

文章编号: 1003-0077(2025)02-0001-26

大语言模型微调方法研究综述

吴春志^{1,2}, 赵玉龙¹, 刘鑫⁴, 司念文^{1,3}, 张鲁飞¹, 范昊¹

(1. 数学工程与先进计算国家重点实验室, 江苏 无锡 214125;

2. 航天工程大学 士官学校, 北京 102200;

3. 信息工程大学 信息工程学院, 河南 郑州 450001;

4. 国家并行计算机工程技术研究中心, 北京 100083)

摘要: 近年来, 大语言模型成为人工智能领域非常受关注的技术, 引发了自然语言处理领域新的研究范式。在大语言模型训练实践中, 参数微调是其中非常重要的一个环节, 它允许用户在资源受限条件下, 通过调整少部分参数来提升模型理解用户指令、解决下游任务的能力。该文全面回顾了 2019—2024 年间 50 余种主要的大语言模型微调方法, 从全新的角度进行了系统性的整理和概括, 分为全参数微调、部分参数微调、新增参数微调和无参数微调方法, 对每种方法的原理、微调位置及方法特点作了总结归纳和比较; 接着, 从计算的视角出发, 着重分析比较了各类方法的参数量、内存消耗和计算量; 最后, 基于该文的微调方法调研及相关的参数微调实践, 对大语言模型微调策略给出建议, 以促进该领域的发展。

关键词: 人工智能; 大语言模型; 微调; Adapter; LoRA

中图分类号: TP391

文献标识码: A

Fine Tuning Methods for Large Language Models: A Survey

WU Chunzhi^{1,2}, ZHAO Yulong¹, LIU Xin⁴, SI Nianwen^{1,3}, ZHANG Lufei¹, FAN Hao¹

(1. State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi, Jiangsu 214125, China;

2. School of Non-Commissioned Officer, Space Engineering University, Beijing 102200, China;

3. School of Information Systems Engineering, Information Engineering University, Zhengzhou, Henan 450001, China;

4. National Research Centre of Parallel Computer Engineering and Technology, Beijing 100083, China)

Abstract: In recent years, the advent of large language models (LLMs) has revolutionized the field of artificial intelligence, particularly in natural language processing (NLP). A pivotal aspect of LLM training is parameter fine-tuning, a technique that enables the enhancement of model capabilities under limited resources. This paper offers an exhaustive review of the primary fine-tuning approaches for LLMs from 2019 to 2024, examining over 50 methods. It categorizes and elucidates these methods into four distinct types: full-parameter tuning, partial-parameter tuning, additional-parameter tuning, and parameter-free tuning. Each approach is discussed in terms of its underlying principles, the specific parameters adjusted, and its unique characteristics. Further, the paper delves into a comparison of the parameter volume, memory usage, and computational demands of these methods. Concluding with insights drawn from the fine-tuning research and practical applications, the paper presents recommendations for optimizing LLM fine-tuning.

Keywords: artificial intelligence; Large Language Model(LLM); fine tuning; Adapter; LoRA

0 引言

随着人工智能 2.0 时代的快速推进, 大模型及

其衍生应用成为其中炙手可热的技术产品。大模型一般是指大语言模型 (Large Language Model, LLM), 是自然语言处理领域的“语言模型”概念的延伸, 其惊人的语言能力及广泛的下游应用已经远

收稿日期: 2024-06-04 定稿日期: 2024-09-09

基金项目: 国家重点研发计划(2018ZX01028102)

超出传统语言模型的范畴^[1]。大语言模型是新一代人工智能浪潮下 AI 技术全栈的综合集成,被认为是互联网行业的重工业产品和大科学装置,在“大算力+大数据+强算法”的支撑下,依靠数千块 GPU 显卡、上百 TB 数据以及百亿甚至千亿以上参数量构成的 Transformer 模型完成训练,从无监督预训练,到中间过程有监督指令精调,再加上强化学习对齐优化,综合集成了人工智能领域“无监督学习+有监督学习+强化学习”等核心技术,是对人工智能综合研发能力的巨大考验^[2]。大语言模型表现出优异的通用能力、强大的推理能力和智能涌现能力,集中体现了当前人工智能领域的技术优势和产品应用前景。

现有的类似于 ChatGPT 和文心一言等大模型,个人和小型科研团队难以获取和训练,可直接使用的开源基础预训练语言模型虽然已经拥有大量知识,具备一定的语言理解和生成能力,但距离特定领域应用还有一段距离,需要对其参数做进一步调整,以提升理解用户语言和遵循用户指令的能力。总的来讲,大语言模型微调的目标包括两方面^[3]:

(1) 在基础预训练模型(Pretrain LLM, PLM, 亦称预训练基础模型, Pretrained Foundation Models, PFM)或大模型之上,通过参数微调进一步提升大语言模型遵循用户指令、解决下游任务的能力。

(2) 微调过程应尽量不损害基础模型的知识 and 能力,避免出现“灾难式遗忘”等问题,同时需要尽可能少的资源消耗。

为此,研究人员从内存消耗、训练时长、推理速度等各方面出发,研究如何提升大语言模型的微调效率,降低微调过程的资源代价,进而推出了一系列参数高效微调(Parameter-efficient Fine-tuning, PEFT)方法。这些参数高效微调方法的出现及广泛应用,极大地降低了大语言模型的训练和应用门槛,使得普通的科研人员能在有限的硬件条件下参与到大语言模型的研究中,不断优化和拓展各种高效的微调方法,同时进一步丰富大语言模型面向领域(如医疗、金融、军事等)的应用^①。

目前已有综述从 PEFT 方法介绍、实验分析以及不同的大模型微调侧重点等方面做了相关研究。Diny 等人^[4]提出“delta-tuning”概念,将变化参数的部分称为增加、规范化和重参数化三种 delta 类型,从理论和大量实验的角度分析了包括全参数微调在内的五种微调方法。Xu 等人^[5]主要总结分析了参数高效微调方法的发展脉络,对主要方法原理进行

了介绍,将 PEFT 方法进一步细分为增加、部分、重参数化、混合、统一微调等方法,并将实验测试微调方法数量增加到 11 种。LiaLin 等人^[6]主要从方法原理和实现流程上总结了 2019 年 2 月至 2023 年 2 月推出的 PEFT 方法,探讨了参数变化量和应用到大模型的情况。Dong 等人^[7]重点分析了情景学习微调方法(In-context learning, ICL)的发展脉络和现状挑战。张等人^[8]分析总结了指令调优(Instruction Tuning, IT)的一般方法,IT 数据集的构建,IT 模型的训练,以及在不同模态、领域和应用中的应用。

与现有综述论文相比,本文对大模型微调进行了全视角的回顾,从微调参数的角度区分微调方法为全参数、部分参数、新增参数以及无参数方法,涵盖内容更全面,概括梳理了 59 种微调方法,理论分析了典型方法的内存消耗和计算性能。

本文组织结构如下:第 1 部分介绍大模型微调的相关概念,并分析了微调选用的大模型和数据集;第 2 部分总结分析了四个类型的微调方法;第 3 部分从微调参数量、内存消耗、计算量等角度分析了典型的参数微调方法;第 4 部分进行总结,基于文献调研和微调实践,提出相关的微调策略建议,并对未来进行展望,期望进一步促进大语言模型微调方法的发展。

1 大模型微调相关概念

1.1 Transformer

2017 年,Google 公司提出 Transformer 模型^[9],相比卷积神经网络^[10-11]和循环神经网络^[12-13],Transformer 能更好地捕捉全局信息,并进行并行计算。Transformer 的核心要素是自注意力机制(Self-attention),其能够广泛地进行词与词之间的长距离依赖关系捕捉,并且自注意力中的 Q 、 K 和 V 矩阵可以进行并行计算,这极大地缓解了循环神经网络结构中时序迭代对时间消耗的问题。

