云边协同大模型块粒度重训方法

张青龙,韩 锐*,刘 驰
(北京理工大学计算机学院,北京 100081)

摘 要: 边缘侧大模型外部环境的不确定性(如路边摄像头画面中天气、光照、物体密度的变化),导致其输入数据分布持续改变,因此需进行重训以维持高精度. 受限于设备可用资源和重训窗口,现有技术仅能训练固定压缩模型,其有限的泛化能力导致模型精度显著降低. 本文提出云边协同大模型块粒度重训方法,引入模型重训缩放定律评估不同块对边缘侧当前数据的精度贡献,以此为依据生成有限资源下最优重训方案,将云平台大模型中精度最相关部分动态转换为边缘侧可重训小模型,构建大小模型协同训练系统. 真实云边平台上对比实验表明,本文方法可以在相同资源消耗下提升大模型重训精度81.24%,并支持最大至330亿参数大模型重训.

关键词: 大模型;边缘侧动态环境;模型重训;缩放定律;云边大小模型协同训练
基金项目: 国家重点研发计划(No.2023YFE0209100);国家自然科学基金(No.62272046,No.62132019,No.61872337)
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112(2025)02-0287-14
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240518

Cloud-Edge Collaborative Retraining of Foundation Models at the Block Granularity

ZHANG Qing-long, HAN Rui*, LIU Chi
(*School of Computer Science*, *Beijing Institute of Technology*, *Beijing* 100081, *China*)

Abstract: Foundation models deployed in dynamic edge environment encounter continuously evolving input data distributions, requiring retraining them to maintain high accuracy. However, existing retraining techniques can only train fixed compressed models within the constraints of device resources and retraining windows, thus considerably lowering accuracies due to these small models' limited generalization ability. For such an issue, this paper proposes BlockTrainer, an edge-cloud collaborative retraining approach of foundation models at the block granularity. BlockTrainer first introduces a model retraining scaling law to evaluate the accuracy contributions of different blocks in a foundation model according to its latest input data at edge. Based on this evaluation, it generates the optimal retraining solution under resource constraints, and dynamically converts the most accuracy-relevant parts of the model into retrainable small models at edge, thereby constructing a collaborative training system between large and small models. Comparative experiments on real edge-cloud platforms show that BlockTrainer improves the retraining accuracy of foundation models by 81.24% using the same resource consumptions, and supports retraining a model of up to 33 billion parameters.

Key words: foundation model; dynamic environment at edge; model retraining; scaling law; edge-cloud collaborative retraining of large and small models

Foundation Item(s): National Key Research and Development Program of China (No.2023YFE0209100); National Natural Science Foundation of China (No.62272046, No.62132019, No.61872337)

1 引言

大模型(如 GPT 和 Swin Transformer)在计算机视觉、自然语言处理和多模态模型等领域已有广泛的应用,其正以卓越的推理和生成能力成为构建通用人工智能的基石[1]. 大模型卓越的能力主要来源于其远超于传统卷积神经网络(Convolutional Neural Networks, CNN)的参数量,例如SwinV2的参数量是ResNet-18的300倍,但相应地,其部署后的微调开销也大大增加. 例

收稿日期:2024-06-04;修回日期:2024-10-27;责任编辑:孙瑶
*通讯作者:韩锐

如,原始大小的SwinV2在目标数据集上进行微调时,即使在云服务器上也需要数天时间. 因此,现有技术首先压缩预训练大模型,并将其部署至距离数据源头更近的边缘设备以避免高传输带宽和隐私泄露,之后利用本地数据微调以提供低延迟、高精度的服务[2]. 然而,边缘侧开放环境下模型输入数据分布的动态变化,往往导致所部署的边缘侧压缩模型的精度显著降低,因此需要持续对其进行重训练[2,3],提高边缘侧压缩模型在本地的精度. 例如,在云端合成自动驾驶数据集GTA5上预训练的SwinV2模型在压缩并部署到边端后,面临真实自动驾驶数据集Cityscapes时精度会下降67.14%,但快速重训边缘侧压缩模型可以恢复其服务精度.

1.1 边缘侧大模型重训性能瓶颈

图1(a)展示了一个边缘端模型重训案例:视觉模型SwinV2首先在源数据集GTA5上进行预训练,之后部署到边缘设备NVIDIA Jetson AGX Xavier(512个图形处理器(Graphics Processing Unit,GPU)核、32 GB内存)上,在运行中需要根据3个新的目标数据集(晴天、阴天和黑夜环境下的Cityscapes数据集)分别进行重训练,并在有限重训窗口(10 min)完成以快速恢复在线服务精度. 然而,在资源受限边缘设备上,基于现有重训练技术(图1(b)中方案一、二)的模型的重训精度远远低于大模型本身学习能力上限(图1(b)红色虚线). 这是由于方案一在边缘设备直接重训大模型[4],即使使用最先进的参数高效微调方法(如LoRA[5]),在批处理大小为16时,对SwinV2进行微调也需要消耗29.4 GB的内存,因此只能以极小的批处理大小进行训练,收敛时间远远长于重训窗口. 使用更常见的方案二直接重训压缩模型[2,6~8],可以利用有限重训窗口完成大部分训练过程,重训精度比方案一更高(如图1(b)中高26.24%),但受限于小模型的泛化能力,方案二精度仍然远远低于理想精度(如图1(b)中低42.58%).

1.2 块粒度模型重训动机

针对以Transformer为代表的大模型,模型块是其基本组成单位,包含1个多头自注意力层、1个投影层和1个前馈网络. 如图1(a)所示,1个典型的大模型由数个模型块顺序构成(例如视觉大模型SwinV2-Large[9]由24个模型块顺序连接而成),2个相邻模型块之间通过输入输出建立连接关系. 不同模型块在模型重训中的前向传播和反向传播中处于不同的位置,因此对模型精度有着不同的影响. 对于其他类型大模型,模型精度取决于其网络架构,一般也以模型块为基本构成单位. 例如,CNN模型中的ResBlock模型块[10],长短期记忆网络(Long Short-Term Memory,LSTM)模型的LSTMCell模型块[11],以及扩散模型的ConvBlock或R-CNNBlock模型块[12]. 现有技术未考虑大模型不同模型块对当前目标数据集精度的重要性,直接重训根据源域数据集生成的压缩模型,因此可能因压缩重要模型块而显著降低大模型泛化能力. 为了验证以上观点,我们定义模型块的重要性为重训中其对精度的贡献百分比:

$$R_{\mathrm{c}}=\Delta A_k/\sum_{j=1}^{N}\Delta A_j \tag{1}$$

其中,ΔA_k为大模型被压缩第$k(1\leqslant k\leqslant N)$个模型块与不进行任何压缩的重训精度之差,即压缩某个模型块后重训精度下降越多,该块的重要性就越高.

图1(c)展示了SwinV2模型在面对3个不同目标数据集时重要性最低的50%模型块的编号及其对精度的重要性. 我们可以看到,不同目标数据集上最不重要的模型块会发生很大变化,在同一个目标数据集上不同模型块的重要性也有差异. 图1(d)进一步展示了不同模型块压缩方案对重训精度的影响:在每次重训中,压缩对当前目标数据集重要性最低的50%模型块时,其重训精度比另外2种方案(压缩最重要模型块、根据源域数据压缩模型块)高47.76%.

块粒度模型重训旨在从更细的粒度探究大模型重训练过程,将有限资源用于对精度最重要的模型块,因此面临一系列新的技术挑战.

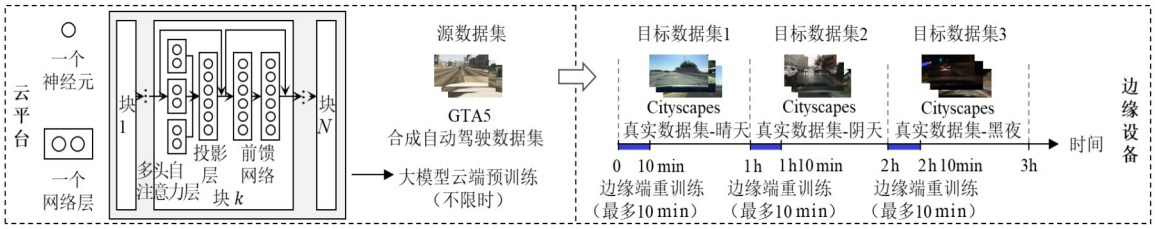
首先,定量评估模型块与重训精度的关系. 现有神经网络缩放定律主要聚焦于云平台大模型预训练场景[13,14],评估模型参数量、训练集和计算资源三要素与重训精度的关系. 然而,边缘侧模型重训面临更多不确定因素,包括持续变化的目标数据集分布特征和模型初始参数(受模型预训练和之前重训影响),需要在考虑这些因素的同时在模型块粒度构建缩放定律.

其次,在边缘侧动态环境下,每次模型重训均会因系统状况(如可用计算资源、输入数据分布)的变化而有不同的最优方案(即最大化模型精度的模型块压缩方案和训练迭代数),需要综合考虑多个重训精度影响因素和系统状况,快速搜索最优方案.

