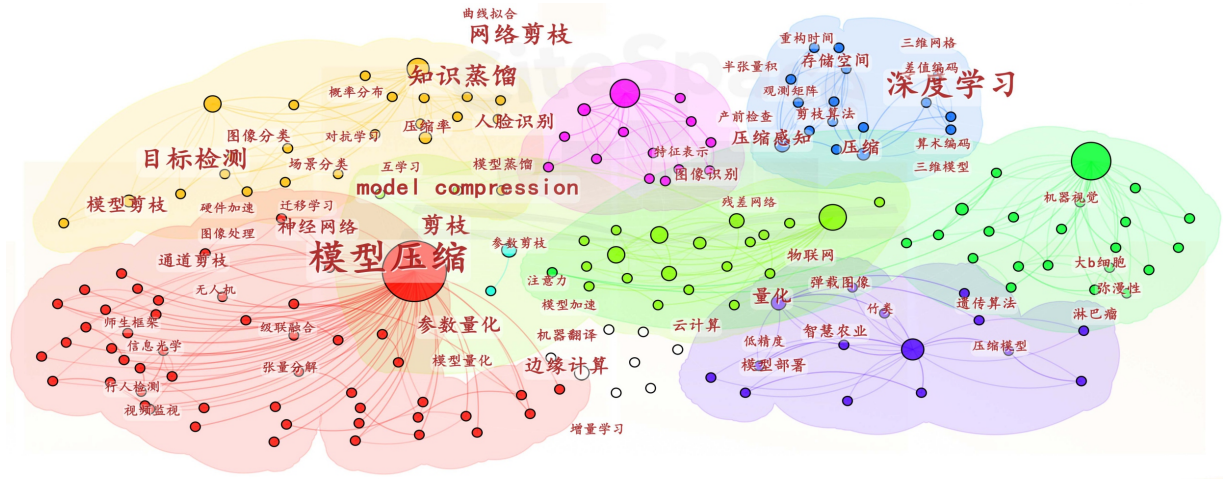

图4 关键词共现和聚类图

新的压缩技术和方法, 结合领域特定的任务需求, 以实现更高效、更精确的模型压缩, 从而推动深度学习技术在更广泛的应用场景中的实用性和普及性. 我们相信, 随着技术的不断发展和突破, 模型压缩将在推动人工智能技术方面发挥重要作用.

综上所述, 本节继续深入探究国内模型压缩领域的发展态势, 关键词共现网络视图、时间突现图以及聚类分析结果共同揭示了该领域的研究趋势. 从这些图表中可以看到, 模型压缩作为一个热门的研究领域, 吸引了大量学者的关注和投入. 随着深度学习的快速发展, 以模型压缩为核心的低碳算法已经成为提高模型效率和部署性能的关键技术.

3 工业界低碳算法设计

算法复杂性的增加对硬件设备提出了更高要求. 为了实现高能效和高性能而不牺牲精度, 业界正在寻求优化的硬件架构设计. 本章重点介绍工业界在低碳算法设计方面的努力与尝试, 分析通过降低运算精度来实现节能效果和提高计算效率的方法, 详细介绍专为低碳算法优化的硬件设计技术, 强调创新结构 (如新型材料和设计方法) 如何助力实现低碳目标. 随后讨论如何利用专门的硬件加速器进一步提高运行效率并显著节能. 最后展示低碳算法在工业领域的具体应用案例及其实际效益. 本章旨在全面介绍工业界在低碳算法设计领域的最新研究成果和实践经验, 并展望这些技术如何推动工业的可持续发展.

3.1 低精度运算

深度学习模型通过学习每层网络的权值和偏差来拟合各种任务的结果. 在训练过程中, 这些张量都是 32 位浮点数的精度. 在推理阶段模型不需要进行反向传播, 可以将数据精度降为 FP16 或者 FP8 来进行模型轻量化并提升推理速度[64].

FP16 是一种半精度浮点数类型. 在双精度 64 位格式下, 1 位表示数字正负, 指数由 11 位构成, 剩下 52 位表示该数字 (有效位数), 能够表示的数字范围和大小十分广阔. 在单精度 32 位格式中, 1 位用于指示数字为正数还是负数, 指数保留了 8 位, 其余 23 位表示数字. 半精度 (16bits) 表示范围则更小, 由 1 位符号位、5 位指数位和 10 位有效位数组成.

通常情况下, 生成半精度 (FP16) 模型只需截断

单精度模型的权重, 并在推理过程中使用特定硬件处理. 例如, NVIDIA 从 Volta 架构开始采用 Tensor Core 加速器, 当检测到 FP16 推理时, TensorRT 会自动使用 Tensor Core 进行半精度计算. 而 HFP8 设计是一种混合精度格式, 在前向传播中, HFP8 使用 1-4-3 格式 (1 位符号位, 4 位指数位, 3 位有效数位), 而在反向传播中使用 1-5-2 格式 (1 位符号位, 5 位指数位, 2 位有效数位). 由于深度学习中的权重和激活值范围较小且对精度要求较高, FP8(1-4-3) 多用于表示这些值; 而梯度值可能非常大或非常小, FP8(1-5-2) 提供了更大的范围, 因此多用于表示梯度. 这种设计使 HFP8 在训练中接近 FP32 的表现.

3.2 硬件设计

在深度神经网络模型中, 乘法累加 (MAC) 运算是最重要的计算. 而 MAC 是可以被并行执行的, 因此, 为了实现卓越的性能, DNN 加速通常使用支持高度并行计算的硬件加速器实现. 随着深度神经网络模型的不断发展, 越来越多的研究专注于神经网络模型在硬件架构上的加速运行, 从通用架构 GPU, 到现场可编程门阵 (FPGA)、专用集成电路 (ASIC). FPGA 具有可编程的硬件资源和灵活的连接性, 可为特定的计算任务进行定制优化.

在通用架构中, 中央处理单元 (CPU) 是一种广泛部署在各种边缘设备上的处理芯片, 如笔记本电脑、手机上的处理芯片, 它的优点是可以使用超线程进行上下文切换, 但内核数量限制了它并行处理大量数据的能力. 因此, DNN 的加速需要专门的硬件设计来实现. 例如, 在图形处理单元 (GPU) 中, Vasudevan 等[65] 提出了支持浮点矩阵计算的架构. 现场可编程门阵列 (FPGA) 与 GPU 相似, 都支持并行计算, 但其与 GPU 不同之处在于它可以被重新编程以执行各种任务, 因此可以更灵活地适应不同的应用场景. Cloutier 等[66] 提出了基于 FPGA 的虚拟图像处理器, 它是使用大型 FPGA 构建的 SIMD 多处理器, 可适用于多种神经网络算法; Han 等[67] 基于 FPGA 的加速器提出了高效语音识别引擎 (ESE), 实现了语音识别的 LSTM 算法; Wang 等[68] 提出了用于对象检测的可重构 YOLOv3 FPGA 硬件加速器.

专用集成电路 (ASIC) 是加速深度神经网络 (DNN) 的重要硬件架构, 具有体积小、功耗低、速度快的优势. NeuFlow[69] 是基于 ASIC 的 CNN 加速器, 其 luaFlow 编译器将 Torch5 中的高级数据流图转为 Neuflow 机器编码, 达到 320 GOPS 的吞吐量, 功耗仅 0.6 W. Chen 等[70] 提出的 DianNao 加速器实现了 CNN 和 DNN 的高效推理. 随后, 2014 年提出的 DaDianNao[71] 和 2017 年提出的 DaDianNao 改进版[72], 通过大量片上内存存储权重, 与 GPU 相比在相同任务下加速 450.65 倍, 能耗降低 150.31 倍.