Transformer 简洁且可堆叠的网络结构,使其能够在更大的数据集上进行训练,这也促进了后续的预训练技术的发展,Transformer 包含编码器和解码器,现在主流的大模型如 GPT 系列、LLaMA 系列等则采用了仅解码器的结构。如图 1 所示,其中包含注

① 如未特别说明,本文后续所称“大模型”或“LLM”一般是指“大语言模型”。

注意力模块和前馈神经网络模块,经过残差连接(Residual Connection)和层归一化(Layer Normalization)^[14]相连,其中注意力模块由自注意力层、线性层(Linear)、随机丢弃层(Dropout)组成,前馈网络一般包含两个线性层、GeLU 激活函数层、Dropout 层。

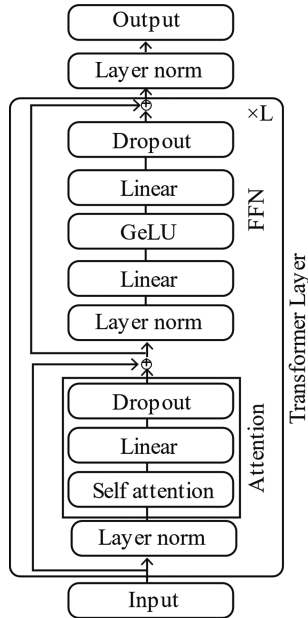


图1 Transformer 结构图

1.2 大模型及其微调

2018 年以来,随着 BERT^[15] 和 GPT^[16] 模型的提出,预训练模型逐渐成为自然语言处理领域的基础性模型。模型微调从以前对有监督学习模型的微调,转变为“自监督预训练+微调”的范式。利用大规模无标注数据,通过自监督学习获得一个语言能力强大的基础模型,接着利用下游任务的带标注数据进行有监督学习来微调模型,以实现对于下游任务适配,如图 2 所示。而随着基于解码器(Decoder-only)架构的 GPT 系列模型^[17] 的快速发展,面向这类生成式模型的提示学习(Prompt Learning)技术逐渐取代面向任务微调,成为主流的微调方式,并逐渐演化出“提示工程”(Prompt Engineering)的概念。这种方式下,选择合适的提示模板(Prompt Template)搭配用户输入,使模型在文本提示的帮助下,按照其在预训练阶段的预测下一个词任务一样去续写用户提示,无须微调参数也能给出下游任务的预测^[18-19]。GPT-3 诞生之后,从提示学习中进一步演化出情景学习,通过在输入中使用特定任务的一些具体示例,同时搭配待测试样本,模型也能够

根据示例续写出待测试样本的答案^[16]。

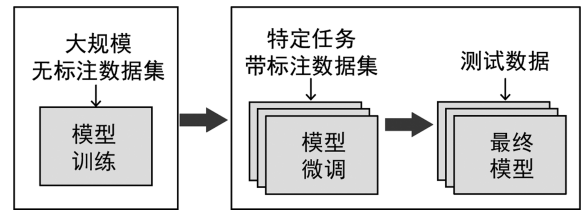


图2 大语言模型的“预训练+微调”范式

随着 Open AI 的 GPT-4^[2] 和谷歌 Gemini 1.0^[20] 的相继推出,大语言模型在 MMLU(大规模多任务语言理解)^[21] 测评上超过人类专家,在 32 个多模态基准中取得 30 个 SOTA(State of The Art),尽管大模型本身效果已经很好,但进行大模型的微调方法研究仍十分必要,主要表现为以下三点:

(1) 相关性。微调是指将通用模型进行微调,并将其转化为专用模型,在通用的预训练模型和特定应用的独特需求之间架起了桥梁^[22]。虽然大模型是根据大量数据进行训练的,但可能不熟悉与特定业务或行业相关的特定术语和细微差别。微调可确保模型理解并生成与业务高度相关的内容,如军事^[23]、金融^[24] 和医疗^[25] 等领域。

(2) 数据隐私和安全。大模型一般根据公开可用的数据生成输出。微调允许对大模型输出的数据进行控制,确保生成的内容不会无意中泄露敏感信息^[26]。

(3) 计算资源。GPT-3 175B 的单个训练需要 460 万美元^[27],ChatGPT 是耗费巨大资源进行训练得到的^[2,20],模型本身规模大而且未公开,使用收费且受限。针对一般的下游任务,可能不需要使用这样的 LLM,仅需微调一定规模的 LLM 即可满足要求^[28]。

在大模型发展的早期,只使用解码器的模型不如使用编码器和编码器-解码器的模型流行。然而,2021 年之后,随着 GPT-3^[29] 的推出,解码器模型受到更多的青睐。同时,BERT、RoBERTa^[30] 等编码器模型逐渐开始淡出^[31]。如何选择基础的大模型,成为模型微调研究的首要步骤。

在实际微调下游任务中,最好选择在相似领域数据上预训练的模型^[31]。在大模型提出的早期,编码器模型 BERT 使用掩码方式对模型进行双向训练,在问答、推理等任务中有较好的表现,解码器的模型更适合生成任务,编码器-解码器模型融合了双方的优点,通常用于需要理解输入并生成相应输出的任务,如机器翻译、摘要生成^[32]。随着大模型的发展,基于解码器模型的 GPT 系列模型(如 ChatGPT)可以把下游任务都建模成生成式任务,用问答的方式解决,解

码器模型成为主流^[31]。然而,受限于计算资源和实验耗时的限制,研究人员在开发新的模型微调算法时,首先会在规模较小的 BERT/RoBERTa-base 等模型上进行对比实验,进一步扩充到 large 模型^[33],这样可以相对快速地获得微调方法的对比结果,但是否能在更大规模模型上进行推广使用,需要进一步研究。LoRA(Low-Rank Adaptation)、QLoRA 等方法在 GPT-3、LLaMa 等大模型中验证后效果较好,在实际任务中得到了较好的推广应用^[34-36]。然而,受限于下游任务配置的计算资源,一些在较小规模的大模型上进行更精准高效的微调研究仍然很有必要。如何选择大模型,除了考虑大模型本身具有的能力外,还要考虑微调的计算资源和下游任务部署的计算资源,以及响应速度要求进行综合考量。

模型微调的目的是让大模型解决自己的下游任务^[37],下游任务包括了词法分析、句子分析、语义分析、信息抽取、翻译、摘要、问答、阅读理解等^[38]。因此选择大模型微调的测试数据集是进行微调方法研究的第二个步骤。

根据大模型在标准数据集上的表现评分可评价大模型的优劣,2018 年初推出的 GLUE (General Language Understanding Evaluation)^[39]是一种评估基准,旨在衡量语言理解模型在一系列自然语言处理任务中的性能。随着大模型研究激增,LLM 在基准上超过了普通人的表现,GLUE 基准变得过时,SuperGLUE^[40]提供了一组全新的更复杂的任务(表 1)。此外,还有对维基百科文章提出的问题组成的阅读理解数据集 SQuAD^[41],由 BBC 文章和附带的单句摘要组成,包含 44 种语言的摘要生成数据集 Xsum^[42],机器翻译数据集 WMT 2016^[43],爬虫数据 Common Crawl 做清洗后得到的语料库 C4^[44]等数据集。微调方法研究通常也以这些标准化的任务合集为基准进行测试,由于这些测试是在一个高度受控的环境中进行的,若使用现实生活中的数据集可能会大大影响模型的性能。研究人员使用 BERT/RoBERTa 等模型在 GLUE/SuperGLUE 数据集上进行研究的另一个重要原因是已有文献中可以找到许多成熟验证过的基线,不需要重复试验^[45],在附表 1 中统计了使用这一基准的测试结果。一些新的微调方法在 GLUE 数据集上表现出来的差距很小,真正在应用微调方法时,应该在尽可能接近生产数据的数据集上进行实验。