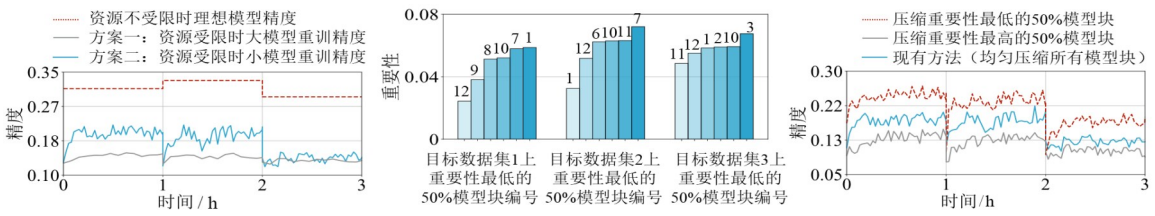
最后,成功实施重训最优方案需要在尽可能少泄露数据隐私的前提下,将云平台原始大模型根据边缘侧本地数据转化为压缩小模型,并保留大模型中精度最相关的部分;并在避免执行高开销大模型训练的情况下,将小模型学习知识快速反馈给大模型,形成云边环境下大小模型协同训练系统.

1.3 本文研究内容

本文提出云边协同块粒度重训方法BlockTrainer,引入大模型重训缩放定律,并以其量化评估为依据指导云边协同大模型训练优化,具体包括4个步骤:①当边缘侧输入数据分布发生变化时,将可用资源情况、输入数据统计特征(如均值和方差)和其中最困难的一个样本传输至云平台;②云平台根据统计特征拟合缩放定律公



(*a*) 边缘端重训练场景



(*b*) 大/小模型的重训精度　(*c*) 模型块对重训精度的贡献百分比　(*d*) 不同模型块压缩方案的重训精度

图1　边缘端重训示例

式参数,并预测不同模型块压缩方案的重训精度;③根据预测输出,云平台从所有模型块压缩方案中选出单位计算资源下最高重训精度提升的方案,并生成对应的小模型;④在边缘侧重训小模型,并将所学知识反馈回云平台大模型实现协同训练. 具体地,本文贡献如下:

(1)大模型重训缩放定律. 在使用形如$y=ax^b$的幂律公式[13]定量建模了模型块参数、训练迭代数与模型精度之间的关系的基础上,进一步通过当前目标数据集统计特征、小模型初始化参数对模型精度进行校准,从而实现对模型性能的精准预测.

(2)基于启发式搜索的重训方案组合优化算法. 将最优重训方案搜索转换为组合优化问题,其优化目标为最大化重训过程中的平均精度,优化约束为重训窗口和可用内存,提出启发式算法在二次多项式时间内求解该问题.

(3)云边协同大小模型训练. 使用边缘侧目标数据集中最困难的样本计算大模型中神经元的重要性,从而在云平台将最重要神经元提取出来组合为小模型,并通过神经元索引建立大小模型间的对应关系. 进一步地,在边缘侧重训小模型,并将所学知识反馈回云平台以更新大模型对应神经元,实现大小模型协同训练.

论文基于PyTorch完整实现了BlockTrainer,其可在边缘设备上支持计算机视觉或自然语言处理大模型的重训练. 在评测中构建了4个边缘端重训练负载,其部署了参数量从0.9亿(0.09B)至30亿(3B)的4个大模型,分别运行图像分类、语义分割、词性标注和任务型对话4个主流应用,分别在4个边缘设备NVIDIA Xavier NX、NVIDIA AGX Xavier、NVIDIA AGX Orin (32 GB)和NVIDIA AGX Orin (64 GB)上运行. 我们在这4个边缘端重训练负载上进行了详尽的模块分析实验,并与最先进大模型压缩重训方法进行了对比实验. 结果显示:①BlockTrainer大模型重训缩放定律的平均绝对误差(预测值与真实值之差)为0.12%,平均相对误差(绝对误差除以真实值)为3.67%,对比云平台预训练缩放定律误差平均降低94.45%;②BlockTrainer重训方案优化器的平均搜索时间为5.30 s(仅占用重训窗口的0.46%),云边协同大小模型训练器的小模型生成和反馈时间平均为6.30 s(仅占用重训窗口的0.58%);③BlockTrainer在同样计算资源下相对提升了81.24%的重训精度,在同样重训精度下平均降低了92.15%的计算资源消耗;④BlockTrainer适用于不同的大模型和重训练算法,并可在边缘设备NVIDIA AGX Orin (64 GB)上支持最大至330亿参数量的大模型.

2　背景与相关工作

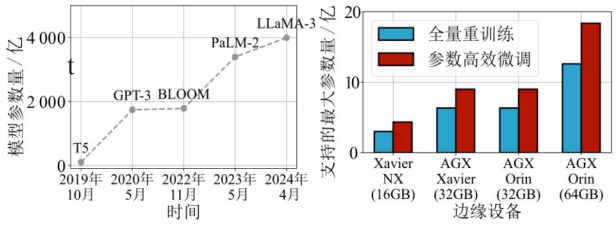
2.1　大模型参数量发展趋势

2020年1月,OpenAI提出大模型缩放定律,指出大模型的性能与模型参数量之间呈正相关幂律关系. 同年5月,OpenAI推出大模型GPT-3,其参数量为1 750亿,是之前大模型(如T5)的数十倍,其以强大的推理能力证明了缩放定律的有效性. 因此,在此之后研究者不断扩大大模型的参数量,如图2(*a*)所示,截至2024年4月,大模型参数量已达4 000亿.

2.2　推理阶段大模型开销优化技术

庞大的参数量使得大模型的推理成本极高,因此平衡推理开销和模型精度的技术成为研究热点,其主要分为以下3类:

(1)模型压缩[6]:移除大模型中不重要的参数;



(*a*) 大模型参数量发展趋势　(*b*) 现有重训练技术在边缘设备上能支持的最大参数量

图2　大模型参数量的发展趋势和现有重训练技术能支持的最大参数量

(2)权重量化[15]:使用计算效率较高的8位或16位来存储原本为32位的权重;

(3)知识蒸馏[16]:将原始大模型的知识蒸馏到小模型中.

上述技术针对稳定输入数据分布设计,根据边缘端可用资源动态切换在该分布上提前生成好的多个待选小模型[16,17],或在推理时根据该分布中输入数据对同分布上提前生成的模型进行动态缩放[18,19],或根据云边端可用资源情况将推理任务卸载至云平台或边缘侧[20]. 因此,上述技术在面对新数据分布时精度会显著下降,不适用于重训阶段开销优化.

2.3　重训阶段大模型开销优化技术

图2(*b*)展示了在20 min重训窗口内保证模型重训练能收敛的前提下,全量重训练和参数高效微调2类现有重训练技术在4个不同边缘设备上能够支持的最大参数量. 全量重训练在目标数据集上训练模型的所有参数,不降低训练开销,因此仅能支持最大约12亿参数量的模型. 而参数高效微调技术[5]指出重训练中反向传播开销通常2倍于前向传播开销,为最大性能瓶颈,因此通过仅训练模型中一小部分关键参数来显著降低反向传播开销,广泛适用于各种大模型(其中提示词微调、指令微调等微调技术仅适用于对话型模型). 然而,该技术依然需要所有模型参数参与前向传播,使得前向传播成为新的性能瓶颈,因此在边缘设备上也仅能支持最大约20亿参数量的大模型,而图2(*a*)中最大的大模型的参数量是其数百倍.

全量重训练和参数高效微调技术均着眼于整个模型进行前向传播,训练模型中固定位置的参数,忽略了边缘端目标数据集的变化,从而无法自适应地将有限资源集中在最重要的模型参数上进行前向和反向传播,重训练开销大. 基于模型块粒度的重训练,即根据边缘端目标数据集动态鉴别出最重要模型块,并将计算资源集中于这些模型块的前向和反向传播上,可在保持高重训精度的同时降低重训开销.

3　BlockTrainer的设计

3.1　概览

BlockTrainer旨在支持云边场景下大模型块粒度重训,包含图3所示的3个设计目标.

(1)定量评估:大模型重训缩放定律(3.2节)定量评估不同重训方案(模型块压缩方案和重训迭代数)下的模型精度. 该模块自适应边缘侧当前目标数据集,在不泄露其数据隐私的情况下通过数据集统计特征和一个最困难样本拟合缩放定律公式参数.

(2)重训优化:重训方案优化器(3.3节)搜索能够在单位资源下带来最高重训精度提升的重训方案. 该优化器以最大化重训过程平均精度为优化目标,以重训窗口和可用内存为约束条件,将最优重训方案优化转换为组合优化问题,并使用二次时间复杂度启发式算法来快速求解.

(3)协同训练:云边协同大小模型训练器(3.4节)以最优重训方案为依据,将大模型中对精度贡献最大的模型块提取出来组成边缘端小模型,并构建大小模型神经元之间的索引,以支持边缘侧小模型重训以及将所学知识反馈回云平台大模型,实现大小模型协同训练.

基于以上模块,BlockTrainer通过如下4步完成一次大模型重训:①将边缘端当前目标数据集统计特征(均值和方差)和最困难数据样本传输至云平台;②使用这些特征对大模型重训缩放定律进行校准,并评估所有压缩方案的重训精度;③以评估结果为依据搜索能在单位资源下带来最高重训精度提升的重训方案;④使用最优重训方案在边缘端重训小模型,反馈知识更新云平台大模型.

3.2　大模型重训缩放定律

现有缩放定律研究大多聚焦于云平台大模型预训练场景[13,14]或迁移学习[21],仅考虑模型参数量、训练数据量和投入计算量三要素对模型预训练精度的影响[13,14]. 在云边协同训练场景下,当前工作存在以下局限性:(1)未在更细的粒度下考虑不同模型块对精度的重要性;(2)每次边缘端目标数据集发生变化时,都需要运行大量实验来重新拟合缩放定律公式中的参数,时间开销大;(3)拟合参数时需要将大量边缘侧目标数据集样本传输到云平台,隐私泄露风险大. 因此,BlockTrainer提出面向大模型块粒度重训的缩放定律,通过面向模型重训的缩放定律公式(3.2.1节)和基于神经网络的公式参数适配器(3.2.2节),建模了模型块参数量、重训迭代数、目标数据集分布和小模型初始化4个因素与重训精度之间的关系.