3.3 硬件加速

CPU 计算能力主要取决于核心数、频率和指令集等因素. 以 Intel Xeon Platinum 8 163 为例, 其核心数为24, 单核频率2.5 GHz, 支持 AVX-512 指令集, 包含两个 AVX-512 FMA, 理论双精度浮点性能为

$$24 \times 2.5\,\text{GHz} \times 32\,\text{FLOPs/cycle} = 1.9\,\text{TFLOPs}.$$

GPU 计算能力与核数和频率相关, 通常通过 CUDA 进行编程, CUDA 中的 grids、blocks 和 threads 结构支持大规模并行计算. 例如, Tesla V100 拥有 5 120个 CUDA 核心, 频率1.37 GHz, 理论双精度性能为

$$5\,120 \times 1.37\,\text{GHz} \times 1\,\text{FLOP/cycle} = 7\,\text{TFLOPs}.$$

V100 的 Tensor Core 可额外提供 112TFLOPs 的性能. TPU(tensor processing unit) 专为深度学习优化, 核心硬件为 Matrix Multiply Unit, 用于加速卷积和矩阵运算. TPU 性能计算公式为

$$\text{TOPS} = \text{运算单元} \times 2 \times \text{主频}.$$

以 TPU1 为例, 运行频率 700 MHz, 包含 65 536 个 8-bit 运算单元, 其算力为 91.75 TOPS.

3.4 应用案例

案例 1 谷歌 Evolved Transformer 在机器翻译领域的应用. 2019 年谷歌提出 Evolved Transformer[73], 通过神经架构搜索技术自动优化网络结构. 在机器翻译领域, 谷歌团队用 Evolved Transformer 替换原有的 Transformer 模型, 结果显示翻译准确率略有提升, 同时浮点运算数量 (FLOPS) 减少 1.6 倍, 总 CO_2e 和净 CO_2e 均减少 1.2 至 1.3 倍. 这显著降低了计算成本和环境影响, 同时保持了性能.

案例 2 谷歌 Gshard 和 Switch Transformer 在 NLP 任务中的应用. 2020 年谷歌提出稀疏激活的专家混合模型 Gshard[74] 和 Switch Transformer[75], 用于 NLP 任务 (如文本分类、情感分析等). 相比密集模型 (如 GPT-3), Gshard 使用的能源减少 55 倍, 总 CO_2e 减少 115 倍, 净 CO_2e 减少 130 倍; Switch Transformer 使用能源减少 7.2 倍, 总 CO_2e 减少 7.7 倍, 净 CO_2e 减少 9.4 倍. 在提升性能的同时, 大幅降低了计算成本和环境影响.

4 大模型低碳算法设计

大语言模型 (LLM) 的出现深刻影响了人工智能的发展, 当前百亿、千亿参数级别的模型已屡见不

鲜. 然而, 这些模型由于规模庞大, 直接运行需要极高的算力资源. 例如, FP16 格式的 GPT-3(175B 参数)加载权重需约 350 GB 显存, 远超单张 GPU 容量, 即便中小型多 GPU 服务器也难以支持. 此外, 训练大型模型消耗巨大资源, 训练周期长达数天或数周, 并带来高昂的电力与冷却成本. 为降低 LLM 的训练与部署门槛, 研究者提出了多种压缩与优化方法. 这里分别从训练后压缩、训练时压缩、模型架构和算法生态这 4 个方向分别对 LLM 压缩与节能相关的研究进展做简要介绍.

4.1 LLMs 的训练时压缩

由于 LLM 依然基于传统神经网络架构, 而大规模神经网络的梯度下降和反向传播运算更需要巨大的计算资源. 为了降低 LLM 的训练门槛, 许多研究机构提出了自己的 LLM 训练时压缩方案.

1) 常规 Adaptation: Adaptation(适配) 指在模型内添加具有一定结构的额外的 adapter 模块或层, 微调时只微调这些添加层的参数, 并冻结原本的模型参数, 以此达到节约模型训练所需硬件资源的目的. 在 LLM 领域, 上海人工智能实验室相继提出对“羊驼”LLaMA 类模型进行多模态指令微调 Adaptation 方法: LLaMA-Adapter[76](V1) 和 LLaMA-Adapter [77](V2). V1 版本相比基于全参数微调的 Alpaca-7B 模型, 只需要更新 1.2 M 参数权重, 便可以在科学问答任务上超越许多更大尺寸的模型, 同时也使模型具有一定的视觉问答能力. V2 版本则解锁了更多的可学习参数, 着重提高了其在零样本和开放场景下的视觉-文本理解能力. 文章指出, 基于开源基座纯文本 LLM 模型只需要在较少量视觉-文本数据集上进行仅 10 M 参数的指令微调, 便可以超越许多需要大量多模态数据预训练加全参数微调的普通尺寸深度模型的多模态理解和泛化能力.

2) LoRA[78]: LoRA(low-rank adaptation) 是由微软提出的一种用于 LLM 领域的基于低秩分解的特殊 Adaptation 方法. LoRA 先以朴素视角回顾了模型微调: 在预训练模型的原始参数权重W_{ori}上通过微调获得一个偏移权重ΔW, 二者的矢量和组成了微调后的下游模型W_{FT}, 即$W_{\text{FT}} = W_{\text{ori}} + \Delta W$. 尽管完整模型的参数尺寸巨大, 但偏移权重$\Delta W$可被近似分解为两个低秩子矩阵的矩阵乘积形式: $\Delta W = BA$, 且两个低秩子矩阵A、B的中间维度远小于模型的本身维度. 在微调阶段只更新A、B的权重, 微调完成后再与原矩阵直接相加获得最终权重, 以此大幅降低微调阶段需更新的模型参数量. 与一般的 Adaptation 方法相比, LoRA 在推理阶段对模型的所有层均使用简单的加法结构实现与 adapter 的融合, 没有引入额外的传播结构, 因此避免了一般 adapter 会产生的额外延迟.

4.2 LLMs 的训练后压缩

LLMs 训练时压缩重于 LLM 模型的训练过程, 在推理阶段的计算资源消耗并未涉及. LLMs 的训练后压缩旨在减少 LLM 的推理运算成本, 降低 LLM 部署与应用的能耗和门槛. 下面分别介绍基于剪枝与量化技术路线的若干篇代表工作. 在剪枝方面, SparseGPT[79] 的工作原理是将剪枝问题简化为大规模的稀疏回归实例. 该方案基于新的近似稀疏回归求解器, 用于解决分层压缩问题, 其效率可以满足在单个 GPU 上消耗数小时完成对 GPT-175B 的剪枝. Wanda[80] 由两个组件构成 —— 剪枝度量和剪枝粒度, 该方法在 65B 参数量的大模型上仅需 5.6 s 即可完成剪枝, 并达到与 SparseGPT 相近的效果.

在量化方面, SmoothQuant[81] 分析了不同尺寸 OPT 类模型的量化情况, 并着重在 LLM 规模增加(至多 175B) 的情景下考察量化性能. 为避免传统通道量化策略处理 token 级值域变化过大面临的硬件运行低效困境, 提出引入一对用于均衡权重和激活值量化难度的平滑超参. GPT-Q[82] 是一种利用二阶 Hessian 信息的针对 LLM 特别优化的训练后量化方案. 其算法原理来自 OBQ[83], 但原始算法的复杂度过高, 于是作者针对 LLM 的量化场景做了 3 点改进: 取消在 OBQ 中使用的贪心算法, 大幅降低了运算复杂度; 使用权重更新批处理策略, 减少由 GPU 内存带宽瓶颈带来的通信延迟; 使用加入正则的 Hessian 矩阵, 提高收敛稳定性.

4.3 LLMs 的模型架构优化

当前的大型语言模型 (LLM), 如 GPT-4, 大多基于 Transformer 架构. Transformer 注意力机制的运算复杂度与序列长度的平方呈正比, 而 LLM 在应用过程中经常面临长文本输出的情景. 因此, 如果能对模型架构进行升级, 从而降低计算开销, 同时保持甚至提升模型性能, 达到压缩优化的目的, 这里对近期出现的两种模型架构做简要介绍.

1) RWKV[84]: RWKV(receptance weighted key value) 是 2023 年由国内外 20 余家研究机构共 30 名学者共同合作发表提出的一种全新模型架构. 与一般 Transformer 不同的是, RWKV 重新拥抱了传统 RNN 思想, 令模型通过保留先前时刻的隐状态进行自回归式生成. RWKV 首先重构了在 Transformer 中的多

头注意力模块, 将其中大量的矩阵运算替换为点积运算, 有效降低了运算量. 在模型具体结构设计上, RWKV 使用了 Time Mixing 和 Channel Mixing 两个核心组件, 这两个组件均继承 RNN 思想, 通过一系列与 RNN 相似的记忆型结构实现了一种新的注意力机制. 得益于这一设计, RWKV 相比已有的 Transformer 模型及其多种变体, 在模型输入序列长度的时间和空间复杂度上均实现全面更优.