表 1 GLUE 和 SuperGLUE 数据集

数据集	子数据集	任务描述	数据条目
GLUE ^[39]	CoLA	确定给定的句子在语法上是否可以接受	8.5k
	SST-2	预测句子的情绪是积极的还是消极的	67k
	MRPC	判断两个句子是否互相转述	3.7k
	QQP	确定两个问题在语义上是否相等	364k
	STS-B	预测句子对之间的相似性得分	7k
	MNLI	对一对句子进行自然语言推理	393k
	QNLI	与 MNLI 类似,但侧重于回答问题	105k
	RTE	确定一个句子是否在逻辑上包含另一个句子	2.5k
	WNLI	解决段落中的代词引用	9.5k
Super-GLUE ^[40]	BoolQ	一个问答任务。它包括一篇维基百科文章中的短文和一个关于该模型的“是/否”问题	68.8k
	CB	包含至少一个嵌入条款的短文本语料库,任务是确定对条款真实性的承诺程度	5.4k
	COPA	因果推理任务。系统会显示一个句子,并且必须从两个选项中辨别因果关系	755k
	MultiRC	多句阅读理解。涉及上下文段落、相关问题和多个潜在答案的 QA 任务	8.4k
	ReCoRD	常识推理数据集的阅读理解,多项选择题 QA 任务	413k
	RTE	来自年度文本蕴涵竞赛,判断包含和非包含	110.4k
	WiC	句子对和多义词的二元分类任务。任务是确定这个词在两个句子中的意义是否相同	5.5k
	WSC	共指解析任务,需要从句子中的名词短语列表中识别代词的正确指代	780

在 ChatGPT 等大语言模型的研发流程中,一般按照“自监督预训练+有监督指令精调+强化学习对齐优化”这三个主要阶段进行。其中,指令精调(Instruct Tuning)是大语言模型参数微调的典型应用,其输入数据的格式是(Instruction, Input, Output)。

指令微调的目的是提升模型理解用户指令的能力,并将这种能力泛化到其他类似任务。在指令微调过程中,可以采用各种参数微调方法,比较常见的微调方法如微调适配器(Adapter)、前缀微调(Prefix Tuning)和 LoRA 低秩微调等,这些是为了降低内存消耗、加速微调过程而提出的参数高效微调方法,当然最直接的方法也可以采用全量参数微调。基于这些方法,研究人员围绕微调过程的内存消耗、训练时长、推理速度等方面不断地进行改进和优化,提出了一系列大语言模型微调方法,这些方法各有优势。本文后续将针对这些微调方法进行分类介绍,然后分析它们的计算性能,并对各种微调策略给出相关建议。

2 微调方法分类

目前,常见的大语言模型包括 OpenAI 公司的 GPT 系列、谷歌公司的 Bard 模型、Meta 公司的 LLaMa 系列等。国内机构目前也相继发布了一些

大语言模型,如百度公司的文心系列^[46]、科大讯飞公司的星火系列、北京智源研究院的 GLM 系列、月之暗面的 Kimi、阿里的通义千问,以及华为公司的盘古系列等模型。为了高效地微调大语言模型,研究人员提出了各种类型的参数微调方法,以提升语言模型遵循用户指令的能力,从而更好地利用其内部知识来回答用户的提问。本节对当前主流的大语言模型微调方法进行回顾,根据微调参数的规模以及是否改变原模型参数,将它们分为 4 种类别:全参数微调、部分参数微调、新增参数微调和无参数微调,如图 3 所示。其中,部分参数微调方法指改变了原模型的部分参数,根据是否有人工参与选择可以分为启发式和自学习两类^[4]。新增参数微调方法指在原模型的基础上增加了附加的参数,在这中有很多方法是在 Adapter 基础上的优化研究,统一放在了 Adapter 的子类下;LoRA 的第二级分类是利用 LoRA 思想进一步发散的研究,对 LoRA 本身做优化的方法被分在了第三级;Prefix、Prompt、P-tuning 等方法作用在 Embedding 层,属于软提示类方法。

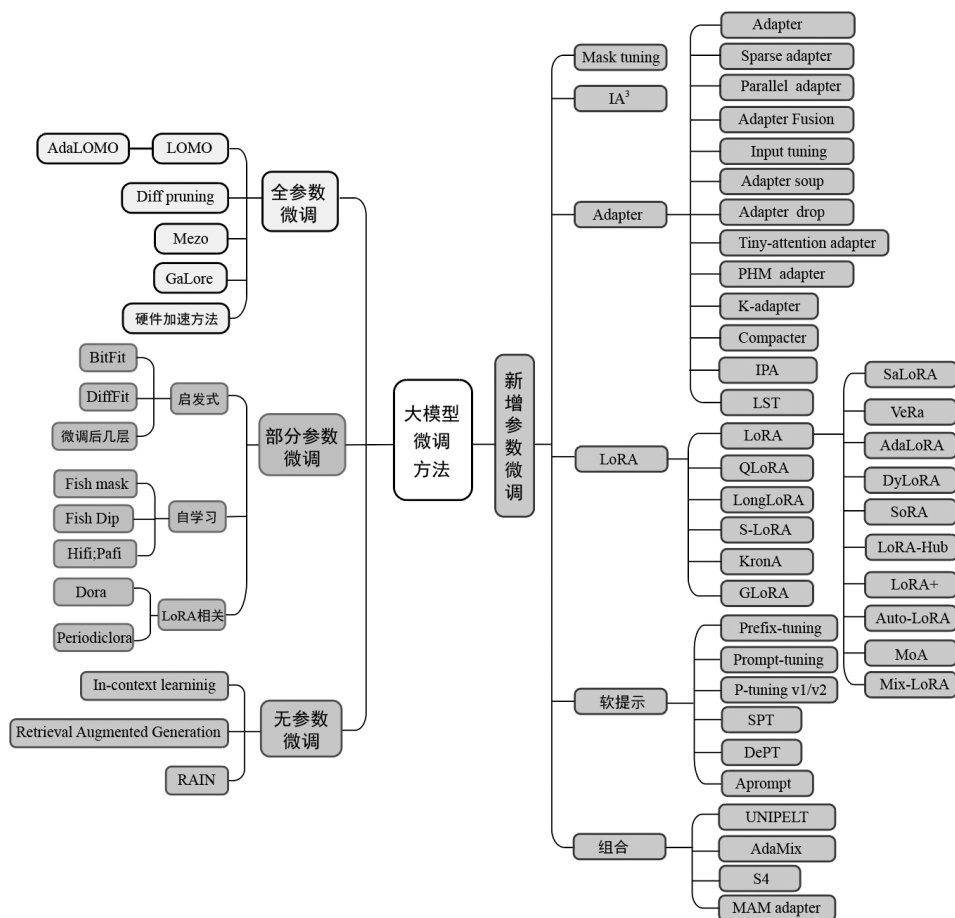


图 3 大模型微调方法分类

adapter^[70] 在新增的 Adapter 中使用稀疏策略, SoRA^[71] 在 LoRA 层中应用稀疏策略等。一些方法侧重调节权重矩阵的规模, IA³^[72] 在注意力计算和 Feed-forward 非线性计算中使用缩放向量 l_k, l_v, l_{ff} , DiffFit^[58] 在整个 Multi-Head 和 Feedforward 中设