3.2.1　面向模型重训的缩放定律公式

重训缩放定律的输入包括2个部分:①压缩大模型以生成小模型时是否压缩第$k(1\leqslant k\leqslant N)$个模型块,用

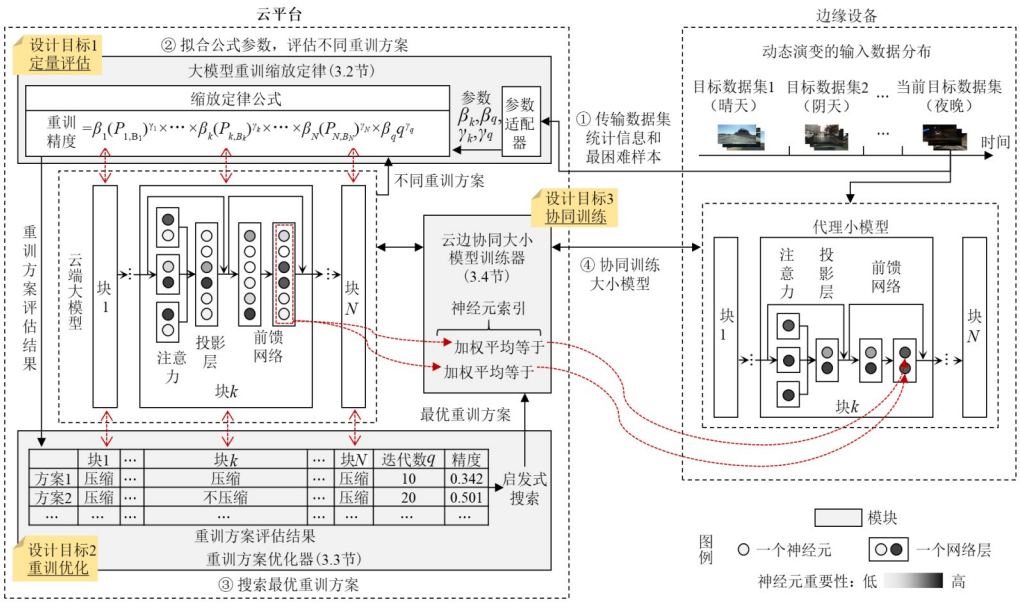


图3　BlockTrainer架构

布尔值B_k来表示，当B_k=1时表示压缩，压缩后该模型块的参数量定义为$P_{k,1}$，当B_k=0时表示不压缩，原始模型块的参数量表示为$P_{k,0}$；②重训迭代数q. 具体评估步骤为：首先计算每个输入对重训精度的影响，然后联合计算2个输入对重训精度的共同影响.

(1)模型块参数量对重训精度的影响. 当重训迭代数不变时，在模型的参数量与重训精度呈幂律关系的基础上[13]，每个模型块的参数量P_{k,B_k}与重训精度$A(B_k)$之间也呈幂律关系：

$$A(B_k)=\beta_k(P_{k,B_k})^{\gamma_k} \tag{2}$$

其中，β_k和γ_k为需要拟合的参数.

该关系的成立可通过DynaBERT[17]和LGViT[22]等大模型提前退出技术进行说明：通过在每个模型块之后增加可用于提前退出推理的任务头，可将模型中的一部分提取为单独的轻量模型(即模型块)，其参数量与重训精度呈幂律关系.

(2)重训迭代数对重训精度的影响. 当模型块参数量不变时，随着重训迭代数q的增加，重训精度$A(q)$快速上升后逐渐稳定，二者之间符合幂律关系：

$$A(q)=\beta_q q^{\gamma_q} \tag{3}$$

其中，β_q和γ_q为需要拟合的参数.

该关系的成立以现有模型训练调度器为支持，例如Optimus[23]也使用形如$A(q)=(\beta_0 q+\beta_1)^{-1}+\beta_2$的幂律公式来建模训练迭代数$q$与精度$A(q)$之间的关系.

综上，模型块参数量和重训迭代数均与重训精度之间均呈幂律关系. 将式(2)、式(3)使用乘法联立起来[14]，即可统一为一个缩放定律公式：

$$A(B_1,B_2,\cdots,B_N,q)=\prod_{k=1}^{N}\beta_k\ (P_{k,B_k})^{\gamma_k}\times\beta_q q^{\gamma_q} \tag{4}$$

当模型块参数量不变时，式(4)中的$\prod_{k=1}^{N}\beta_k\ (P_{k,B_k})^{\gamma_k}$退化为常数项，等价于式(3)；当重训迭代数和第$1,2,\cdots,i-1,i+1,\cdots,N$个模型块的参数量均不变时，式(4)中除了$\beta_i(P_{i,B_i})^{\gamma_i}$之外的项均退化为常数项，等价于式(2).

3.2.2　基于神经网络的公式参数适配器

每当边缘侧目标数据集和模型初始化参数发生变化时，上述缩放定律公式中的参数$\beta_1,\beta_2,\cdots,\beta_N,\beta_q$和$\gamma_1,\gamma_2,\cdots,\gamma_N,\gamma_q$的值都需要重新拟合. BlockTrainer利用神经网络拟合能力强大、拟合速度快的特点，根据目标数据集和模型初始化参数2个因素快速拟合缩放定律公式参数. 具体来说，该参数适配器的输入分为2个部分.

(1)边缘端目标数据集特征的均值和方差[24]和目标数据集中的最困难样本，即信息熵最大样本[25]. 对比其他样本，该样本可代表当前数据集最多的信息. 通过将该样本传输至云端，可以在尽量减少泄露隐私的情况下，尽可能地保留边缘侧数据集信息以生成最相关的小模型.

(2)大模型在边缘侧最困难样本上的损失函数值，其可代表模型初始化参数的好坏. 这是因为小模型是从大模型中动态生成的，所以大模型损失函数值与小模型损失函数值呈正相关，该值越小代表小模型初始化参数越好. 参数适配器将上述输入信息展开并连接为一维向量$\boldsymbol{I}$，使用体积小于10 MB的3层全连接网络在1 s内对其进行处理，最后输出包含公式参数值的向量$\boldsymbol{O}=[\beta_1,\beta_2,\cdots,\beta_N,\beta_q,\gamma_1,\gamma_2,\cdots,\gamma_N,\gamma_q]$.

BlockTrainer参数适配器中的神经网络需要在离线阶段使用若干个数据点进行一次训练. 每个数据点的生

成过程如下：①将随机数据增强技术（如针对图像的旋转、缩放、色调改变、亮度调整和背景替换，针对文本的词汇替换、插入、删除）应用到源数据集上，生成一个模拟的目标数据集；②随机选取部分模型块进行压缩以生成小模型；③在模拟目标数据集上重训该小模型若干个迭代；④统计以上步骤对应的向量$\boldsymbol{I}$和$\boldsymbol{O}$，形成一个数据点$(\boldsymbol{I},\boldsymbol{O})$. 重复该过程多次，即可生成用于训练参数适配器的数据集，随后使用随机梯度下降（Stochastic Gradient Descent，SGD）算法对参数适配器进行训练.

3.3 重训方案优化器

3.3.1 利用重训缩放定律评估结果

BlockTrainer优化器寻找能够在有限重训窗口$t^{\max}$和可用内存$m^{\max}$下最大化重训精度的方案：每个模型块是否被压缩$B_1,B_2,\cdots,B_N$和训练迭代数q. 将其转换为组合优化问题：

$$\max_{B_1,B_2,\cdots,B_N,q}\frac{1}{q}\sum_{i=1}^{q}A(B_1,B_2,\cdots,B_N,i) \tag{5}$$

$$\text{s. t.}$$

$$\forall 1\leqslant k\leqslant N,B_k\in\{0,1\} \tag{6}$$

$$q\cdot\sum_{k=1}^{N}T_{k,B_k}\leqslant t^{\max} \tag{7}$$

$$\sum_{k=1}^{N}M_{k,B_k}\leqslant m^{\max} \tag{8}$$

其中，$T_{k,0}$和$T_{k,1}$分别为第k个模型块未被压缩和被压缩时训练一个迭代所需的时间；$M_{k,0}$和$M_{k,1}$分别为第k个模型块未被压缩和被压缩时训练所占用的内存.

式(6)表示每个模型块要么压缩要么不被压缩，每个模型块的压缩配置（例如压缩率和压缩算法）需预先确定且保持一致；式(7)计算了q个重训迭代中所有模型块训练时间的总和，即重训时间，需要小于重训窗口$t^{\max}$；式(8)计算了所有模型块所占内存之和，即重训所占内存，需要小于当前边缘侧可用内存$m^{\max}$.

3.3.2 基于启发式搜索的快速求解算法

对于某个重训方案$B_1,B_2,\cdots,B_N,q$，有两类操作可以将其修改为消耗更多重训窗口但重训精度更高的新方案.

(1)保持训练迭代数不变，将某个原本将被压缩的第k个模型块变为不压缩. 经过该操作后，新方案相较于旧方案的重训精度提升为$\frac{1}{q}\sum_{i=1}^{q}A(\cdots,B_{k-1},0,B_{k+1},\cdots,i)-A(\cdots,B_{k-1},1,B_{k+1},\cdots,i)$，额外消耗的重训窗口为$q(T_{k,0}-T_{k,1})$，额外消耗的内存为$M_{k,0}-M_{k,1}$.