2) RetNet[85]: 传统观点认为, 语言模型存在着一个“不可能三角”, 即训练可并行、推理成本低和良好的性能这三者不可能同时实现. 而近期, 微软与清华大学联合推出了一种新模型架构 RetNet(retentive network), 称该模型似乎打破了这一限制, 是 Transformer 的有力继承者. RetNet 结合了已有的线性注意力模型的思想, 并加入记忆权重系数, 构建了一种新的 Retention 模块来代替 Transformer 的多头注意力. 与 RWKV 的 RNN 思想类似, Retention 模块整体上也通过记录已有的隐状态来摆脱对长序列的线性运算限制. 与一般 RNN 不同的是, 通过记忆权重系数, RetNet 可以将 Retention 运算过程分别表示为类 Transformer 注意力的并行形式和类 RNN 的串行形式, 实现了并行训练与$O(1)$复杂度串行解码的兼得, 除了进行推理效率的实验比较, 其也在下游任务使用 67B 参数量的模型进行了实验.

3)Mamba[86]: Mamba 模型是一种高效的深度学习架构, 它通过创新的状态空间模型和卷积化技术显著提升了计算效率, 同时优化了资源使用. 与传统的 RNN 和 Transformer 相比, Mamba 模型在处理长序列任务时, 不仅计算复杂度更低, 而且能够实现并行计算, 大幅度减少了训练和推理的时间. 此外, Mamba 模型在数据存取交互上的优势, 特别是在 GPU 与 SRAM 之间的快速数据交换, 进一步降低了内存占用和提高了运算速度. 这些特点使得 Mamba 模型在资源受限的环境中具有明显的优势, 有望在计算机视觉等数据密集型领域得到广泛应用.

4.4 LLMs 的低碳算法生态

除了直接对神经网络模型进行压缩优化外, 系统和平台级的算法优化在大型语言模型 (LLM) 压缩中同样重要. 子图融合方法通过合并冗余算子减少 GPU 调用, 从而降低模型开销. 典型方法包括 NVIDIA 的 FasterTransformer 和微软的 DeepSpeed. 为优化 attention 模块, FlashAttention 采用 tiling 机制重构注意力模块, 将输入分块, 并递增执行 softmax 操作, 减少输入访问, 同时通过存储前向归一化因子加速反向计算, 提高运算速度. vLLM 框架则通过 PagedAttention 技术, 将键值缓存划分为块, 允许非连续内存存储连续数据, 显著减少显存占用, 提高理论并行吞吐量 2 到 3 倍. 此外, 商汤科技与北航推出的 LightLLM 框架具有以下特点: 1) 使用三进程架构异步处理 tokenize 和 detokenize 操作, 避免 CPU 处理阻碍 GPU 调度; 2) 提出基于 token 的高效键值缓存管理方案; 3) 通过高效路由结合 token attention 精准管理推理请求, 大幅提升吞吐量, 特别适用于长度差异较大的任务.

综上, LLM 的压缩优化需结合多种技术, 从模型基础到工程部署进行全面优化, 不同方向技术相互促进 (如量化与 LoRA 结合得到 QLoRA), 并在实际应用中根据具体需求精细化调研和优化实现最佳效果.

5 对比典型低碳算法的效果

本节将深入探讨并对比几种典型的低碳算法在各种任务中的表现. 为了评估效果, 以 GLUE 基准测试中的 8 项任务作为标准进行比较.

5.1 实验数据集

实验数据集来自 GLUE 基准测试[87]. GLUE 是一项多任务基准测试, 包括 8 个不同的任务, 用以评估和理解不同的模型在各种自然语言处理任务上的表现. 这些任务包括: CoLA(语言可接受性的语料库)[88], MNLI(多种类型的自然语言推理)[89], SST-2(斯坦福情感树库)[90], QNLI(问答自然语言推理)[91], MRPC(微软研究人类对比语料库)[92], QQP(Quora 问题对)[93], RTE(识别文本蕴含)[94], 以及 STS-B(语义文本相似性基准)[95]. 在评估过程中, 本文根据之前的研究, 选取了各任务的主要评估指标. 具体来说, 报告了 MNLI、SST-2、QNLI、QQP 和 RTE 的准确度, MRPC 的 F1 分数, CoLA 的 Matthews 相关系数, 以及 STS-B 的 Spearman 等级相关系数. 这些评估指标可以全面反映模型在各项任务上的性能, 有助于我们更好地理解和对比不同低碳算法.

5.2 实验对比方法

本文比较了经典与最新的知识蒸馏和剪枝方法. 在知识蒸馏部分, 报告了传统方法及教师-学生模型的结果, 同时展示了预训练蒸馏模型 TinyBERT[96] 的性能. 对于基于关系的蒸馏方法, CKD[97] 利用了成对距离和三元角度建模水平与垂直方向的令牌表示关系, Liu 等[98] 在此基础上从多粒度表示中提取结构关系, 并层次化蒸馏多粒度知识. AD-KD[99] 从归因角度揭示教师推理行为, 用以传递数据特定知识. 此外,

本文选取典型模型进行实验对比分析. MetaDistil[100]关注学生在验证集上的泛化能力, 并通过指导教师学习优化泛化效果. LGTM[101]提出蒸馏影响评估, 根据训练样本对学生性能的影响为样本分配损失权重.

在剪枝方法中, Magnitude[102]基于零阶信息, 通过权重的绝对值衡量重要性进行剪枝. 然而, 预训练语言模型 (PLM) 的权重通常已固定, 限制了其适应性. 为此, Movement[103]利用一阶信息, 根据训练过程中权重的变化而非绝对值选择权重, 并通过逐步增加稀疏性, 使剪枝与微调同步完成, 从而适应下游任务. Static Model Pruning (SMP)[104]提出仅使用一阶剪枝即可使 PLM 适应下游任务, 无需微调, 同时实现目标稀疏性. L_0正则化[105]则通过重参数化和L_0惩罚来训练掩码变量. PLATON[106]采用敏感性变体, 使用指数移动平均重新加权不确定性. ContrAstive Pruning (CAP)[107]在预训练和微调阶段分别学习任务不可知和任务特定知识. PINS[108]提出一种高效的自我调整方法, 将最优剪枝决策与整数线性规划问题等价, 提高模型的泛化能力.

5.3 实验效果对比

主要结果如表 4 所示, 该表格清晰地展示了各种知识蒸馏方法在不同数据集上的表现. 通过对表格内容的仔细分析发现, AD-KD[99]在接近一半的数据集上表现优于所有的基线方法, 表明 AD-KD 方法在知识蒸馏领域具有强大的性能. 而 LGTM[101]和 MGSKD[98]的整体性能也不容忽视, 它们在许多数据集上超过了许多基线方法. 这些结果表明, 不同的知识蒸馏方法在不同的数据集上会有不同的表现, 原因可能是由于各种方法针对不同类型的数据和任务有不同的优化策略. 因此, 选择合适的知识蒸馏方法应根据具体任务需求来进行.