置缩放向量 r , LoRA 和 Parallel adapter 在增加的模块中设置缩放向量等。

下面分别介绍全参数微调、部分参数微调、新增参数微调和无参数微调四类微调方法, 表 2 到表 5 总结了各方法的贡献、不足, 以及微调的参数量占比。

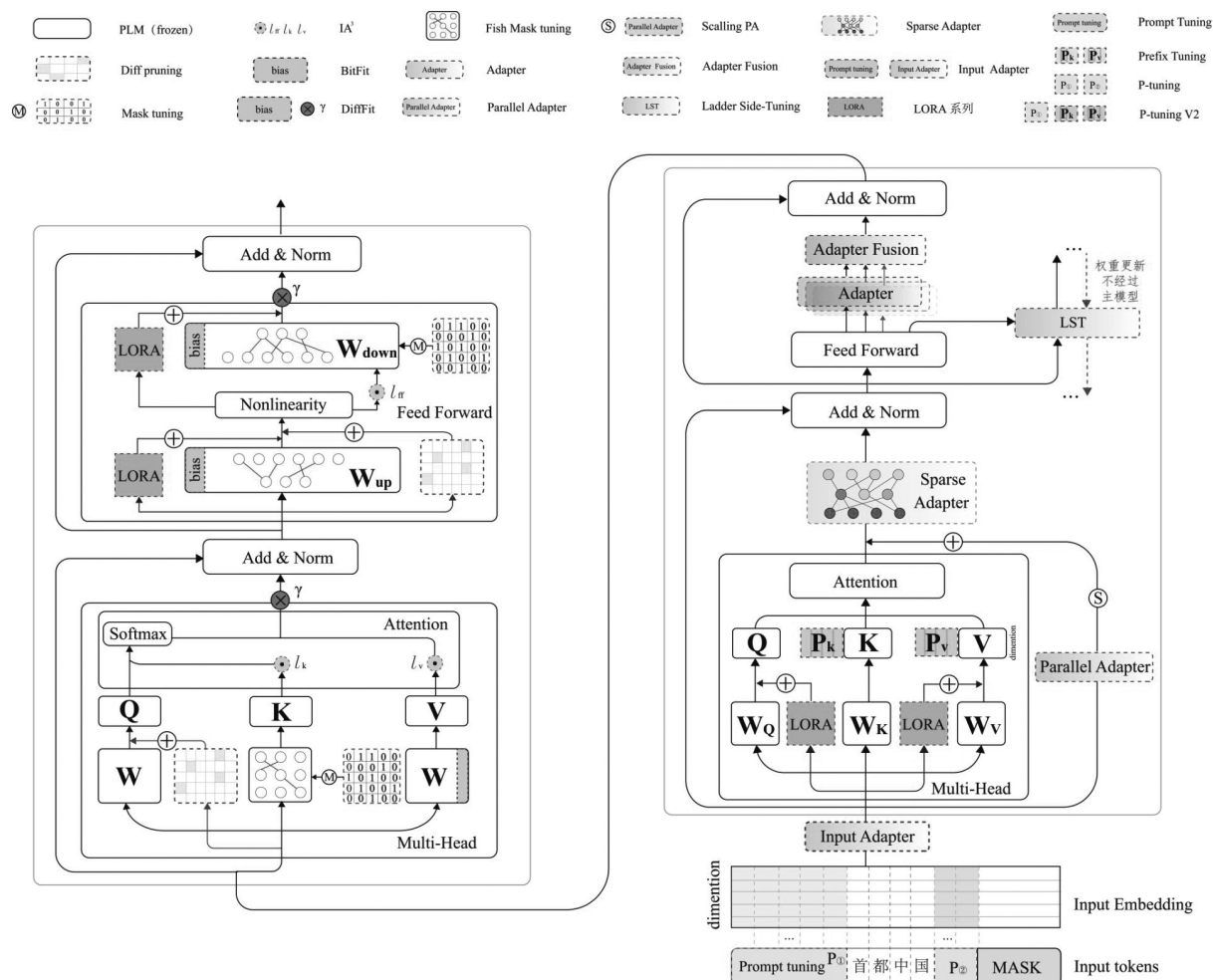


图 5 大模型参数微调的位置

表 2 全参数微调方法对比

序号	方法名称	主要贡献	不足	微调 参数量/%
1	LOMO ^[56]	提出一种新的优化器,通过融合梯度计算和更新,利用有限的计算资源实现了 LLMs 的全参数微调	对超参数敏感,在其他数据集中收敛差,无法与大型语言模型的主流优化器 AdamW 的性能相匹配	100
2	AdaLOMO ^[73]	提出了具有自适应学习率的低内存优化算法,为每个参数提供了一个自适应的学习率,显著降低内存需求	对于有大量激活值占用内存的模型,内存减少是有限的;会引入额外的开销	100
3	Diff pruning ^[67]	仅需存储 0.5% 的全部参数量即可实现与参数微调相当的效果	比全参数微调占用更多的 GPU 内存	100
4	MeZO ^[74]	使用零阶优化器,可以在只进行前向传递的情况下,提高对语言模型进行微调的效率,避免了反向传播所需的过多内存	MeZO 为了获得强大的性能使用了许多中间步骤,一定程度上增加了计算时间	100

续表				
序号	方法名称	主要贡献	不足	微调 参数量/%
5	Galore ^[75]	在模型训练阶段使用梯度低秩投影方法减少内存占用,在优化器状态下减少了高达 65.5 % 的内存使用,同时保持了在 LLaMA 1B 和 7B 架构上的效率和性能,优于 LoRA 方法	比 LoRA 微调耗时要长	100

表 3 部分参数微调方法对比

序号	方法名称	主要贡献	不足	微调 参数量/%
1	BitFit ^[55]	只修改模型的偏置项,占有模型参数的 0.04 %,效果接近全参数微调	在 T0-3B 或 GPT-3 等更大的模型上,性能明显低于全微调和其他 PEFT 方法	0.04
2	DiffFit ^[58]	在 BitFit 的基础上增加了缩放因子,加速了模型训练降低了内存开销	在 NLP 任务中仅使用指令调优测试了问答任务,对其他任务的泛化性研究不足	0.1
3	Final layer tuning	验证仅微调四分之一的最终层就能达到原始质量的 90 %	改变原有模型最终层参数,影响原模型效果	—
4	Fish mask ^[69]	利用 Fisher 信息的经验近似来估计每个参数的重要性,只微调重要参数,大幅降低存储和通信开销	改变模型本身的参数,不利于多下游任务的模型存储,与全微调耗费相同的训练时间	0.08
5	Fish-Dip ^[75]	根据具有最大损失的可用样本的反馈来动态地选择和调整最相关的参数的一小部分,优先考虑那些与期望结果偏差最大的样本,并减少学习良好的样本的影响	只针对序列标注任务,其他任务没有研究	0.5-1
6	HiWi;PaFi ^[57]	提出的方法在性能和存储效率方面都取得了更好的结果,只存储了 0.03 % 的完全微调参数	(Diff Pruning, FISH Mask 和 的 PaFi)仍然需要计算所有参数的全梯度,并选择性地更新屏蔽参数,这使得它与全微调耗费相同的训练时间	0.5~2
7	Dora ^[76]	将预训练的权重分解为大小和方向两个分量进行微调,在避免任何额外的推理开销的同时,提高了 LoRA 的学习能力和训练稳定性	相比 LoRA 微调时间有所增加	0.7
8	Periodiclora ^[77]	在多个阶段将 LoRA 权重卸载到骨干参数中,然后重新初始化 LoRA 状态,通过多次累积低秩更新矩阵来达到更高的更新秩	改变模型本身的参数,不利于多下游任务的模型存储	0.01~0.5

表 4 新增参数微调方法对比

序号	方法名称	主要贡献	不足	微调 参数量/%
1	Mask tuning ^[68]	可以为每个下游任务学习一个选择性掩码,仅更新特定任务的关键权重,存储内存占用小,不改变原有模型权重	没有说明微调的计算消耗	3
2	Adapter ^[50]	开创性的提出增加适配器微调模型,不改变模型权重	增加的适配器提高了推理时的计算开销	0.5~8
3	Sparse adapter ^[70]	证实在稀疏率高达 80 % 的情况下可以达到与标准适配器相当或更好的性能,进一步减少训练的参数量	计算消耗的资源没有减少,没有在更大的模型上进行实验验证	2
4	Parallel adapte ^[62]	优化了适配器模块的放置位置,提出并行适配器的性能优于顺序放置的适配器	需要的训练时间超过全参数微调	6.7