(2)保持压缩方案不变，增加Δq个训练迭代（如10个）. 经过该操作后，新方案相较于旧方案的重训精度提升为$\frac{1}{\Delta q}\sum_{i=q}^{q+\Delta q}A(B_1,B_2,\cdots,B_N,i)$，额外消耗的重训窗口为$\Delta q\cdot\sum_{k=1}^{N}T_{k,B_k}$，额外消耗的内存为0. 本文将能在重训窗口和内存约束内、在单位重训窗口消耗下带来最大重训精度提升的操作定义为最优操作.

基于这两类操作，以初始方案$(B_1=B_2=\cdots=B_N=1,q=0)$为起点，BlockTrainer进行多轮搜索，每轮搜索通过3个步骤来寻找最优操作以将当前方案修改为更优方案：①检查每个操作是否符合重训窗口和内存约束，计算每个操作在单位重训窗口消耗下带来的重训精度提升；②选择最优操作并将当前方案修改为更优方案，同时更新可用重训窗口和内存；③检查重训窗口是否消耗完毕，若消耗完毕，则搜索结束，输出此时的方案作为最优重训方案；否则回到第1步继续搜索.

3.3.3 时间复杂度分析

该算法最多运行$\left(t^{\max}/\Delta q\right)+N$轮，每轮中最多运行$N+1$次循环以枚举$N$种将某个未压缩模型块进行压缩的情况和1种将重训迭代数增加Δq的情况，因此该算法的时间复杂度为$O\left[(N+1)\left(\frac{t^{\max}}{\Delta q}+N\right)\right]$，为二次多项式时间算法，仅占枚举所有方案的极小百分比（小于0.01%）即可得到最优方案. 例如，若模型块数量为20，迭代数可选范围为10、20、…、100，该快速求解算法仅需枚举所有情况的0.006%（总共有$2^{20}\times10$种情况，快速求解算法仅需枚举最多(20+10)×(20+1)=630种情况）即可.

3.4 云边协同大小模型训练器

基于最优重训方案，BlockTrainer训练器首先从云平台大模型中选出与重训精度最相关的部分神经元来组成最优小模型，然后将该小模型传输到边缘端重训练. 在重训练完成后，将小模型所学知识反馈回云平台大模型，实现大小模型协同训练. 以上训练过程以大小模型之间构建的神经元索引为基础.

3.4.1 神经元索引

神经元是大模型中每个网络层的基本组成单位，每个网络层包括成百上千个神经元，以不同的特征提取函数对该层的输入数据进行处理. 由于小模型以更少的神经元实现了与大模型类似的特征提取功能，每个小模型神经元的特征提取函数是多个大模型神经元的特征提取函数的组合. 神经元的特征提取函数由其权重矩阵决定，因此每个小模型神经元$\boldsymbol{\theta}^{\text{small}}$的权重矩阵可近似表示为大模型中同一层所有神经元$\left\{\boldsymbol{\theta}_1^{\text{large}},\boldsymbol{\theta}_2^{\text{large}},\cdots,\boldsymbol{\theta}_i^{\text{large}},\cdots,\boldsymbol{\theta}_N^{\text{large}}\right\}$的权重矩阵的线性组合（即加权平均）：

$$\boldsymbol{\theta}^{\text{small}}=\sum_{i}\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_i^{\text{large}})\cdot\boldsymbol{\theta}_i^{\text{large}} \tag{9}$$

BlockTrainer将用来做加权平均的权重矩阵的线性组合$\left\{\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_1^{\text{large}}),(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_2^{\text{large}}),\cdots,\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_i^{\text{large}}),\cdots,\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_N^{\text{large}})\right\}$定义为小模型神经元$\boldsymbol{\theta}^{\text{small}}$和大模型中同

一层所有神经元$\{\boldsymbol{\theta}_1^{\text{large}},\boldsymbol{\theta}_2^{\text{large}},\cdots,\boldsymbol{\theta}_i^{\text{large}},\cdots,\boldsymbol{\theta}_N^{\text{large}}\}$之间的神经元索引.$\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_i^{\text{large}})$越大,说明小模型神经元$\boldsymbol{\theta}^{\text{small}}$与大模型神经元$\boldsymbol{\theta}_i^{\text{large}}$的相关性越高.

3.4.2　精度感知的小模型动态生成

根据重训方案优化器输出的模型块压缩方案,BlockTrainer对大模型中需要压缩的模型块进行压缩来生成小模型,包括4步:

(1)预先向大模型中每个网络层插入1个特征激励压缩模块[18],其由1个池化层、线性层和ReLU激活函数层组成;

(2)将边缘端当前目标数据集的最困难样本输入大模型进行前向传播,当前向传播进行到某个层时,该层的特征压缩激励模块根据该层的输入数据输出1个一维向量,向量中每个值即代表该层每个神经元对于正确推理输入样本的重要性;

(3)根据重训方案优化器输出的模型块压缩方案,优先从无需压缩的(即最重要的)大模型块中选择最重要神经元来组成最优小模型;

(4)为每个小模型神经元和与之同一层的大模型神经元之间建立一个神经元索引.

3.4.3　小模型反馈所学知识给大模型

动态生成的小模型在新分布的数据上进行重训练.重训练完成后,小模型所学知识即小模型重训练前后其所有神经元权重矩阵的差值.令小模型神经元$\boldsymbol{\theta}^{\text{small}}$重训练前后的权重矩阵的差值为$\Delta\boldsymbol{\theta}^{\text{small}}$,BlockTrainer根据神经元索引将该知识存储到大模型同一层的所有神经元$\{\boldsymbol{\theta}_1^{\text{large}},\boldsymbol{\theta}_2^{\text{large}},\cdots,\boldsymbol{\theta}_i^{\text{large}},\cdots,\boldsymbol{\theta}_N^{\text{large}}\}$中:

$$\begin{aligned}
\boldsymbol{\theta}_1^{\text{large}} &\leftarrow \boldsymbol{\theta}_1^{\text{large}}+\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_1^{\text{large}})\cdot\Delta\boldsymbol{\theta}^{\text{small}}\\
\boldsymbol{\theta}_2^{\text{large}} &\leftarrow \boldsymbol{\theta}_2^{\text{large}}+\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_2^{\text{large}})\cdot\Delta\boldsymbol{\theta}^{\text{small}}\\
&\cdots\\
\boldsymbol{\theta}_i^{\text{large}} &\leftarrow \boldsymbol{\theta}_i^{\text{large}}+\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_i^{\text{large}})\cdot\Delta\boldsymbol{\theta}^{\text{small}}\\
&\cdots\\
\boldsymbol{\theta}_N^{\text{large}} &\leftarrow \boldsymbol{\theta}_N^{\text{large}}+\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_N^{\text{large}})\cdot\Delta\boldsymbol{\theta}^{\text{small}}
\end{aligned}\tag{10}$$

即,若某个大模型神经元$\boldsymbol{\theta}_i^{\text{large}}$与该小模型神经元$\boldsymbol{\theta}^{\text{small}}$的相关性$\Gamma(\boldsymbol{\theta}^{\text{small}},\boldsymbol{\theta}_i^{\text{large}})$越高,其就存储更多来自该小模型神经元的知识.

4　基于Hugging Face的BlockTrainer实现

BlockTrainer使用PyTorch 1.8.0开发,代码量约8 000行,适用于运行Linux操作系统的边缘设备,支持以Transformer为代表的大模型(图4).对于每个接入BlockTrainer的模型,需要实现4类接口:(1)对输入样本进行推理;(2)在指定数据集上测试精度;(3)对模型块进行压缩;(4)对模型结构进行操作(例如修改、替换某个网络层).

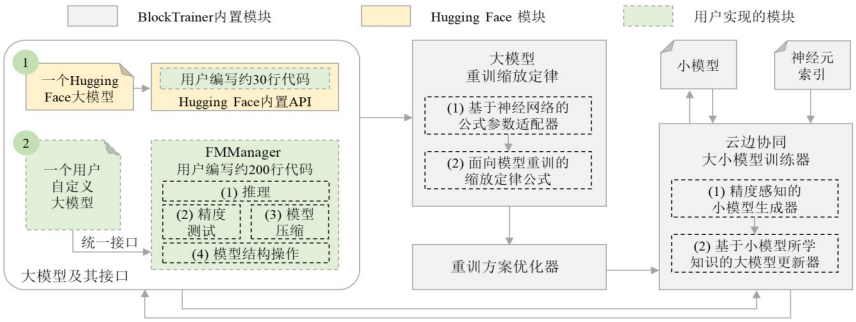


图4　BlockTrainer实现

4.1　Hugging Face开源社区原生大模型

BlockTrainer使用Hugging Face[26]内置应用程序接口(如模型基类AutoModel)来实现以上4类接口,例如,使用AutoModel. forward(·)函数实现对输入样本的推理,使用AutoModel. prune_heads(·)函数压缩模型块中的注意力层等.因此,用户仅需额外补充约30行代码即可将一个Hugging Face模型接入BlockTrainer.

4.2　用户自定义大模型

BlockTrainer构建抽象类FMManager以解耦和桥接模型具体实现和BlockTrainer内部模块.接入一个自定义大模型,用户需要编写约200行代码来完整实现FMManager中的4类抽象接口.

4.3　系统评测工作负载

本文选取了4个代表性的计算机视觉和自然语言处理应用,基于HuggingFace模型构建边缘端重训工作负载.