表4 经典低碳绿色算法在 GLUE 数据集上的效果表现

Method	Sparsity	#Param.	CoLA (Mcc)	MNLI-(m/mm) (Acc)	SST-2 (Acc)	QNLI (Acc)	MRPC (F1)	QQP (Acc)	RTE (Acc)	SST-B (Spear)
BERT_{base}(Teacher)	—	110 M	60.1	84.7/85.0	93.2	91.6	89.4	91.4	70.4	89.2
BERT_6(Student)	—	66 M	51.2	81.7/82.6	91.0	88.8	84.7	90.4	66.1	88.3
Knowledge Distillation										
KD[50]	—	66 M	53.6	82.7/83.1	91.1	89.6	86.7	90.5	66.8	88.7
PKD[109]	—	66 M	54.5	82.7/83.3	91.3	89.5	85.0	90.9	66.6	88.2
TAKD[110]	—	66 M	53.8	82.5/83.0	91.4	89.6	85.0	90.7	67.5	88.0
CKD[97]	—	66 M	55.1	**83.6/84.1**	91.7	90.5	87.1	91.0	67.3	88.6
MetaDistil[100]	—	66 M	—	83.5/83.8	92.5	89.3	85.2	90.3	67.5	—
MGSKD[98]	—	66 M	49.0	83.0/83.6	90.6	**91.1**	87.4	**91.2**	67.9	88.2
AD-KD[99]	—	66 M	**58.1**	83.2/84.0	91.9	**91.1**	88.3	90.9	**69.3**	**88.7**
LGTM[101]	—	66 M	—	82.6/82.9	**92.8**	90.2	**88.6**	91.1	67.5	—
Pruning										
Magnitude[102]	90 %	—	—	78.3/79.3	—	—	—	79.8	—	—
L_0-regularization[105]	90 %	—	0.0	78.0/78.8	82.5	82.8	79.5	87.6	59.9	82.7
Movement[103]	90 %	—	—	80.1/80.4	—	—	—	89.7	—	—
Soft-Movement[103]	90 %	—	—	81.2/81.8	—	—	—	90.2	—	—
PLATON[106]	90 %	—	44.3	81.5/82.1	90.5	88.9	88.8	90.2	65.3	87.4
SMP[104]	90 %	—	—	82.5/82.3	—	—	—	**90.8**	—	—
PINS[108]	90 %	—	**49.8**	**82.5/82.9**	**91.0**	**89.5**	**90.1**	90.6	**68.5**	**87.4**
Parameters Sharing										
ProKT[111]	—	66 M	54.3	82.8/83.2	91.3	89.7	85.0	90.9	67.4	88.6
SFTN[112]	—	66 M	53.6	82.4/82.9	91.5	89.5	85.3	90.4	67.5	88.5
TinyBERT[96]	—	66 M	53.8	83.1/83.4	92.3	89.9	87.3	90.5	66.9	88.3
ALBERT[113]	—	64 M	—	—	—	—	—	—	—	—

具体来说, AD-KD[99]在较小数据集 (如 RTE 和 SST-B) 上表现更优, 可能是其在小型数据集中更擅长提取和利用关键信息. MGSKD[98]和 LGTM[101]在不同数据集上达到最佳性能, 表明它们能针对不同

数据类型找到高效的知识蒸馏策略. 值得一提的是, CKD[97] 在所有数据集上均表现良好, 显示关系型知识蒸馏方法 (如 CKD、MGSKD 和 AD-KD) 在提取和利用模型间关系知识方面具有优势. 相比之下, TAKD[110]、ProKT[111] 和 SFTN[112] 的表现较为均衡, 可能因为它们未能针对特定数据集找到最佳策略. 在模型剪枝比较中, PINS[108] 在所有数据集上表现领先, 表明其在模型压缩中具有高效性. PINS 在相同稀疏条件下, 能更好保留模型性能并降低复杂性. 同时, SMP[104] 也表现良好, 可能是其剪枝策略在降低复杂性的同时保持了较高性能.

为了对比低碳算法中典型应用的性能, 本文使用两个经典的深度学习模型 AlexNet 和 ResNet18, 对比它们在剪枝前后和量化前后在 CIFAR10、MNIST 和 ImageNet 数据集上的表现. 从表 5 中可以看出, 某些情况下, 剪枝和量化后的模型能够接近甚至超过基准模型的准确率. 例如, 在 CIFAR10 数据集上, 剪枝后的 AlexNet 在 Top1 精度上略有下降, 但 Top5 精度保持稳定, 这表明剪枝技术在不显著影响模型精度的情况下, 有助于显著减少参数量和计算开销. 此外, 在 MNIST 数据集上, 剪枝后的模型的 Top1 精度甚至有所提升, 这表明剪枝在此类较小规模的数据集上可能有助于提高模型性能. 在 ImageNet 数据集上, 量化后的模型精度也有所提升. 例如, ResNet18 在 ImageNet 数据集上经过 5 位量化后, Top1 和 Top5 精度均高于基准模型. 这表明适当的量化策略不仅能有效降低模型的计算需求, 还可能通过优化网络结构进一步提升模型的预测精度. 表 4 中的数据基于文献 [21,114] 中的典型参数设置和模型结构, 并在实验中进行了复现. 由于实验配置不同 (如学习率、训练轮数等), 结果中可能会出现细微差异.

表5　典型应用的低碳算法性能分析

深度学习模型	数据集	Method	Top1 accuracy	Top5 accuracy	参数数量/精度	浮点数
AlexNet[21,114]	CIFAR10	base model	78.45	98.45	2.87	92.27
		剪枝后[114]	76.24	98.32	0.99	30.74
	MNIST	base model	98.58	100.00	2.85	86.33
		剪枝后[115]	98.63	100.00	0.98	27.10
	ImageNet	base model	56.62	79.05	32 bit	—
		量化后[115]	58.00	81.10	4 bit	—
ResNet18[3]	CIFAR10	base model	—	93.00	32 bit	—
		量化后[116]	—	91.50	1 bit	—
	ImageNet	base model	68.27	88.69	32 bit	—
		量化后[117]	68.98	89.10	5 bit	—

6　低碳算法的评价标准

在优化算法设计和硬件架构领域, 诸多研究人员已进行了大量探索, 涉及低碳压缩算法设计、大模型加速框架研发以及工业界硬件架构设计等多个方面. 然而, 这些研究和实践中采用了不同的衡量方法, 导致结果难以直接对比. 因此, 亟需建立一套标准化的评估体系, 以确保不同方法间的可比性. 本章提出了现有的碳足迹计算方法以及关键指标来评估算法性能, 涵盖模型功耗与能耗、准确性与鲁棒性、吞吐量与延迟等重要特征. 通过对比这些指标, 可以更全面地评估不同技术的优劣, 并从中选择最适合特定应用场景的压缩技术.

6.1　低碳算法的碳足迹计算

为评估轻量化算法、硬件加速器及工业应用在低碳实践中的效果, 学术界与工业界正积极提出多样化的碳排放测算方法. 这些方法既能准确评估技术方案的环境影响, 又提高了低碳算法发展的透明度. 借助标准化体系, 各类企业可在统一基准下量化碳足迹, 并制定减碳措施. 例如, Strubell[17] 还提出了一套计算模型训练碳排放量的方法, 通过下式计算训练总功率:

$$p_t = \frac{1.58 \times t \times (p_c + p_r + g \times p_g)}{1\ 000}. \tag{12}$$

其中: t表示训练时间 (h), p_c和p_r分别代表 CPU 和内存的功耗 (W), p_g是单个 GPU 的功耗 (W), g是 GPU 的数量, 而 1.58 是所考虑数据中心的电源使用效率 (PUE) 系数. 式 (5) 综合硬件组件的能耗与数据中心的整体效率, 以精确量化训练期间的总电能消耗. 随后, 依据美国环境保护署 (EPA) 的数据, 将计算得到的总功率转换为相应的二氧化碳排放量

$$\mathrm{CO_2e} = 0.954 \times p_t. \tag{13}$$

其中: 0.954 是每千瓦时电力产生的平均二氧化碳量 (磅). 通过将总功率乘以这个系数, 可以估算出训练过程中的二氧化碳排放量. 此外, Dodge[118] 提出的公

式提供了一个简化的方法来估算机器学习模型的碳足迹. 公式如下:

$$\begin{aligned}&\text{Footprint} = \\&(\text{electrical\_energy}_{\text{train}} + \\&\text{queries} \times \text{electrical\_energy}_{\text{inference}}) \times \frac{\text{CO}_2\text{e}_{\text{datacenter}}}{\text{kWh}}.\end{aligned} \tag{14}$$

式 (14) 综合考虑了模型训练期间的电能消耗 ($\text{electrical\_energy}_{\text{train}}$) 与推理阶段每次查询的电能消耗 ($\text{electrical\_energy}_{\text{inference}}$) 乘以查询总数, 最终结果乘以数据中心每千瓦时电力产生的二氧化碳当量, 以此来全面评估模型在其生命周期内的碳排放量.