续表

序号	方法名称	主要贡献	不足	微调 参数量/%
5	Adapter Fusion ^[78]	通过分离知识抽取和知识组合两个阶段,实现了多个适配器模块之间的最大任务迁移,有效改善了单个模型在多任务学习时产生的灾难性遗忘和数据平衡困难问题	分两个阶段进行训练,每个任务的训练次数没有减少	—
6	Input tuning ^[61]	改善了 prompt tuning 在多个不熟悉的 NLG 输入时与全参数微调之间的性能差距	仅对比了全参数微调 和 prompt tuning,没有对比其他方法	0.1~6
7	LST ^[63]	提出了阶梯边调,只在一侧添加适配器,设置反向传播不经过主干模型,减少计算内存开销	微调效果相较于其他微调方法,没有提升,只优于 prompt tuning	0.08-1.74
8	IA ³ ^[72]	只需改变权重的缩放向量即可微调大模型,所需参数量非常少	文本生成任务效果未知	0.02
9	Adapter drop ^[79]	实验表明在训练和推理过程中从下层 Transformer 层移除适配器,可以减少反向传播和推理时的计算开销	Adapter Fushion 的改进工作,没有对比其他方法	—
10	compacter ^[80]	使用参数化的超复数乘法替代 Adapter 的上下投影矩阵,有效减少学习参数量	没有减少推理时的计算量,超参数 n 需要人为选择	0.047
11	PHM adapter ^[81]	用四元数的哈密顿乘积代替适配器中全连接层中的实值矩阵乘法,减少到 Adapter 1/4 的参数量	没有减少推理时的计算量,超参数 n 需要人为选择	0.2
12	MAM Adapter ^[67]	在注意力子层使用较小的瓶颈维度进行前缀调优,并使用缩放的并行适配器,分配更多的参数预算来修改 FFN 表示,比单个 PEFT 方法效果好	需要的训练时间超过全参数微调	6.7
13	AdaMix ^[82]	在每个 Transformer 层引入一个混合的自适应模块,改变的参数量小	训练成本是一般 PEFT 方法的 1~2 倍	0.1~0.2
14	K-adapter ^[83]	每个任务分配一组适配器,适配器之间没有梯度传递,适合多任务学习	每个任务整合新的知识,持续增加知识需要更多的适配器	—
15	Adapter soup ^[84]	针对不同任务训练多个适配器,在测试阶段对域适配器进行加权平均,提高了对新任务的适配效率	需要使用文本聚类和语义相似度等方法确定加权平均哪些适配器	—
16	Tiny-attention adapter ^[85]	探索了微小的单头维度的注意力代替上下投影适配器的有效性,减少了训练参数量	仅在 GLUE 数据集上表现好,在摘要生成等文本生成任务上不如其他方法	0.05
17	IPA ^[86]	在解码时将一个大的语言模型裁剪到期望的目标,而不需要对其进行微调,只更新适配器的参数	一个固有限制是它需要访问 PLM 的输出日志。这种约束限制了 IPA 与某些模型的兼容性,如 GPT-4	—
18	LoRA ^[53]	提出对低秩分解进行优化,减少大量训练参数,表现出与全参数微调相当的性能,不增加推理时间	秩 r 作为超参数需要人为设置,不能自适应改变	0.01~0.5
19	Krona ^[87]	使用高效的 Kronecker 积-向量积运算矩阵乘,减少了参训参数计算量	未在更大规模模型上验证	0.07
20	SaLoRA ^[88]	自适应选择 LoRA 模块的内在秩,性能优于 LoRA	会增加训练时的计算开销	1
21	VeRa ^[89]	对低秩矩阵的重新参数化,与 LoRA 相比,可训练参数的数量减少了 10 倍,但保持了相同的性能	代码尚未公开	0.01
22	GLoRA ^[90]	提出了一种通用的参数高效微调方法,LoRA 为其中一个特例	会增加训练的参数量	0.29~0.86

续表

序号	方法名称	主要贡献	不足	微调 参数量/%
23	AdaLoRA ^[91]	通过调整增量矩阵的秩来减少参数计算量,可以动态为关键增量矩阵赋予较高的秩	使用 SVD 方法会增加训练成本	0.1~0.4
24	DyLoRA ^[92]	使用动态低秩自适应技术改变 LoRA 的秩,不需要增加穷尽搜索成本	还需要设置秩的选择范围	0.1~0.5
25	QLoRA ^[54]	引入多个减少内存使用的创新方法,在单个 48GB GPU 上微调 65B 参数模型	没有证明能够在 33B 和 65B 两个尺度上匹配完整的 16 位微调性能	—
26	LongLoRA ^[93]	以有限的计算成本扩展了预先训练的大型语言模型的上下文大小(4k~100k)	无法发现特别长的关联性	—
27	SoRA ^[71]	在自适应过程中动态地调整内在的秩,通过稀疏的方式减少增加的参数数量	未在更大规模模型上验证,仅与 LoRA 进行对比,没有对比其他自适应 r 方法	0.01~0.5
28	S-LoRA ^[94]	实现了为许多 LoRA 适配器的可扩展服务而设计的系统,提供大规模、定制化的微调服务		—
29	LoRAHub ^[87]	引入了一个策略框架 Lora Hub,用于组成在不同任务上训练的 LoRA 模块,以实现对新任务的适应性性能。	如何选择 LoRA 模块,需要引入额外计算资源	—
30	LoRA+ ^[95]	对 LoRA 适配器矩阵 A 和 B 设置不同的学习率,在相同的计算代价下,提高了性能和微调速度	没有考虑到任务和模型对最优比率更精确的估计	0.01~0.5
31	AutoLoRA ^[96]	开发了一种基于元学习的方法来学习这些选择变量。通过阈值化这些变量的值来确定最佳秩	与 AdaLoRA 等其他排序搜索技术相比,引入了一些额外的计算和内存开销	0.1~0.4
32	MoA ^[97]	为 LLM 提供了一种高效的多任务微调方法,解决了任务之间的干扰和训练不稳定问题。每个 LoRA 模型可以单独迭代,以快速适应新的领域	需要额外训练路由器参数	0.5×8
33	Mix-LoRA ^[98]	将多个 LoRA 与共享的 FFN 层融合,并在微调过程中为每个专家存储更新的参数,使用自注意力 LoRA 适配器和一个辅助负载均衡损耗来提高性能和解决路由器不平衡问题	需要提前准备不同任务的 LoRA 模块	2.9
34	Prefix tuning ^[51]	创造性地提出前缀调优,使得指令调整变成可学习的变量,参数量少	需要增加 MLP 层辅助训练,增加训练参数量	0.1~4
35	Prompt tuning ^[52]	仅需在 Embedding 层头部增加有限的可训练 token,不改变模型结构	在模型规模较小时效果不如其他 PEFT 方法	0.1
36	P-tuning ^[47]	针对 NLU 任务设计,在 Embedding 层头部和中部增加可训练的 token,不改变模型结构	仅和 Prompt tuning 等提示调优方法对比,没有对比其他常用微调方法	0.1
37	P-tuning V2 ^[66]	结合了 Prefix tuning 方法,对与训练每一层都增加可学习的提示 token	增加了微调参数量,没有探讨微调计算开销	0.1~3
38	SPT ^[99]	在 Prompt tuning 的基础上自动确定提示层的最佳设置	仅在较小规模的大模型上做了实验研究	0.1
39	DePT ^[100]	将软提示分解为一个较短的软提示和一对低秩矩阵,然后用两种不同的学习率进行优化,在不改变可训练参数大小的情况下,实现了更好的性能,同时节省了大量的内存和时间成本	引入额外的超参数进行调优,例如低秩矩阵的学习率	0.1
40	Aprompt ^[101]	除了在每个 Transformer 层的输入序列开始处插入输入提示外,Aprompt 还在自注意力块中的各自查询、键和值矩阵中预先添加额外的可学习提示,以学习新的注意力模式	在部署过程中潜在的 FLOPs 和延迟的增加	0.08~0.7