(1)图像分类.该负载的应用是判断一张输入图片属于哪一个类别,精度评测指标为top-1精度.本文选取了参数量为0.9亿(0.09 B)的计算机视觉大模型ViT-B/16[27],使用通用物体数据集ImageNet(包括超过1 000 000张图片)和合成交通标志数据集SYNSIGNS(包括64 000张图片)作为源数据集,使用通用物体数据集Caltech256(包括30 000张图片)、DomainNet(包括600 000张图片)和真实交通标志数据集GTSRB(包括39 209张图片)作为目标数据集.所有数据集中图片尺寸均统一为224像素×224像素,特征数量为224×224=50 176.本

文取所有数据集类别的交集(139个类别)作为模型需要处理的类别. 该负载在可用内存为16 GB的NVIDIA Xavier NX设备上运行.

(2)语义分割. 该负载的应用是判断一张输入图片中每个像素点属于哪一个类别,常用于自动驾驶领域,精度评测指标为平均交并比(mean Intersection over Union,mIoU). 本文选取了参数量为2亿(0.20 B)的计算机视觉大模型SwinV2-Large[9],使用合成自动驾驶数据集GTA5(包括24 966张图片)和人像数据集SuperviselyPerson(包括5 711张图片)作为源数据集,使用真实自动驾驶数据集CityScapes(包括5 000张图片)和人像数据集BaiduPerson(包括3 447张图片)作为目标数据集. 所有数据集中图片尺寸均统一为224像素×224像素,特征数量为224×224=50 176. 本文取所有数据集类别的交集(21个类别)作为模型需要处理的类别. 该负载在可用内存为32 GB的NVIDIA AGX Xavier设备上运行.

(3)词性标注. 该负载的应用是判断一个句子中每个单词的词性(如名词、动词、形容词等),精度评测指标为top-1精度. 本文选取了参数量为13亿(1.3 B)的自然语言处理大模型OPT[28],使用文本数据集HL5Domains(包括1 184个句子)作为源数据集,使用文本数据集Liu3Domains(包括620个句子)、Ding9Domains(包括1 684个句子)和SemEval14(包括3 253个句子)作为目标数据集. 每个句子平均包括21.6个单词(即特征),所有数据集共覆盖了43种不同词性. 该负载在可用内存为32 GB的NVIDIA AGX Orin设备上运行.

(4)任务型对话. 该负载的应用是扮演机器人与人类进行对话,是目前自然语言处理大模型最流行、最热门的应用(如ChatGPT),精度评测指标为BLEU4. 本文选取了参数量为30亿(3 B)的自然语言处理大模型LLaMA[29],使用文本数据集MultiWoz -MultiDomains(包括113 556个句子)作为源数据集,使用文本数据集MultiWoz-Taxi(包括16 998个句子)和MultiWoz-Hotel(包括17 253个句子)作为目标数据集. 每个句子平均包括13.1个单词(即特征),所有数据集共包括23 689个不同的单词. 该负载在可用内存为64 GB的NVIDIA AGX Orin设备上运行.

针对每类应用,为构造更加符合边缘端输入数据分布持续演变的场景,我们构建了长度为30的目标数据集序列,其中目标数据集随机出现. 对于前2个图像分类应用,因模型较小,将其在每个目标数据集上的重训窗口设置为10 min;对于后2个应用,重训窗口设置为20 min. 此外,实验使用一台搭载NVIDIA Quadro RTX 8000显卡(显存48 GB),内存为256 GB的服务器作为云平台.

进一步地,我们基于BlockTrainer用户自定义模型接口,实现了3个更大规模的大模型,以评测设备资源和重训窗口约束下边缘侧所能支持的最大模型. ①Qwen:国产大模型,由阿里巴巴发布,拥有70亿(7 B)参数,重训窗口设置为30 min. ②GPT-NeoX:由EleutherAI发布的GPT开源版本,拥有200亿(20 B)参数,重训窗口设置为45 min. ③DeepSeek:国产大模型,由幻方量化发布,拥有330亿(33 B)参数,重训窗口设置为60 min.

5 实验评测

基于真实边缘设备平台,本节首先评测BlockTrainer中3个模块的性能(如重训缩放定律预测准确度)和开销(5.1节);然后,在不同边缘侧应用场景下,对比BlockTrainer和最新的8个大模型微调方法重训性能(5.2节);最后,讨论本文方法对于不同无监督重训练算法的适用性,以及在主流边缘设备所能支持的最大模型(5.3节).

5.1 BlockTrainer模块测试

5.1.1 大模型重训缩放定律评测

5.1.1.1 实验设置

对于表1的每个工作负载,以8∶2的比例将用于训练缩放定律参数适配器的数据点划分为训练集和测试集,在训练集上训练参数适配器,在测试集上比较BlockTrainer重训缩放定律和云平台预训练缩放定律的预测精度,其中云平台预训练缩放定律[13]的公式定义为

$$A(p,q)=\beta_p p^{\gamma_p}+\beta_q q^{\gamma_q} \tag{11}$$

其中,$A(p,q)$为精度,p为模型参数量,q为重训迭代数). 在源数据集上使用L-BGFS算法和网格搜索来拟合该缩放定律公式中的参数β_p、β_q、γ_p和γ_q.

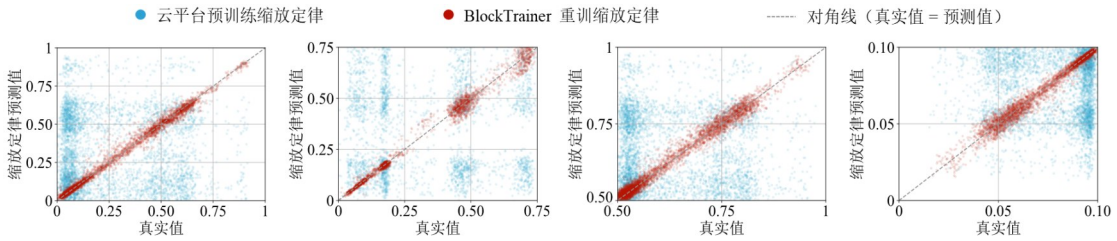
随后,在真实目标数据集上以3组随机重训方案进行训练,展示真实的精度曲线和2类缩放定律预测的精度曲线.

5.1.1.2 实验结果

图5展示了缩放定律预测值与真实模型的分布情况,其中每个点代表一对预测值和对应的真实值,该点越靠近图中对角线,说明缩放定律预测误差越小. 可观察到,BlockTrainer重训缩放定律几乎所有的预测点都靠近对角线分布,而云平台预训练缩放定律的预测点分布散乱,预测误差很大. 进一步地,由图6可知,在各种工作负载和重训配置下,BlockTrainer缩放定律预测的重训精度曲线都与真实精度曲线高度贴合. 这是因为BlockTrainer缩放定律全面考虑了模型重训诸多影响要素,如不稳定的输入数据分布、小模型初始参数等,并实时根据当前输入数据对模型公式进行拟合校正. 与之对比,云平台预训练缩放定律中的参数在源数据集上确定,只适用于输入数据和模型初始化固定的大模型预训练场景,在本实验场景中平均绝对误差(预测值与真实值之差)达到16.11%,平均相对误差(绝对误差除以真实值)达到128.13%.

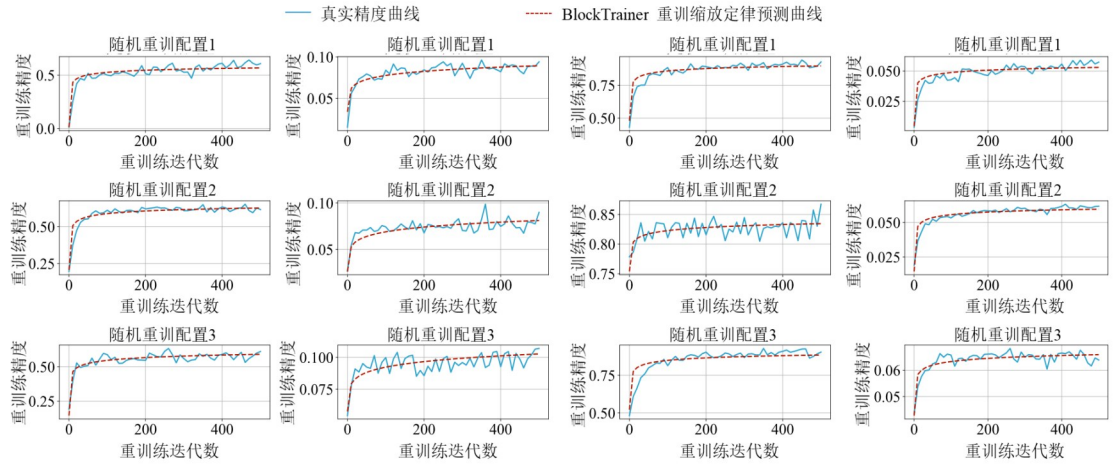
表1　工作负载设置

工作负载	模型	参数量	源数据集	目标数据集	重训窗口/min	边缘设备	精度指标
图像分类	ViT-B/16	0.9亿(0.09 B)	ImageNet, SYNSIGNS	Caltech256,DomainNet, GTSRB	10	Xavier NX 16 GB	top-1
语义分割	SwinV2-Large	2亿(0.20 B)	GTA5, SuperviselyPerson	CityScapes, BaiduPerson	10	AGX Xavier 32 GB	mIoU
词性标注	OPT	13亿(1.3 B)	HL5Domains	Liu3Domains, Ding9Domains,SemEval14	20	AGX Orin 32 GB	top-1
任务型对话	LLaMA	30亿(3 B)	MultiWoz-MultiDomains	MultiWoz-Taxi, MultiWoz-Hotel	20	AGX Orin 64 GB	BLEU4



(*a*) ViT-B/16 图像分类应用　(*b*) SwinV2-Large 语义分割应用　(*c*) OPT 词性标注应用　(*d*) LLaMA 任务型对话应用

图5　缩放定律预测值与真实值的分布情况



(*a*) ViT-B/16 图像分类应用　(*b*) SwinV2-Large 语义分割应用　(*c*) OPT 词性标注应用　(*d*) LLaMA 任务型对话应用

图6　BlockTrainer缩放定律应用示例

5.1.1.3　实验结果总结

BlockTrainer 重训缩放定律的平均绝对误差为0.12%,平均相对误差为3.67%,对比云平台预训练缩放定律误差平均降低94.45%.