为了在训练和推理过程中测算碳排放, 除了碳测算的理论研究, 研究人员还开发了多种工具和插件. 例如, eco2AI[119] 系统可以实时监测模型训练过程中的能耗, 计算出相应的碳排放量. CodeCarbon[120] 则可以集成到开发环境中, 实时监测计算任务的能耗和碳排放. Green Algorithms[121] 是一个在线计算器, 用户可以通过参数来估算其碳排放量. 这些工具和插件的开发, 有助于形成统一的碳排放测算系统, 推动算法领域的可持续发展.

6.2 低碳算法的性能指标分析

为了能够公正地比较不同的低碳算法, 除了碳足迹之外, 还需要关注一系列性能指标. 这些指标包括但不限于准确率、精确率、召回率、F1 值、可解释性以及计算复杂度.

准确率通常指在标准数据集 (如 ImageNet) 上前 5 名误差的准确率; 精确率是指正确预测为正样本的样本数与所有预测为正样本的样本数之比; 召回率是指正确预测为正样本的样本数与所有实际为正样本的样本数之比; F1 值是精确率和召回率的加权平均值, 全面评估模型性能; 可解释性是指算法输出应能被解释和理解, 例如在股票价格预测中, 模型应能解释其预测基于哪些经济变量及其影响; 计算复杂度是指算法的处理时间和内存占用等资源要求, 影响实际应用的可行性, 不同应用场景对计算复杂度的要求不同.

在评估低碳算法时, 模型性能指标与碳足迹指标之间的平衡至关重要. 优秀的低碳算法模型应该是在性能与碳足迹之间的权衡, 即在这两个维度上, 寻找帕累托最优的低碳算法模型. 根据定义, 帕累托最优的点组成了帕累托边界, 而帕累托边界中的模型比其他模型更有效, 因为它们在给定的权衡下能够获得最佳的表现. 例如, 在相同的推理延迟下, 在帕累托最优边界上的模型能够获得最高的准确率. 因此, 当对某一低碳算法设计进行改进时, 应该使模型向帕累托边界方向上改进.

7 未来低碳算法发展方向

随着全球对气候变化和环境保护的重视, 低碳算法的发展备受关注. 现有模型压缩技术已能在不显著影响精度的情况下减少训练或部署的碳排放. 随着深度学习的不断推进, 低碳算法研究将愈发重要. 本章将探讨低碳算法未来的发展方向.

1) 自动化的模型压缩. 当前模型压缩技术多依赖人工干预和专家经验 (如量化中的位宽选择、低秩分解中的秩选择), 这种方式可能限制性能或带来误导. 因此, 未来重要方向是实现自动化模型压缩, 通过算法搜索最佳压缩方案. 一些研究已尝试利用强化学习技术[122], 但受限于启发式标准的泛化能力不足. 近年来, 神经架构搜索 (NAS) 技术[123] 逐渐成熟, 通过算法化评估, 可以高效找到最优网络架构. 在低碳算法背景下, 自动化模型压缩将成为关键技术, 用于构建高性能、低能耗的深度学习模型.

2) 模型压缩技术的优化组合方案设计. 目前, 模型压缩研究多集中于单一技术的开发和测试, 缺乏系统性组合设计的探索. 研究表明, 单一压缩技术的效果存在局限性, 多种技术结合可以进一步提升压缩效果和效率. 未来研究可根据不同任务场景和模型特性, 综合决策因素, 设计最优的压缩方案. 例如, 在剪枝和低秩组合优化中, 可通过最大化能耗降低、最小化精度损失为目标函数, 结合量化与剪枝等约束条件, 求解出最优方案. 模型如下所示:

$$\begin{aligned}&\text{maximize } E,\ \text{minimize } P;\\&\text{s.t. } f(\text{model}) \leqslant \theta,\\&\quad N \leqslant \omega,\\&\quad \Delta Q \leqslant \epsilon,\\&\quad Q(W,n) \leqslant m,\\&\quad Q_{\text{range}} \in [\min_{\text{val}}, \max_{\text{val}}].\end{aligned} \tag{15}$$

其中: E代表能耗降低, P代表精度损失. 约束条件包括: 模型结构约束$f(\text{Model}) \leqslant \theta$, 其中$f(\text{Model})$表示剪枝中模型结构的某个属性或参数的函数, θ是阈值; 模型大小约束$N \leqslant \omega$, 其中N表示剪枝后模型的参数数量, ω是大小限制; 量化误差约束$\Delta Q \leqslant \epsilon$, 其中$\Delta Q$是量化误差, ϵ为最大允许误差; 量化比特宽度约束$Q(W,n) \leqslant m$, W表示原始参数, n表示位数, 不能超过最大位数m; 量化范围约束$\min_{\text{val}} \leqslant Q_{\text{range}} \leqslant \max_{\text{val}}$, 确保量化的范围在给定区间内.

3) 低碳算法的框架设计和开发. 近些年, 学术界发布了一些针对低碳算法的模型压缩框架或算法库.

例如 NVIDIA 的 SparseBLAS/cuSPARSE 及 Intel 的 MKLSparseBLAS/cuSPARSE 和 MKL 是用来加速稀疏矩阵的操作库, 可以用于执行稀疏矩阵向量乘法、稀疏矩阵矩阵乘法、稀疏矩阵转置、稀疏矩阵求解等操作. 此外, Tensorflow 提供了一个量化库, 用于量化感知的训练和推理. 未来可以关注开发框架级的压缩技术, 以加速低碳算法的训练和部署.

4) 硬件-算法协同设计[124]. 低碳的模型压缩算法若没有与其兼容的硬件加速器, 则难以获得效率的最大提升. 例如, 不同的边缘设备处理不同精度的能力各异, 但硬件加速器通常只支持统一位宽的张量. 在这种情况下, 处理不同位宽的精度需要进行零填充, 从而导致降低内存处理的效率. 因此, 设计能够兼容低碳算法的硬件加速器, 同时关注硬件感知的压缩技术研究, 可以使得现有的模型压缩技术更容易部署, 提高内存的处理效率和模型压缩的性能.

5) 低碳算法的安全性. 对深度学习模型进行压缩后, 可能使得原始模型中的敏感信息暴露, 或使得攻击者可以更容易地发现模型中的漏洞, 对模型发起对抗攻击等[125]. 前文也提到, 针对隐私和安全性产生的机器遗忘技术的某些方法能够起到减小模型计算量的作用. 因此, 在追求降低能耗和保持精度的目标下, 需要探究通过不同压缩方式获得的模型是否具有更强的鲁棒性, 以及如何进一步提升这种鲁棒性. 同时, 将这种探索与机器遗忘技术相结合, 设计出既具高安全性又具优异性能的轻量级模型, 也是值得深入研究的方向.

6) 低碳算法与博弈结合. 博弈论作为一种强大的工具, 应用在模型训练能耗与所期望性能方面, 研究人员可以在能源消耗与模型性能之间找到平衡点, 以满足性能需求的同时降低能源成本; 此外, 各种压缩方法之间的博弈也能够作为一个研究的方向. 在大模型压缩领域, 每种方法都具有自身的优势和局限. 从博弈论的角度分析, 可以探究不同压缩方法之间的权衡, 找到最佳的组合策略, 从而在保持模型性能的同时减少模型的大小和能耗, 进一步推动低碳算法发展.

7) 低碳算法资源管理研究. 随着大模型算法的不断发展, 也可将这一理论应用在低碳算法资源管理领域, 研究资源管理方面的问题. 例如, 如何高效、经济地管理和利用物理计算资源; 如何对人力资源进行培训和指导, 使其能够适应不断变化的算法需求; 如何优化组织资源, 如流程、团队合作和决策机制, 以更好地应对计算和算法挑战; 如何在有限的计算资源下, 为不同的任务和模型分配合适的计算能力; 如何在不同的场景和需求下, 寻找算法资源与环境资源的平衡; 等等. 这些问题需要跨学科的合作, 包括计算机架构、算法及相关领域管理者的共同努力, 以达到资源的最优分配和管理.