续表

序号	方法名称	主要贡献	不足	微调 参数量/%
41	S4 ^[102]	提出了一种参数有效的微调设计范式,发现了适用于不同实验设置的设计模式	对何种任务应用的组合形式没有进一步解释	—
42	UNIPELT ^[103]	将不同的微调方法作为子模块,通过门控机制学习激活最适合当前数据或任务设置的微调方法	增加推理时间,没有对比计算开销	0.99~1.26
43	MEFTs ^[85]	通过适配器修改预训练模型架构,使其具有可逆性,在微调时减少内存占用	在使用 fp16 或层数更深的模型上会因误差增大而降低效果	0.2~0.64

表 5 无参数微调方法对比

序号	方法名称	主要贡献	不足
1	In-context learning ^[7]	不需要微调学习更新参数,仅使用提示模板让模型根据上下文学习下游任务	受限于模型本身能力,需要人为设计提示,鲁棒性差
2	RAG ^[104]	探索了检索式增强生成的微调方法,检索到的内容可作为模型的硬提示,缓解了大模型获取新的精确知识的效果不好的问题	需要依赖额外的知识库,且要保证知识库的正确性
3	RAIN ^[59]	通过综合自我评价和回卷机制,模型模拟人类的思考方式进行自对齐,无须额外的计算资源来进行微调	推理时间比较长,前向搜索过程和后向回溯过程都比较耗费时间

2.1 全参数微调

全参数微调是一种相对彻底的微调方法,原始权重全部参与参数更新,大模型可以更好地适配特定的任务^[105-106]。现有的一些参数量达到 30B 以上的大模型,也是经过全参数微调得到的,如在 LLaMa2 的基础上,微调出适合中文语言的 LLaMa-Chinese,适合代码生成的 CodeLLaMa^[107],进一步的适合 Python 语言的 CodeLLaMa-Python,使用指令微调的 CodeLLaMa-Instruction 等。

但随着大模型参数规模的增大(从 100M 到 100B 甚至更高),在计算和存储资源有限的情况下,传统的全参数微调大模型的方法对于个人和小型研究团队变得越来越困难^[108],部分参数微调和新增参数微调方法成为解决这一矛盾的有效方式。全参数微调的结果成为这些新生方法对比的上限^[4]。为了解决全参数微调后对各种下游任务都需要单独保存大模型,造成的存储资源紧张问题,Diff pruning^[67]通过学习神经网络权重的稀疏更新来实现参数效率。该方法在权值上引入一个可学习的二进制掩码矩阵 Diff,以 L0 正则化方法尽可能稀疏存储参数,仅存储与原始权重相比变化的部分,大大减少了全参数微调后模型的存储空间,但是并没有改变计算量大的问题,需要全参数微调之后再计算得到 Diff,增加了计算负担,对规模较大的模型不适用^[6]。

相比于存储资源,内存等计算资源则更为珍贵^[31],为了以更小代价实现全参数微调,一些研究人员开始从微调的计算过程着手,利用一些内存优化方法改善全参数微调对计算资源的依赖。LOMO^[56] (Low-Memory Optimization) 方法提出了一种新的优化器,采用梯度值裁剪、分离梯度范数计算和动态损失缩放来稳定训练。激活检查点和 ZeRO 优化方法的集成,节省了内存^[7-8]。具有自适应学习率的低内存优化算法 AdaLOMO^[73] 又对 LOMO 优化,进一步适配主流优化器 AdamW,同时显著降低了内存需求,从而降低了训练大型语言模型的硬件障碍。与传统的 PEFT 技术依赖反向传播计算梯度更新模型参数不同,MeZO^[74] 仅通过前向通道微调 LLM。MeZO 实现了经典 ZO 梯度估计器的原位解决方案,有效地减轻了推理执行过程中的内存消耗。这种创新的方法允许在具有 8011GB 内存的单个 GPU 上对包含 300 亿参数的 LLM 进行有效的微调,同时保持与使用反向传播进行微调相当的性能。在模型训练阶段已经应用的 Flash attention^[109]、Galore、FSDP (Fully Sharded Data Parallel)^[110] 等内存优化计算的方法也可应用到大模型微调中,利用更小的计算资源进行更大规模 LLM 的全参数微调,是一个亟待解决的问题。

2.2 部分参数微调

部分参数微调早在迁移学习方法应用到深度学

习领域中时就成为主流的参数调整方法,作为全参数微调在参数更新上的优化方式^[111],取得了一些很好的效果^[112]。这类方法会改变一些原始参数,但大部分是不变的^[113],因此不会改变模型的结构。Ding 等人^[4]对这种方法进一步分类为启发式和自学习式两种。启发式需要人为规定微调哪些部分, Lee 等人^[114]发现只需微调 BERT 和 RoBERTa 四分之一的最终层,即可达到全参数微调 90% 的效果, BitFit 和 DiffFit 在权重的偏置上着手,改变的参数数量降低到 0.04%。自学习式则通过学习规则,使模型改变一些特定的权重,如 Fish mask tuning 和 HiWi。部分参数微调方法一个普遍的不足是需要计算所有参数的全梯度^[115],仍然需要耗费大量的训练时间,同时由于原始参数发生了改变,因此对于不同的下游任务,模型整体的权重都需要存储,没有解决存储资源占用大的问题^[6],而且效果没有全参数微调效果好^[56]。部分参数微调类的方法随着模型规模的逐渐增大,微调代价也在增大,除 BitFit 经常作为对比方法被引用次数多以外,其余的方法关注度一直不高(图 4)。新增参数微调方法可以较好地解决占用存储资源和微调代价大的缺点。DoRA^[76]和 PeriodicLoRA^[77]改造了 LoRA 方法,将新增参数合并回原权重,改变了权重参数,因此划归到部分参数微调方法内。

2.3 新增参数微调

随着模型规模的增大,传统的全参数微调和部分参数微调方法面临两大问题,一是基础模型太大,对于多种下游任务,多个微调后的大模型会占用存储空间,而且模型权重修改后会出现知识的遗忘^[78]。二是微调模型所需要的计算资源较大,全参数微调代价太高^[116]。减少权重存储量,降低计算资源,是研究人员进一步优化模型的方向,也是新出 PEFT 方法的评价标准。新增参数微调方法可以较好地解决上述两大问题,并且不改变原模型权重,避免模型的灾难式遗忘。其代表方法是增加模块的 Adapter 系列、重参数化的 LoRA 系列、软提示系列以及组合调优系列方法。

2.3.1 Adapter 系列方法

2019 年, Houlsby 等人^[50]创造性地提出在 Transformer 模块内部增加一块可训练的参数,每个适配器都由低秩权重矩阵组成,由一个下投影矩阵 $\mathbf{W}_{\text{down}}$ 、一个非线性激活函数 ReLU、一个上投影矩阵 $\mathbf{W}_{\text{up}}$ 和一个残差连接组成。在微调过程中,只需要更新 Adapter 的 $\mathbf{W}_{\text{down}}$ 和 $\mathbf{W}_{\text{up}}$ 参数,即可使 PLM

适应具体的下游任务,不改变原模型的权重,对不同的下游任务可以存储相应的 Adapter 块,减少了模型的存储空间。在此基础上,研究人员面向不同的侧重方向对方法进行了优化改进。