5.1.2　重训方案优化器评测

5.1.2.1　实验设置

实验评测了BlockTrainer重训方案优化器的时间开销和重训精度,并与2类基准优化方法对比.

(1)n次基于全量重训练的随机搜索. 随机选取n种不同的随机重训方案进行完整的重训练,并选取平均重训精度最高的重训方案. 理论上,n越大,搜索效果会越好,但时间开销也越大.

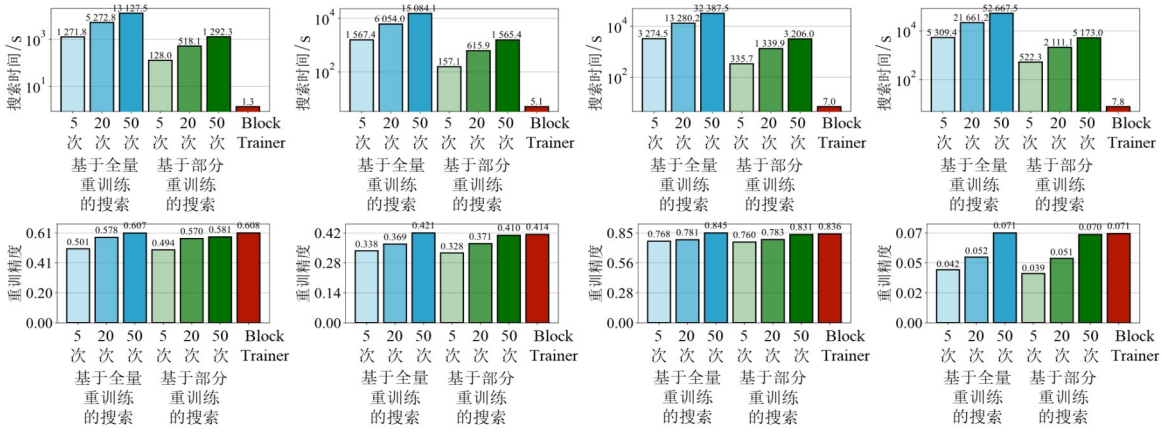
(2)n次基于部分重训练的随机搜索[2]. 随机选取n种不同的随机重训方案并重训练仅10%的迭代数,选取平均重训精度最高的重训方案. 该方法在搜索精度相似的情况下搜索速度更快.

需要注意的是,这2类优化方法的搜索结果能代表最好的重训搜索方案,但它们需要大量的边缘侧数据,因此可能在云平台造成数据隐私泄露.

5.1.2.2　实验结果

图7的第1行和第2行分别展示了所有优化方式在4个工作负载上的时间开销与重训精度. 可观察到,2个对

比方法的搜索时间远远超出了可用重训窗口，而BlockTrainer的搜索时间仅需数秒，并达到了与基准方法类似的重训精度. 这是因为BlockTrainer在搜索时利用了轻量但准确的大模型缩放定律评估结果，通过启发式方法在加速搜索的同时保证了搜索精度. 与之对比，基准方法依赖重训练的方式搜索最优方案，需要模型加载、数据加载、工作负载初始化等步骤. 这些步骤单次执行的时间即为8.73 s，而整个搜索时间往往长达上百秒.



(a) ViT-B/16 图像分类应用 (b) SwinV2-Large 语义分割应用 (c) OPT 词性标注应用 (d) LLaMA 任务型对话应用

图7 不同重训方案搜索算法的时间消耗和重训精度

5.1.2.3 实验结果总结

BlockTrainer重训方案优化器的平均搜索时间为5.30 s(仅占用重训窗口的0.46%)，相较于基准方法，搜索速度提升1 646.36倍，重训精度提升14.04%.

5.1.3 云边协同大小模型训练器评测

5.1.3.1 实验设置

实验评测了云边协同大小模型训练器中3个阶段的时间开销：(1)大模型对最困难样本进行前向传播以计算神经元重要性；(2)提取最重要神经元组成小模型；(3)小模型将所学知识反馈至大模型. 并将这些时间开销与重训窗口对比. 在评测过程中，我们首先通过CUDA提供的同步功能torcu. cuda. synchronize(·)来避免GPU异步计算导致的测量不准，之后通过CUDA计时器torch. cuda. Event(·)来测量每个阶段开始和结束之间的GPU时钟时间.

5.1.3.2 实验结果

表2展示了4个工作负载上的大小模型协同训练器时间开销. 可观察到，小模型生成和反馈的开销随大模型参数量的增加而增加，但均控制在数秒内，不会对边缘侧重训练的进度产生影响. 云边协同大小模型训练器3个阶段具体开销如下：(1)计算大模型神经元重要性仅需大模型对单个最困难的数据样本进行前向传播，完成单次模型推理；(2)提取最重要神经元涉及对大模型权重矩阵进行切片和拷贝，需要执行内存读写和申请分配操作；(3)小模型反馈所学知识需要利用神经元索引，将小模型权重和大模型权重进行加法和乘法运算. 综上，BlockTrainer操作均不涉及反向传播等复杂运算，因此即使在最大模型(任务型对话模型)上，占用也仅为重训窗口的0.94%.

表2 大小模型协同训练器时间开销 单位：s

工作负载	计算大模型神经元重要性	提取最重要神经元组成小模型	小模型反馈所学知识	总和
图像分类	0.049 2	0.861 1	1.512 3	2.422 6
语义分割	0.081 9	1.886 9	2.314 7	5.283 5
词性标注	0.101 8	2.712 0	3.338 9	6.152 7
任务型对话	0.159 7	6.075 3	5.121 9	11.356 9

5.1.3.3 实验结果总结

BlockTrainer大小模型协同训练器平均用时为6.303 9 s，仅占用重训窗口的0.58%.

5.2 边缘侧大模型重训对比实验

5.2.1 实验设置

5.2.1.1 基准方法

实验将BlockTrainer与最先进的重训练方法进行比较，包括3类全量重训练基准方法：(1)压缩大模型后直接在目标数据集上训练其所有模型参数——LoSparse[6]；(2)压缩大模型后先在源数据集上调优，再在目标数据集上训练其所有模型参数——DistilBERT[7]、TinyBERT[8]；(3)压缩大模型后先在源数据集上调优，然后直接在目标数据集上使用——FLAP[30]、SliceGPT[31]. 同时，将重训方法LoSparse、DistilBERT和TinyBERT与参数高效微调算

法 LoRA[5]结合,构建基准方法 LoSparse+LoRA、DistilBERT+LoRA 和 TinyBERT+LoRA.

5.2.1.2　云平台大模型预训练

表 1 中的 4 个工作负载模型,首先在云平台上使用源数据集进行预训练,其训练迭代数为 80 000,学习率分别为 3×10^{-4}、3×10^{-4}、1×10^{-4}、1×10^{-4},批处理大小分别为 128、16、8、4.

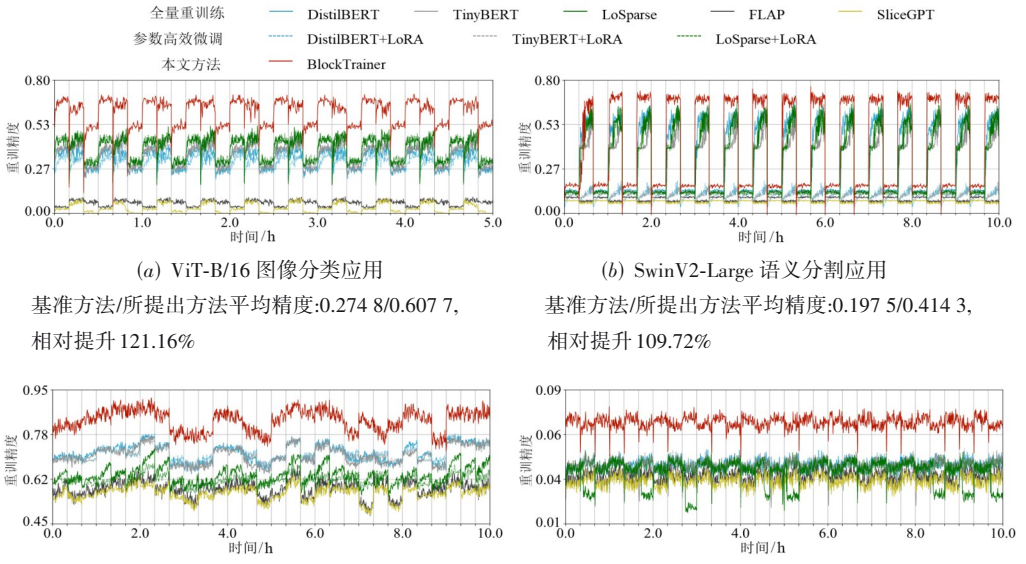
5.2.1.3　边缘侧小模型重训练

基于边缘侧重训练基准测试集[32],自动构建持续变化的目标数据集序列,并为被测试方法自动搜索重训练超参数. 在对比实验中,BlockTrainer 生成的小模型和基准方法重训的小模型有相同的参数量. 同时,由于边缘场景下实时获取的训练数据通常没有标签,实验采用经典无监督领域自适应算法,通过特征对齐进行模型重训.