8 结 语

随着计算能力和数据量的增加, 越来越多的领域如自然语言处理、计算机视觉和语音识别等领域开始使用大规模的模型进行任务效果的提升, 且模型的规模具有持续增长的趋势. 例如, Google 在 2017 年提出了 Transformer 架构, 解决了多个 AI 领域的发展瓶颈. 同时, 因为它便于分布式计算, 使得训练效率大大提升, 此后几年之中各种大型语言模型如同雨后春笋般冒出. 如 BERT, Pegasus, GPT 等, 这些大模型各自在不同应用场景下有着出众表现, BERT 相对更擅长于对文本的理解, Pegasus 相对更擅长于对文本做摘要, 而 GPT 更擅长于文本的生成. GPT 发布以后, OpenAI 通过不断增加模型的参数量, 从 GPT1 的亿级参数量到 GPT2 的 10 亿级参数量, 再到 2020 年发布的 GPT3, 具有 1 750 亿参数量. ChatGPT 不仅掌握了续写小说、写代码、做数学题的能力, 而且能够做到真正像人类一样聊天交流. 虽然大模型在性能和表现方面取得了显著的突破, 但其计算和存储需求也带来了挑战. 因此, 低碳算法成为一个重要的研究领域, 即在保证深度学习模型性能的同时减少其训练、部署所产生的碳足迹. 本文对该领域的经典及前沿研究进行了全面介绍, 还提出了比较不同低碳算法模型的评价指标和衡量策略. 希望本文能够为研究者们提供有益的参考和启示, 推动低碳算法模型的发展, 实现深度学习技术的可持续发展. 在不断发展的未来, 将大模型压缩到边缘设备可用体量和对计算资源的合理分配管理会是新的研究趋势和研究方向, 期待看见更多创新的模型压缩技术给大模型领域带来的改变.

参考文献 (References)

[1] Duan H, Zhou S, Jiang K, et al. Assessing China's efforts to pursue the 1.5℃ warming limit[J]. Science, 2021, 372(6540): 378-385.

[2] 鲍健强, 苗阳, 陈锋. 低碳经济: 人类经济发展方式的新变革[J]. 中国工业经济, 2008(4): 153-160.
(Bao J Q, Miao Y, Chen F. Low carbon economy: revolution in the way of human economic development[J]. China Industrial Economics, 2008(4): 153-160.)

[3] He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition[C]. 2016 IEEE Conference on Computer Vision and Pattern

Recognition. Las Vegas, 2016: 770-778.

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

[5] Szegedy C, Liu W, Jia Y Q, et al. Going deeper with convolutions[C]. 2015 IEEE Conference on Computer Vision and Pattern Recognition. Boston, 2015: 1-9.

[6] 侯志强, 郭凡, 杨晓麟, 等. 基于混合注意力的Transformer视觉目标跟踪算法[J]. 控制与决策, 2024, 39(3): 739-748.
(Hou Z Q, Guo F, Yang X L, et al. Transformer visual target tracking algorithm based on mixed attention[J]. Control and Decision, 2024, 39(3): 739-748.)

[7] 闫涵, 卢伟, 吴玉虎. 基于轻量化卷积神经网络的金属断口图像识别[J]. 控制与决策, 2024, 39(9): 2858-2866.
(Yan H, Lu W, Wu Y H. Metal fracture recognition based on lightweight convolutional neural network[J]. Control and Decision, 2024, 39(9): 2858-2866.)

[8] Devlin J, Chang M W, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding[J/OL]. 2018, arXiv: 1810.0480.

[9] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]. Advances in Neural Information Processing Systems(NIPS'17). Long Beach, 2017: 6000-6010.

[10] 肖红军. 算法责任: 理论证成、全景画像与治理范式[J]. 管理世界, 2022, 38(4): 200-226.
(Xiao H J. Algorithmic responsibility: Theoretical justification, panoramic portrait and governance paradigm[J]. Journal of Management World, 2022, 38(4): 200-226.)

[11] Zhao H, Wu L, Shan Y, et al. A comprehensive survey of large language models in management: Applications, challenges, and opportunities[J]. Jornal of Latex Class Files, 2024, 14(8): 1-22.

[12] Murugesan S. Harnessing green IT: Principles and practices[J]. IT Professional, 2008, 10(1): 24-33.

[13] UNFCCC. Go climate neutral now[DB/OL]. (2015-09-22). https://unfccc.int/news/go-climate-neutral-now.

[14] Google. Using location to reduce our computing carbon footprint[DB/OL]. (2021-05-18)[2023-12-10]. https://blog.google/outreach-initiatives/sustainability/.

[15] Microsoft Corporation. Microsoft will be carbon negative by 2030[DB/OL]. (2020-01-16)[2023-12-10]. https://blogs.microsoft.com/blog/2020/01/16/.

[16] Schwartz R, Dodge J, Smith N A, et al. Green AI[J]. Communications of the ACM, 2020, 63(12): 54-63.

[17] Strubell E, Ganesh A, McCallum A. Energy and policy considerations for deep learning in NLP[J/OL]. 2019, arXiv: 1906.02243.

[18] Patterson D A, Gonzalez J, Le Q V, et al. Carbon emissions and large neural network training[J/OL]. 2021, arXiv: 2104.10350.

[19] HAI. AI index report 2024: Version 1.0[R]. Stanford: Stanford Institute for Human-Centered Artificial Intelligence, 2024: 02-94.

[20] Luccioni S, Jernite Y, Strubell E. Power hungry processing: Watts driving the cost of AI deployment? [C]. The 2024 ACM Conference on Fairness, Accountability, and Transparency. New York: ACM, 2024: 85-99.

[21] Choudhary T, Mishra V, Goswami A, et al. A comprehensive survey on model compression and acceleration[J]. Artificial Intelligence Review, 2020, 53(7): 5113-5155.

[22] Zhao H K, Zheng S M, Wu L K, et al. LANE: Logic alignment of non-tuning large language models and online recommendation systems for explainable reason generation[J/OL]. 2024, arXiv: 2407.0283.

[23] Shuvo M M H, Islam S K, Cheng J L, et al. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review[J]. Proceedings of the IEEE, 2023, 111(1): 42-91.

[24] Deng B L, Li G Q, Han S, et al. Model compression and hardware acceleration for neural networks: A comprehensive survey[J]. Proceedings of the IEEE, 2020, 108(4): 485-532.

[25] Hassibi B, Stork D G, Wolff G J. Optimal brain surgeon and general network pruning[C]. IEEE International Conference on Neural Networks. San Francisco, 1993: 293-299.

[26] Dong X, Chen S Y, Pan S J. Learning to prune deep neural networks via layer-wise optimal brain surgeon[J/OL]. 2017, arXiv: 2407.0283.

[27] Suzuki K, Horiba I, Sugie N. A simple neural network pruning algorithm with application to filter synthesis[J]. Neural Processing Letters, 2001, 13(1): 43-53.

[28] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[J/OL]. 2015, arXiv: 1506.02626.

[29] Liu Z, Li J G, Shen Z Q, et al. Learning efficient convolutional networks through network slimming[C]. 2017 IEEE International Conference on Computer Vision. Venice, 2017: 2755-2763.

[30] He Y H, Zhang X Y, Sun J. Channel pruning for accelerating very deep neural networks[C]. 2017 IEEE International Conference on Computer Vision. 2017, arXiv: 1707.06168.

[31] Wu Z X, Nagarajan T, Kumar A, et al. BlockDrop: Dynamic inference paths in residual networks[C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, 2018: 8817-8826.

[32] Wang X, Yu F, Dou Z Y, et al. SkipNet: Learning dynamic routing in convolutional networks[C]. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018: 420-436.

[33] Yu R C, Li A, Chen C F, et al. NISP: Pruning networks using neuron importance score propagation[C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, 2018: 9194-9203.

[34] Horowitz M. 1.1 Computing's energy problem (and

what we can do about it)[C]. 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers. San Francisco, 2014: 10-14.

[35] Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, 2018: 2704-2713.

[36] Jiang J J, Zhao H K, He M, et al. Knowledge-aware cross-semantic alignment for domain-level zero-shot recommendation[C]. Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. Birmingham United Kingdom, 2023: 965-975.

[37] Courbariaux M, Bengio Y, David J P. BinaryConnect: Training deep neural networks with binary weights during propagations[C]. Proceedings of the 28th International Conference on Neural Information Processing Systems. Montréal, 2015: 3123-3131.