(1) 一个模型适配多种下游任务。AdapterFusion^[78]、AdapterSoup^[84]、K-adapter^[83]等方法通过在二次训练时增加 Fusion 层、分配适配器合适的权重、分组训练适配器的形式使一个模型适配多种下游任务。

(2) 扩宽 Adapter 模块应用范围。Zhu^[62]和 He^[117]等人提出的并行 Adapter[结构如图 6(b)所示],将原来串行在 Feedforward 层后的 Adapter 层并行到多个模块外,效果优于串行 Adapter。LST^[63]在主干网络外训练一个梯子侧小型且独立的网络,将中间激活作为输入并进行预测。LST 与以前的方法相比,内存需求显著降低,因为它不需要通过主干网络反向传播,而只需要通过侧网络和阶梯连接。但对于规模越来越大的大模型,改善效果还有待检验。An 等人^[61]提出的 Input tuning 在输入层应用 Adapter,即在 Embedding 层和 PLM 的底层之间添加了一个轻量级的可训练模块,可直接微调输入的编码。IPA^[86]通过训练轻量级 Adapter,对大模型进行剪裁以适配下游任务,这对处理较大规模 LLM 有较好的效果。

(3) 优化 Adapter 模块的内部结构。Sparse adapter 通过稀疏的方式减少了模块内的训练参数, Tiny-attention adapter^[85]则将 Attention 计算方式引入 Adapter 层减少训练参数, Compacter^[80]和 PHM Adapter^[81]通过改变矩阵乘法规则,降低计算量。Adapter drop^[79]通过丢弃底层的 Adapter 模块,减少增加的推理时间。

2.3.2 LoRA 系列方法

随着大模型规模的增大,模型本身蕴含的知识和推理能力也不断增强,微调大模型只需改变增加的更少部分的参数即可“唤醒”大模型对特定下游任务的适配^[118],因此对原有 PEFT 方法的进一步降本增效是研究人员努力的方向。微软研究团队在 2021 年推出的 LoRA 开辟了一个新思路,低秩 r 和重参数化,使用两个低秩矩阵 $\mathbf{W}_A$ 和 $\mathbf{W}_B$ 相乘后与原权重矩阵规模一致的方式,大大降低了需要训练的参数(如图 6(a)所示),所有预训练的模型参数都被冻结,只有 $\mathbf{W}_A$ 和 $\mathbf{W}_B$ 矩阵是可训练的。此外,增加一个缩放因子,通常是 $1/r$ 。在推理时加回原矩阵,不增加推理时间,只需存储新增的 LoRA 部分

参数,且参数量比 Adapter 方法低,相比于全参数微调可以降低 1/3 的内存资源,并且在 175B 规模的

LLM 上进行了验证。许多研究论文在 LoRA 的基础上进行了改进研究。

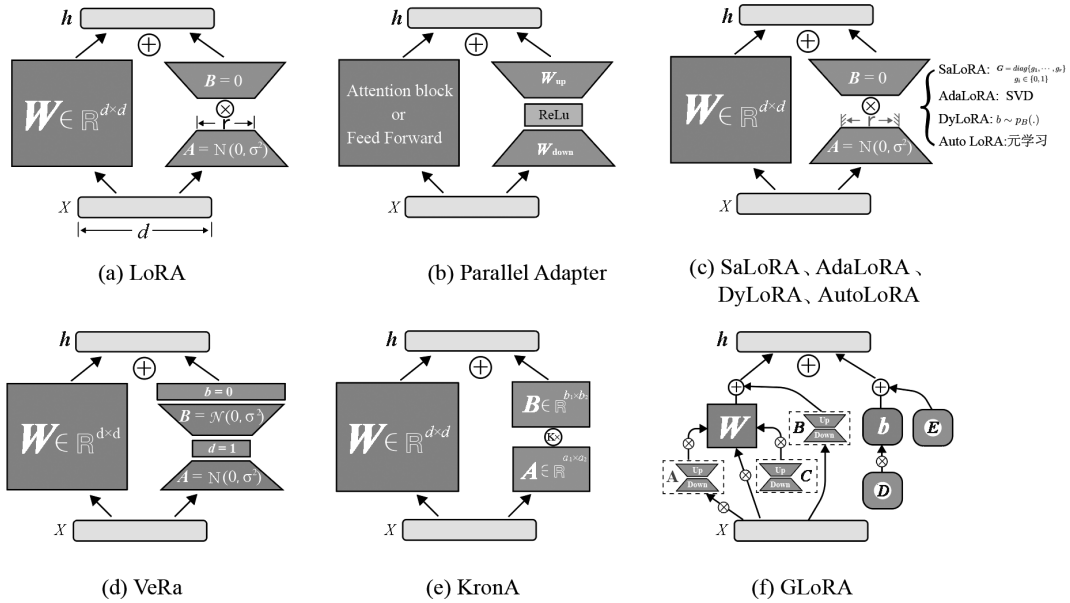


图 6 LoRA 相关微调方法结构示意图

(1) 自适应秩 r 。由于 LoRA 需要人为确定一个不可更改的秩 r , SaLoRA^[88]、AdaLoRA^[91]、DyLoRA^[92]、AutoLoRA^[96] 等方法利用不同的方式可自适应选择秩的大小,增加了模型的灵活性[如图 6(c)所示]。为了在训练过程中以最小的计算开销找到最优的秩, SaLoRA (Structure-Aware LoRA) 引入了一个门矩阵 $\mathbf{G}$ 来定义结构敏感的秩,如式(1)所示。

$$\Delta \mathbf{W} = \mathbf{W}_B \mathbf{G} \mathbf{W}_A = \sum_{j=1}^r g_j \mathbf{W}_{B \times j} \mathbf{W}_{A \times j} \quad (1)$$

其中, $g_j \in \{0, 1\}$ 充当二元“门”,表示第 j 个秩的存在与否。门矩阵 $\mathbf{G} = \text{diag}(g_1, \dots, g_r)$ 是由修剪变量组成的对角矩阵。通过学习变量 g_j ,可以单独控制每个增量矩阵的秩,而不是对所有矩阵应用相同的秩。DyLoRA (Dynamic LoRA) 对 LoRA 模块训练一系列不同大小的秩 $r \in [r_{\min}, r_{\max}]$ 并进行一系列迭代。每次迭代中随机选取秩 r_b ,然后将下投影矩阵截断为 $\mathbf{W}_B^{r_b} = \mathbf{W}_B[:, :r_b, :]$ 上投影矩阵截断为 $\mathbf{W}_A^{r_b} = \mathbf{W}_A[:, :, :r_b]$,只更新截断矩阵,通过允许动态低秩自适应和免搜索低秩自适应, DyLoRA 减少了为特定任务确定最优秩所需的计算成本和训练时间。在 AdaLoRA 中,增量更新矩阵 $\Delta \mathbf{W}$ 使用奇异值分解 (Singular Value Decomposition, SVD) 重新参数化,然后截断最小的奇异值,即 $\Delta \mathbf{W} = \mathbf{P} \mathbf{\Lambda} \mathbf{Q}$ 。 $\mathbf{P}$ 和 $\mathbf{Q}$ 都是正交矩阵, $\mathbf{\Lambda}$ 是包含奇异值的对角矩阵。在训练

过程中, $\mathbf{P}$ 和 $\mathbf{Q}$ 用高斯分布初始化,并使用正则化器来保证正交性,而 $\mathbf{\Lambda}$ 用零初始化,并通过迭代剪枝来调整秩。AutoLoRA 使用元学习思想通过两个迭代步骤学习选择变量。先在训练数据集上优化更新矩阵中的权重,后在验证数据集上更新选择变量。这两个步骤不断迭代,直到收敛。在获得选择变量的最优值后, AutoLoRA 通过阈值化这些值来确定最优矩阵的秩。随后, LoRA 层中更新矩阵的秩被设置为学习到的最优秩,并在训练和验证数据的组合上重新训练。