5.2.2　固定重训窗口下的重训精度

5.2.2.1　实验结果

图 8 展示了 4 个工作负载上 BlockTrainer 和基准方法的重训精度,子图中的垂直网格线代表目标数据集发生切换的时间点. 可观察到,BlockTrainer 在所有工作负载和目标数据集都取得了最高的重训精度. 这是因为:(1)BlockTrainer 通过预测不同模型块对重训精度贡献,搜索在单位资源下带来最高重训精度的重训方案,例如图 8(*b*)中,基准方法需要一定时间后才能让精度显著提升,而 BlockTrainer 能够在重训窗口初期就显著提高精度;(2)BlockTrainer 根据当前目标数据集从大模型中动态生成小模型,最大限度地保留了原始模型的泛化能力;(3)BlockTrainer 的神经元索引使小模型所学知识可快速存储到大模型中,实现大小模型协同重训和知识积累,进一步提高重训精度,例如图 8(*b*)中,BlockTrainer 在第 4 个目标数据集上的精度明显高于前 2 个目标数据集. 与之对比,基准方法的低重训精度主要原因是它们根据源域数据集对大模型所有块进行压缩,显著降低了大模型泛化能力,生成的小模型没有足够的知识容量来学习新到达的目标数据集. FLAP 和 SliceGPT 这类不在目标数据集上进行重训的方法取得了最低的精度,这证明了在动态输入数据分布下对模型进行重训的必要性.



(*a*) ViT-B/16 图像分类应用

基准方法/所提出方法平均精度:0.274 8/0.607 7,相对提升 121.16%

(*b*) SwinV2-Large 语义分割应用

基准方法/所提出方法平均精度:0.197 5/0.414 3,相对提升 109.72%

(*c*) OPT 词性标注应用

基准方法/所提出方法平均精度:0.655 6/0.835 8,相对提升 27.50%

(*d*) LLaMA 任务型对话应用

应用基准方法/所提出方法平均精度:0.042 4/0.070 6,相对提升 66.58%

图 8　固定重训窗口下的重训精度

5.2.2.2　总结

BlockTrainer 在相同计算资源(即重训窗口)下相对提升了 81.24% 的重训精度,或在同样重训精度下平均降低了 92.15% 的计算资源消耗.

5.2.3　不同重训窗口下的重训精度

5.2.3.1　实验设置

在 5.2.1 节实验设置的基础上,进一步探究当工作负载中重训窗口的长度变短时所有方法重训精度的变化情况. 当重训窗口长度为 0 时,所有方法都不能进行重训练,此时直接测量未重训小模型在目标数据集上的精度.

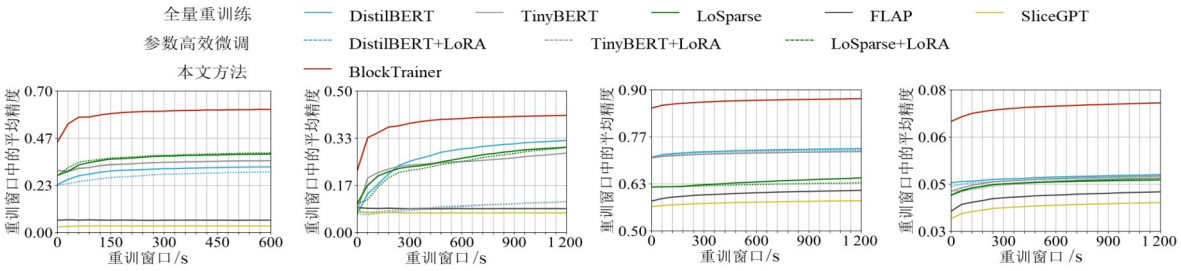
5.2.3.2　实验结果

图 9 展示了不同重训窗口下每个方法的平均重训精度. BlockTrainer 在不同的重训窗口,甚至在不进行重训时依然取得了最高的重训精度,这进一步说明了 Block-

Trainer生成的小模型保留了与当前目标数据集最相关的模型块,同时具备更好的泛化能力. 与之对比,基准方法即使消耗了数倍于BlockTrainer的重训时间,精度仍然较低,这是因为其小模型缺乏足够针对新目标数据集的学习和泛化能力.

5.2.3.3 总结

在所有重训窗口设置下,BlockTrainer相对提升了86.25%的重训精度;特别地,当重训窗口小于原始重训窗口的20%时,BlockTrainer相对提升了97.13%的重训精度.



(*a*) ViT-B/16 图像分类应用 (*b*) SwinV2-Large 语义分割应用 (*c*) OPT 词性标注应用 (*d*) LLaMA 任务型对话应用

图9 不同重训窗口下的平均重训精度

5.3 讨论

5.3.1 无监督重训练算法适用性

除了经典的特征对齐无监督算法外,BlockTrainer还可与其他无监督重训练算法相结合以获取更高的重训精度. 本实验选取图像分类工作负载,将BlockTrainer和最佳基准方法LoSparse、SHOT[25]结合,并在图10中展示了二者的重训精度. 结果显示,BlockTrainer适用于不同模型重训算法,并将其重训精度提升了101.34%.

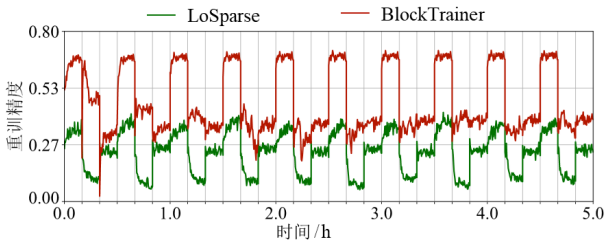
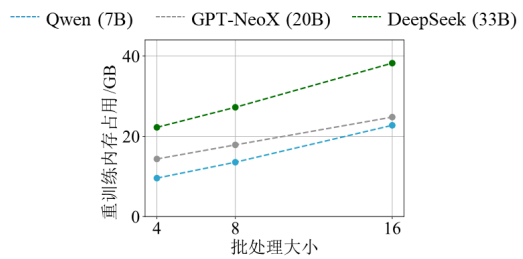


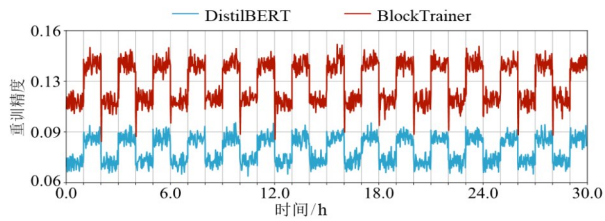
图10 BlockTrainer与无监督重训算法SHOT的结合

5.3.2 BlockTrainer可支持最大模型

本实验探讨在资源受限边缘设备NVIDIA AGX Orin(64 GB)上BlockTrainer能够支持的最大模型参数量,来证明BlockTrainer对当今百亿参数量大模型的适用性. 图11(*a*)展示了BlockTrainer可在边缘设备上以16的批处理大小顺利运行最大至330亿参数量的大模型(此时重训内存占用接近边缘设备可用内存45 GB),是现有最先进重训练方法(如图2(*b*)所示)的16.5倍. 进一步地,图11(*b*)展示了BlockTrainer在330亿参数量大模型上也能获取更高的重训精度. 当前BlockTrainer支持模型参数量的上限,取决于模型网络架构和所采取的压缩技术,即边端小模型体积无法低于云端大模型体积的某个百分比(例如,ViT-B为1.45%,SwinV2-Large为1.14%).



(*a*) BlockTrainer中边缘端重训练内存占用



(*b*) BlockTrainer和DistilBERT在DeepSeek(33 B)上的重训精度

图11 BlockTrainer在更大模型上的内存占用和重训精度

5.3.3 不同大模型适用性

BlockTrainer对于一个大模型的适用性取决于该模型是否由多个模型块组成,以及模型的基本操作单位(如Transformer中的神经元). 当前典型的大模型类型包括:(1)自然语言处理大模型和多模态大模型,由Transformer模型块组成;(2)基于CNN或扩散模型的大模型[10,12],由类似ResBlock或ConvBlock的模型块组成,其基本操作单位为等价于神经元的卷积滤波器;(3)基于LSTM的大模型,由LSTMCell模型块组成,其基本操作单位为神经元.

5.3.4 与不同模型块压缩算法结合

BlockTrainer可以与不同的模型块压缩技术结合.我们以图像分类工作负载为例测试了3种不同模型块压缩技术:(1)基于特征激励压缩模块(FBS)的动态压缩技术(现采用的技术);(2)Dejavu[33],基于2个不同的全连接网络,来分别预测模型块中最不重要的注意力头和神经元并丢弃;(3)LLM in a flash[34],基于低秩预测器来预测输出为0的神经元并丢弃.如图12所示,BlockTrainer在采用不同压缩技术时的精度差距很小(小于1.83%).这是因为本文方法在不同压缩方法下,均能精准预测不同模型块及其压缩方案对精度的影响,从而生成最优的重训方案.