[38] Rastegari M, Ordonez V, Redmon J, et al. XNOR-net: ImageNet classification using binary convolutional neural networks[C]. European Conference on Computer Vision. Cham: Springer, 2016: 525-542.

[39] Nagel M, Amjad R A, Van Baalen M, et al. Up or down? adaptive rounding for post-training quantization[C]. Proceedings of the 37th International Conference on Machine Learning. Vienna, 2020: 7197-7206.

[40] Bourlard H, Kamp Y. Auto-association by multilayer perceptrons and singular value decomposition[J]. Biological Cybernetics, 1988, 59(4): 291-294.

[41] Shim K, Lee M, Choi I, et al. SVD-Softmax: Fast softmax approximation on large vocabulary neural networks[C]. Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach California, 2017: 5469-5479.

[42] Zhang X Y, Zou J H, Ming X, et al. Efficient and accurate approximations of nonlinear convolutional networks[C]. 2015 IEEE Conference on Computer Vision and Pattern Recognition. Boston, 2015: 1984-1992.

[43] Kolda T G, Bader B W. Tensor decompositions and applications[J]. SIAM Review, 2009, 51(3): 455-500.

[44] Tucker L R. Some mathematical notes on three-mode factor analysis[J]. Psychometrika, 1966, 31(3): 279-311.

[45] Cichocki A, Mandic D, De Lathauwer L, et al. Tensor decompositions for signal processing applications: From two-way to multiway component analysis[J]. IEEE Signal Processing Magazine, 2015, 32(2): 145-163.

[46] Swaminathan S, Garg D, Kannan R, et al. Sparse low rank factorization for deep neural network compression[J]. Neurocomputing, 2020, 398: 185-196.

[47] Buciluǎ C, Caruana R, Niculescu-Mizil A. Model compression[C]. Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. Philadelphia, 2006: 535-541.

[48] Ba J, Caruana R. Do deep nets really need to be deep?[C]. Proceedings of the 27th International Conference on Neural Information Processing Systems. Montréal, 2014: 2654-2662.

[49] Hinton G E, Vinyals O, Dean J. Distilling the knowledge in a neural network[J/OL]. 2015, arXiv: 1503.02531.

[50] Pan S J, Yang Q. A survey on transfer learning[J]. IEEE Transactions on Knowledge and Data Engineering, 2010, 22(10): 1345-1359.

[51] Wang J H, Wu L K, Zhao H K, et al. Multi-view enhanced zero-shot node classification[J]. Information Processing & Management, 2023, 60(6): 103479.

[52] Chen W L, Wilson J T, Tyree S, et al. Compressing neural networks with the hashing trick[C]. The 32nd International Conference on Machine Learning, Montreal, 2015: 2275-2284.

[53] Cheng Y, Yu F X, Feris R S, et al. An exploration of parameter redundancy in deep networks with circulant projections[C]. 2015 IEEE International Conference on Computer Vision. Santiago, 2015: 2857-2865.

[54] Wang Y, Xu C, Xu C, et al. Beyond filters: Compact feature map for portable deep model[C]. International Conference on Machine Learning. Sydney, 2017: 3703-3711.

[55] Wu J R, Wang Y, Wu Z Y, et al. Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions[C]. International Conference on Machine Learning. Stockholm, 2018: 5363-5372.

[56] Sifre L, Mallat S. Rigid-motion scattering for texture classification[J/OL]. 2014, arXiv: 1403.1687.

[57] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]. 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, 2016: 2818-2826.

[58] Chollet F. Xception: Deep learning with depthwise separable convolutions[C]. 2017 IEEE Conference on Computer Vision and Pattern Recognition. Honolulu, 2017: 1800-1807.

[59] Howard A G, Zhu M L, Chen B, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications[J/OL]. 2017, arXiv: 1704.04861.

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

[61] Han S, Mao H Z, Dally W. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding[J/OL]. 2016, arXiv: 1510.00149.

[62] Iandola F, Moskewicz M W, Karayev S, et al. DenseNet: Implementing efficient ConvNet descriptor

Pyramids[J/OL]. 2014, arXiv: 1404.1869.

[63] Huang G, Liu S C, van der Maaten L, et al. CondenseNet: An efficient denseNet using learned group convolutions[C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, 2018: 2752-2761.

[64] Micikevicius P, Narang S R, Alben J, et al. Mixed precision training[J/OL]. 2014, arXiv: 1710.03740.

[65] Vasudevan A, Anderson A, Gregg D. Parallel multi channel convolution using general matrix multiplication[J/OL]. 2017, arXiv: 1704.04428.

[66] Cloutier J, Cosatto E, Pigeon S, et al. VIP: An FPGA-based processor for image processing and neural networks[C]. Proceedings of 5th International Conference on Microelectronics for Neural Networks. Lausanne, 1996: 330-336.

[67] Han S, Kang J L, Mao H Z, et al. ESE: Efficient speech recognition engine with sparse LSTM on FPGA[J/OL]. 2016, arxiv: 1612.00694.

[68] Wang J, Gu S S. FPGA implementation of object detection accelerator based on vitis-AI[C]. The 11th International Conference on Information Science and Technology. Chengdu, 2021: 571-577.

[69] Pham P H, Jelaca D, Farabet C, et al. NeuFlow: Dataflow vision processing system-on-a-chip[C]. 2012 IEEE 55th International Midwest Symposium on Circuits and Systems. Boise, 2012: 1044-1047.

[70] Chen T, Du Z, Sun N, et al. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning[C]. Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems. Salt Lake City Utah, 2014: 269-284.

[71] Chen Y J, Luo T, Liu S L, et al. DaDianNao: A machine-learning supercomputer[C]. The 47th Annual IEEE/ACM International Symposium on Microarchitecture. Cambridge, 2014: 609-622.

[72] Luo T, Liu S L, Li L, et al. DaDianNao: A neural network supercomputer[J]. IEEE Transactions on Computers, 2017, 66(1): 73-88.

[73] So D R, Liang C, Le Q V. The Evolved Transformer[J/OL]. 2019, arXiv: 1901.11117.

[74] Lepikhin D, Lee H, Xu Y Z, et al. GShard: Scaling giant models with conditional computation and automatic sharding[J/OL]. 2020, arXiv: 2006.16668.

[75] Fedus W, Zoph B, Shazeer N M. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity[J/OL]. 2022, arXiv: 2101.03961.

[76] Zhang R R, Han J M, Zhou A J, et al. LLaMA-adapter: Efficient fine-tuning of language models with zero-init attention[J/OL]. 2023, arXiv: 2303.16199.

[77] Gao P, Han J M, Zhang R R, et al. LLaMA-adapter V2: Parameter-efficient visual instruction model[J/OL]. 2023, arXiv: 2304.15010.

[78] Hu J E, Shen Y L, Wallis P, et al. LoRA: Low-rank adaptation of large language models[J/OL]. 2021, arXiv: 2106.09685.

[79] Frantar E, Alistarh D. Sparsegpt: Massive language models can be accurately pruned in one-shot[C]. Proceedings of the 40th International Conference on Machine Learning. Honolulu, 2023: 10323-10337.

[80] Sun M J, Liu Z, Bair A, et al. A simple and effective pruning approach for large language models[J/OL]. 2023, arXiv: 2306.11695.

[81] Xiao G, Lin J, Seznec M, et al. Smoothquant: Accurate and efficient post-training quantization for large language models[C]. International Conference on Machine Learning. Hawaii, 2023: 38087-38099.

[82] Frantar E, Ashkboos S, Hoefler T, et al. GPTQ: Accurate post-training quantization for generative pre-trained transformers[J/OL]. 2022, arXiv: 2210.17323.

[83] Frantar E, Alistarh D. Optimal brain compression: A framework for accurate post-training quantization and pruning[C]. Proceedings of the 36th International Conference on Neural Information Processing Systems. New Orleans, 2022: 4475-4488.

[84] Peng B, Alcaide E, Anthony Q, et al. RWKV: Reinventing RNNs for the transformer era[C]. Findings of the Association for Computational Linguistics: EMNLP 2023. Singapore, 2023: 14048-14077.

[85] Sun Y T, Dong L, Huang S H, et al. Retentive network: A successor to transformer for large language models[J/OL]. 2023, arXiv: 2307.08621.