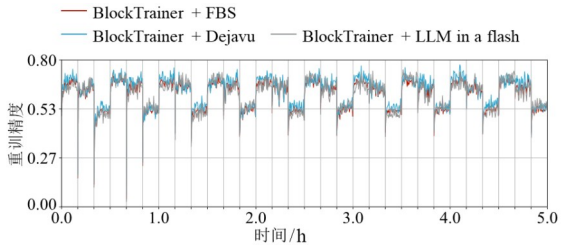


图12　BlockTrainer与不同模型块压缩算法的结合

5.3.5 对比直接重训练大模型

本实验选取图像分类工作负载,对比BlockTrainer和直接在边缘端重训练云端大模型的精度和时间开销.实验结果显示,在相同计算资源下,直接重训云端大模型需288 h收敛,是本文云边协同方法加速模型重训时间的3 457倍,仅损失9.92%精度;在相同计算资源和重训窗口约束(10 min)下,本文方法能够重训练更多迭代,提升了21.13%的模型精度.

6 结论

本文提出了BlockTrainer——一个模型块粒度高效重训方法,旨在支持云边协同场景下大模型高效重训.BlockTrainer的关键思想是从更细粒度探究大模型重训练问题,引入重训缩放定律,在输入数据动态变化的边缘侧场景准确精准预测不同模型块对重训精度的重要性差异.以此为基础,提出启发式重训方案搜索方法,在综合考虑各类精度影响因素的情况下快速搜索最优重训方案,从云平台大模型中提取与当前目标数据集精度最相关的小模型,实现大小模型协同训练.基于Hugging Face开源社区实现BlockTrainer系统,并在多个边缘侧应用场景下验证了其对模型精度的提升、低运行开销以及广泛适用性.

参考文献

[1] BOMMASANI R, HUDSON D A, ADELI E, et al. On the opportunities and risks of foundation models[EB/OL]. (2021-07-12)[2024-06-04]. https://arxiv.org/abs/2108.07258v3.

[2] BHARDWAJ R, XIA Z X, ANANTHANARAYANAN G, et al. Ekya: Continuous learning of video analytics models on edge compute servers[EB/OL]. (2020-12-19)[2024-06-04]. http://arxiv.org/abs/2012.10557.

[3] KHANI M, ANANTHANARAYANAN G, HSIEH K, et al. RECL: Responsive resource-efficient continuous learning for video analytics[C]//20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23). Boston: USENIX Association, 2023: 917-932.

[4] XU S K, YAO J C, LUO R, et al. Towards efficient task-driven model reprogramming with foundation models[EB/OL]. (2023-04-05)[2024-06-04]. http://arxiv.org/abs/2304.02263.

[5] HU E J, SHEN Y, WALLIS P, et al. LoRA: Low-rank adaptation of large language models[C]//International Conference on Learning Representations. Virtual Event: OpenReview.net, 2022: 1-13.

[6] LI Y X, YU Y F, ZHANG Q R, et al. LoSparse: Structured compression of large language models based on low-rank and sparse approximation[EB/OL]. (2023-06-26)[2024-06-04]. https://arxiv.org/abs/2306.11222v2.

[7] SANH V, DEBUT L, CHAUMOND J, et al. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter[EB/OL]. (2020-03-01)[2024-06-04]. https://arxiv.org/abs/1910.01108v4.

[8] 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: Association for Computational Linguistics, 2020: 4163-4174.

[9] LIU Z, HU H, LIN Y T, et al. Swin transformer V2: Scaling up capacity and resolution[C]//2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2022: 11999-12009.

[10] TAN M X, LE Q V. EfficientNet: Rethinking model scaling for convolutional neural networks[EB/OL]. (2020-09-11)[2024-06-04]. https://arxiv.org/abs/1905.11946v5.

[11] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[12] HO J, JAIN A, ABBEEL P. Denoising diffusion probabilistic models[C]//NIPS´20: Proceedings of the 34th International Conference on Neural Information Processing Systems. New York: Curran Associates Inc, 2020: 6840-6851.

[13] KAPLAN J, MCCANDLISH S, HENIGHAN T, et al. Scaling laws for neural language models[EB/OL]. (2020-01-23)[2024-06-04]. http://arxiv.org/abs/2001.08361.

[14] ZHANG B, LIU Z, CHERRY C, et al. When scaling meets LLM finetuning: The effect of data, model and finetuning method[C]//The Twelfth International Conference on Learning Representations. Virtual Event: OpenReview.net, 2024: 1-14.

[15] DETTMERS T, LEWIS M, BELKADA Y, et al. GPT3.int8(): 8-bit matrix multiplication for transformers at scale[C]//Advances in Neural Information Processing Systems. New Orleans: ACM, 2022: 30318-30332.

[16] HAN R, ZHANG Q L, LIU C H, et al. LegoDNN: Block-grained scaling of deep neural networks for mobile vision[C]//Pro-

ceedings of the 27th Annual International Conference on Mobile Computing and Networking. New York: ACM, 2021: 406-419.
[17] HOU L, HUANG Z Q, SHANG L F, et al. DynaBERT: Dynamic BERT with adaptive width and depth[EB/OL]. (2020-10-09)[2024-06-04]. https://arxiv.org/abs/2004.04037v2.
[18] GAO X T, ZHAO Y R, DUDZIAK Ł, et al. Dynamic channel pruning: Feature boosting and suppression[EB/OL]. (2018-10-12)[2024-06-04]. http://arxiv.org/abs/1810.05331.
[19] WEN H, LI Y C, ZHANG Z S, et al. AdaptiveNet: Post-deployment neural architecture adaptation for diverse edge environments[C]//Proceedings of the 29th Annual International Conference on Mobile Computing and Networking. New York: ACM, 2023: 1-17.
[20] 陈思光, 陈佳民, 赵传信. 基于深度强化学习的云边协同计算迁移研究[J]. 电子学报, 2021, 49(1): 157-166.
CHEN S G, CHEN J M, ZHAO C X. Deep reinforcement learning based cloud-edge collaborative computation offloading mechanism[J]. Acta Electronica Sinica, 2021, 49(1): 157-166. (in Chinese)
[21] HERNANDEZ D, KAPLAN J, HENIGHAN T, et al. Scaling laws for transfer[EB/OL]. (2021-02-02)[2024-06-04]. https://arxiv.org/abs/2102.01293v1.
[22] XU G Y, HAO J W, SHEN L, et al. LGViT: Dynamic early exiting for accelerating vision transformer[C]//Proceedings of the 31st ACM International Conference on Multimedia. New York: ACM, 2023: 9103-9114.
[23] PENG Y H, BAO Y X, CHEN Y R, et al. Optimus: An efficient dynamic resource scheduler for deep learning clusters[C]//Proceedings of the Thirteenth EuroSys Conference. New York: ACM, 2018: 1-14.
[24] DENG W J, ZHENG L. Are labels always necessary for classifier accuracy evaluation? [C]//2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2021: 15064-15073.
[25] LIANG J, HU D P, FENG J S. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation[EB/OL]. (2021-06-01)[2024-06-04]. http://arxiv.org/abs/2002.08546v6.
[26] WOLF T, DEBUT L, SANH V, et al. HuggingFace's transformers: State-of-the-art natural language processing[EB/OL]. (2020-07-14)[2024-06-04]. http://arxiv.org/abs/1910.03771v5.
[27] DOSOVITSKIY A, BEYER L, KOLESNIKOV A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[EB/OL]. (2020-10-22)[2024-06-04].http://arxiv.org/abs/2010.11929.
[28] ZHANG S S, ROLLER S, GOYAL N, et al. OPT: Open pre-trained transformer language models[EB/OL]. (2022-06-21)[2024-06-04]. https://arxiv.org/abs/2205.01068v4.
[29] TOUVRON H, LAVRIL T, IZACARD G, et al. LLaMA: Open and efficient foundation language models[EB/OL]. (2023-02-27)[2024-06-04]. https://arxiv.org/abs/2302.13971v1.
[30] AN Y Q, ZHAO X, YU T, et al. Fluctuation-based adaptive structured pruning for large language models[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2024, 38(10): 10865-10873.
[31] ASHKBOOS S, CROCI M L, M G do NASCIMENTO, et al. SliceGPT: Compress large language models by deleting rows and columns[C]//The Twelfth International Conference on Learning Representations. Virtual Event: OpenReview.net, 2024: 1-12.
[32] ZHANG Q L, HAN R, LIU C H, et al. EdgeVisionBench: A benchmark of evolving input domains for vision applications at edge[C]//2023 IEEE 39th International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2023: 3643-3646.
[33] LIU Z, WANG J, DAO T, et al. Deja vu: Contextual sparsity for efficient llms at inference time[C]//International Conference on Machine Learning. Hawaii: PMLR, 2023: 22137-22176.
[34] ALIZADEH K, MIRZADEH I, BELENKO D, et al. LLM in a flash: Efficient large language model inference with limited memory[EB/OL]. (2023-12-12) [2024-06-04]. https://arxiv.org/abs/2312.11514v1.

作者简介

张青龙 男,2000年8月出生于云南省昭通市. 现为北京理工大学计算机学院博士研究生. 主要研究方向为边缘智能.
E-mail: 3120211050@bit.edu.cn

韩 锐 男,1985年5月出生于湖北省武汉市. 2014年毕业于帝国理工学院. 现为北京理工大学计算机学院副教授、博士生导师. 主要研究方向为云计算和边缘计算等. 在国内外发表学术论文80余篇. 中国电子学会会员编号:E190026390M.
E-mail: hanrui@bit.edu.cn

刘 驰 男,1984年5月出生于北京市. 2010年毕业于帝国理工学院. 现为北京理工大学计算机学院副院长、教授、博士生导师. 主要研究方向为人工智能、物联网、大数据和边缘计算等. 获2023年中国电子学会自然科学一等奖. 发表CCF-A类论文74篇. 中国电子学会会员编号:E1900130637.
E-mail: chiliu@bit.edu.cn