[86] Gu A, Dao T. Mamba: Linear-time sequence modeling with selective state spaces[J/OL]. 2024, arXiv: 2312.00752.

[87] Wang A, Singh A, Michael J, et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding[C]. Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Brussels, 2018: 353-355.

[88] Warstadt A, Singh A, Bowman S R. Neural network acceptability judgments[J]. Transactions of the Association for Computational Linguistics, 2019, 7: 625-641.

[89] Williams A, Nangia N, Bowman S. A broad-coverage challenge corpus for sentence understanding through inference[C]. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics. New Orleans, 2017: 1112-1122.

[90] Socher R, Perelygin A, Wu J, et al. Recursive deep models for semantic compositionality over a sentiment treebank[C]. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle, 2013: 1631-1642.

[91] Rajpurkar P, Zhang J, Lopyrev K, et al. SQuAD: 100, 000+ questions for machine comprehension of text[C]. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, 2016: 2383-2392.

[92] Dolan W B, Brockett C. Automatically constructing a corpus of sentential paraphrases[C]. In Proceedings of the Third International Workshop on Paraphrasing (IWP2005). Jeju Island, 2005: 9-16.

[93] Wang Z G, Hamza W, Florian R. Bilateral multi-perspective matching for natural language sentences[C]. Proceedings of the 26th International Joint Conference on Artificial Intelligence. Melbourne, 2017: 4144–4150.

[94] Luisa Bentivogli P C, Dagan I. The fifth pascal recognizing textual entailment challenge[C]. Machine Learning Challenges Workshop. Berlin, 2009: 177-190.

[95] Cer D, Diab M, Agirre E, et al. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation[C]. Proceedings of the 11th International Workshop on Semantic Evaluation. Vancouver, 2017: 1-14.

[96] Jiao X Q, Yin Y C, Shang L F, et al. TinyBERT: Distilling BERT for natural language understanding[C]. Findings of the Association for Computational Linguistics: EMNLP 2020. Stroudsburg, 2020: 4163-4147.

[97] Park G, Kim G, Yang E. Distilling linguistic context for language model compression[C]. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Punta Cana, 2021: 364-378.

[98] Liu C, Tao C Y, Feng J Z, et al. Multi-granularity structural knowledge distillation for language model compression[C]. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. Dublin, 2022: 1001-1011.

[99] Wu S, Chen H, Quan X, et al. Ad-kd: Attribution-driven knowledge distillation for language model compression[J/OL]. 2023, arXiv: 2305.10010.

[100] Zhou W, Xu C W, McAuley J. BERT learns to teach: Knowledge distillation with meta learning[C]. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. Dublin, 2022: 7037-7049..

[101] Ren Y X, Zhong Z Q, Shi X J, et al. Tailoring instructions to student's learning levels boosts knowledge distillation[J/OL]. 2023, arXiv: 2305.09651.

[102] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[C]. Proceedings of the 28th International Conference on Neural Information Processing Systems. Montreal, 2015: 1135-1143.

[103] Sanh V, Wolf T, Rush A M. Movement pruning: Adaptive sparsity by fine-tuning[J/OL]. 2020, arXiv: 2005.07683.

[104] Jiang T, Wang D Q, Zhuang F Z. Pruning pre-trained language models without fine-tuning[J/OL]. 2022, arXiv: 2210.06210.

[105] Louizos C, Welling M, Kingma D P. Learning sparse neural networks through L0 regularization[J/OL]. 2017, arXiv: 1712.01312.

[106] Zhang Q R, Zuo S M, Liang C, et al. PLATON: Pruning large transformer models with upper confidence bound of weight importance[C]. International Conference on Machine Learning. Maryland, 2022: 26809-26823.

[107] Xu R X, Luo F L, Wang C Y, et al. From dense to sparse: Contrastive pruning for better pre-trained language model compression[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2022, 36(10): 11547-11555.

[108] Ren S Y, Zhu K. Pruning pre-trained language models with principled importance and self-regularization[C]. Findings of the Association for Computational Linguistics. Toronto, 2023: 8995-9008.

[109] Sun S, Cheng Y, Gan Z, et al. Patient knowledge distillation for bert model compression[C]. Conference on Empirical Methods in Natural Language Processing. Hong Kong, China: Association for Computational Linguistics, 2019: 4323-4332.

[110] Mirzadeh S I, Farajtabar M, Li A, et al. Improved knowledge distillation via teacher assistant[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 34(4): 5191-5198.

[111] Shi W, Song Y, Zhou H, et al. Learning from deep model via exploring local targets[DB/OL]. (2023-05-06)[2023-12-10]. https://openreview.net/forum?id=5slGDu_bVc6.

[112] Park D Y, Cha M, Jeong C, et al. Learning student-friendly teacher networks for knowledge distillation[C]. Proceedings of the 35th International Conference on Neural Information Processing Systems. New York, 2021: 13292-13303.

[113] Lan Z, Chen M, Goodman S, et al. ALBERT: A lite BERT for self-supervised learning of language representations[J/OL]. (2020-02-09)[2023-12-10]. https://doi.org/10.48550/arXiv.1909.11942.

[114] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[115] Zhuang B H, Shen C H, Tan M K, et al. Towards effective low-bitwidth convolutional neural networks[C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, 2018: 7920-7928.

[116] Qin H T, Gong R H, Liu X L, et al. Forward and backward information retention for accurate binary neural networks[C]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Seattle, 2020: 2250-2259.

[117] Zhou A J, Yao A B, Guo Y W, et al. Incremental network quantization: Towards lossless CNNs with low-precision weights[J/OL]. 2017. arXiv: 1702.03044.

[118] Dodge J, Prewitt T, Tachet des Combes R, et al. Measuring the carbon intensity of AI in cloud instances[C]. 2022 ACM Conference on Fairness,

Accountability, and Transparency. Seoul, 2022: 1877-1894.

[119] Budennyy S A, Lazarev V D, Zakharenko N N, et al. eco2AI: Carbon emissions tracking of machine learning models as the first step towards sustainable AI[J]. Doklady Mathematics, 2022, 106(1): S118-S128.

[120] Courty B, Schmidt V, Luccioni S, et al. mlco2/codecarbon: v2.4.1[CP/OL].(2024-05-10)[2024-08-05]. https://doi.org/10.5281/zenodo.11171501.

[121] Lannelongue L, Grealey J, Inouye M. Green algorithms: Quantifying the carbon footprint of computation[J]. Advanced Science, 2021, 8(12): 2100707.

[122] Sun S Q, Cheng Y, Gan Z, et al. Patient knowledge distillation for BERT model compression[C]. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. Hong Kong, 2019: 4323-4332.

[123] Lyu B, Wen S, Shi K, et al. Multiobjective reinforcement learning-based neural architecture search for efficient portrait parsing[J]. IEEE Transactions on Cybernetics, 2021, 53(2): 1158-1169.

[124] Ren P, Xiao Y, Chang X, et al. A comprehensive survey of neural architecture search: Challenges and solutions[J]. ACM Computing Surveys (CSUR), 2021, 54(4): 1-4.

[125] Chakraborty A, Alam M, Dey V, et al. A survey on adversarial attacks and defences[J]. CAAI Transactions on Intelligence Technology, 2021, 6(1): 25-45.

作者简介

赵洪科 (1988−), 男, 副教授, 博士, 硕士生导师, 主要研究方向为数据挖掘、算法管理, E-mail: hongke@tju.edu.cn;

叶倩彤 (2000−), 女, 硕士生, 主要研究方向为数据挖掘、算法管理, E-mail: qye63@tju.edu.cn;

张志勇 (1995−), 男, 博士生, 主要研究方向为决策分析、机器学习, E-mail: 1021209009@tju.edu.cn;

张凯 (1993−), 男, 副研究员, 博士, 主要研究方向为人工智能、自然语言处理, E-mail: kkzhang08@ustc.edu.cn;

汪珂航 (1999−), 男, 硕士, 主要研究方向为数据挖掘、人工智能, E-mail: wangkehang@mail.ustc.edu.cn;

黄振亚 (1992−), 男, 副教授, 博士, 硕士生导师, 主要研究方向为数据挖掘、推荐系统, E-mail: huangzhy@ustc.edu.cn.