(2) 减少训练参数。为进一步减少参数量, LoRA-FA^[119] 将 LoRA 中的 $\mathbf{A}$ 矩阵冻结,仅更新 $\mathbf{B}$ 矩阵,减少了一半的可训练参数; VeRa^[89] 则进一步冻结大部分 $\mathbf{A}$ 、 $\mathbf{B}$ 矩阵中的参数并且作为多个 LoRA 模块的共享,每个模块仅需调整两个向量[如图 6(d)所示]; KronA^[81] 使用科罗纳尔积 (Kronecker) 代替传统的矩阵乘,降低了 $\mathbf{AB}$ 矩阵的规模[如图 6(e)所示]; SoRA^[71] 通过稀疏的方式减少 LoRA 的可训练参数。

(3) 扩宽适配范围。由于 LoRA 方式在大模型微调领域的成功应用,研究人员进一步提出 S-LoRA^[94], 为许多 LoRA 适配器的可扩展服务而设计并提供大规模、定制化的微调服务。LoRAHub^[120]、MoA^[97]、Mix-LoRA^[98] 面向多任务微调统筹了不同任务的 LoRA 模块。GLoRA^[90] 从统一的视角扩展了 LoRA 的结构范围,使得 LoRA 成为 GLoRA 的一个特例。GLoRA 架

构可以简洁地表述为一个统一的数学方程。表示所有可调空间的统一公式可以表示为:

$$f(X) = (W_0 + W_0 A + B)X + C W_0 + D b_0 + E + b_0 \quad (2)$$

其中, W_0 和 b_0 不动, 其他变量根据需要设置是否参与训练。

此外, 除了特定的下游任务需要进行模型微调, 微调方法还可以解决大模型自身的一些限制, 例如, LLaMa2-7B^[121] 处理的上下文长度是 4 096, LongLoRA^[93] 借鉴 Swin Transformer^[122] 的思想, 将长度提升至 100k, 解决了大模型不能处理著作、论文等长文本的限制。LoRA⁺^[95] 对 LoRA 适配器矩阵 A 和 B 设置不同的学习率, 相同的计算代价下, 提高了性能和微调速度, 可为进一步降低微调对内存的占用。QLoRA^[54] 使用 4-bit 量化模型、分页优化管理内存等技术, 大大降低了微调模型所需的内存占用, 在一些领域取得了很好的效果。

2.3.3 软提示系列方法

软提示系列方法是相对于离散的硬提示微调方法提出的, 硬提示微调方法属于无参数调整的微调, 在 2.4 节进一步介绍。软提示微调目的是使用可学习的额外上下文来包装原始的输入, 在微调过程中表现出更大的灵活性和适应性, 因为这些提示可以根据具体的任务和训练数据进行优化和调整, 代表方法是 Prompt tuning、Prefix tuning 和 P-tuning。

Prompt tuning 在模型输入 X 上加入 l 个可学习的 prompt token, 然后将它们串联起来生成最终的输入, 在微调过程中, 模型自动学习到需要提示的规则, 而预训练参数保持冻结。文献[123]所做的从 1 到 150 个 token, 模型大小从 10M 到 11B 参数的消融实验表明, 模型越大, Prompt tuning 越有效。在模型规模较小时, Prompt tuning 远落后于其他 PEFT 方法^[4]。

Prefix tuning 将可学习的前缀提示范围扩大到每一个 Transformer block 中计算后的 K 、 V 矩阵, 进一步加深对模型的提示作用。为了保证训练的稳定性, 引入多层感知器 (MuLtilayer Perceptron, MLP) 进行参数化, 辅助更新参数, 因为直接优化会导致不稳定。训练完成后, 舍弃 MLP, 保留 P_k 和 P_v 进行推理 (图 5)。

P-tuning V1 和 V2 是对前面两种方法的进一步融合。P-tuning 使用双向长短时记忆网络和 ReLU 激活的 MLP 辅助参数训练, 训练完毕后舍弃。软提示调优方法作用区域与其他新增参数调优

方法不同, 因此可以进一步结合其他方法使用, 如 Input tuning 等。

Aprompt^[101] 将软提示插入的范围扩展到自注意力模块中的 K 、 Q 、 V 中; SPT^[99] 可以自动确定软提示的最佳设置; DePT^[100] 借用 LoRA 的思想, 将软提示分解为一个较短的软提示和一对低秩矩阵, 然后用两种不同的学习率进行优化。

2.3.4 组合调优系列方法

诸多微调方法的作用区域和作用机制是不相同的, 研究人员发现在不同位置作用的微调方法组合起来应用, 也会产生意想不到的效果。

UNIPELT^[103] 通过门控机制将 Adapter、Prefix tuning 和 LoRA 整合在一起。门控机制控制每个子模块的激活, 对给定任务做出积极贡献的子模块动态分配较高的权重, 除了上述三种方法对应的权重可训练外, 还有门控函数的权重可调节。与单个 PEFT 方法相比, UniPELT 具有更好的性能, 但是会增加推理时间。

AdaMix^[82] 堆叠了多个相同类型的 Adapter, 通过随机路由由训练激活避免了更多的计算成本。AdaMix 在串行 Adapter 和 LoRA 上实现了算法, 但它可以被看作是一种适用于任何 Adapter 类的通用技术^[124]。

He 等人^[67] 对 Adapter 放置和软提示进行了深入的研究, 提出了 MAM Adapter。他们得出以下结论: 缩放的并行 Adapter 优于顺序放置的 Adapter, 将 Adapter 与 FFN 并行放置优于多头注意力并行, MAM Adapter 是 FFN 层的缩放并行 Adapter 和软提示的组合。

S4^[102] 则设计了一种统一的 PEFT 方法应用架构。使用“纺锤形”模式将层分为四组 (G_1 、 G_2 、 G_3 、 G_4): 更多的层分配给中间组, 更少的层分配给顶部组和底部组。所有组都是可训练的, 可训练的参数均匀地分布在各层中。PEFT 方法的不同组合应用于不同的群体。实验结果表明, 当每个组都配备以下 PEFT 方法的组合时, 可以获得更好的性能。A 表示 Adapter, P 表示 Prefix tuning, B 表示 BitFit, L 表示 LoRA。

$$\begin{aligned} G_1: (A, L); \quad G_2: (A, P); \\ G_3: (A, P, B); \quad G_4: (P, B, L) \end{aligned} \quad (3)$$

S4 的工作可以作为大模型微调的可解释性探索。在实际微调中, 应用哪几种微调方法微调自己的下游任务效果更好, 成为了一个超参数。

2.4 无参数微调

无参数微调的本质是通过外部给模型离散的提

示,外部可以是人也可以是其他大模型给的提示,激发唤醒模型本身的知识能力,得到符合预期结果的过程^[125]。随着模型规模的增大,模型已经具有从上下文中的少数实例中学习的能力,还可以执行一系列复杂任务,如解决数学推理问题^[126-127]。情景学习(ICL)将一个查询问题和一段演示上下文串联在一起形成一个提示,ICL 的性能对特定的设置比较敏感,包括提示模板、语境示例的选择、示例的顺序等^[7]。ICL 使人类知识更容易融入到 LLMs 中,类似于人类通过类比学习的过程。与监督训练相比,ICL 是一个无须训练学习的框架。这不仅可以大大降低模型适应新任务的计算成本^[7],而且可以很容易地应用于大规模的现实任务^[128]。

检索增强生成 (Retrieval-Augmented Generation, RAG)^[104]探索了检索式增强生成的微调方法,检索到的内容可作为模型的硬提示,缓解了大模型获取精确知识效果不好的问题,如何使得模型的参数记忆和检索到内容的非参数记忆相互作用,以及如何有效地组合值得进一步研究。可回放的自回归参考方法 (Rewindable Auto-regressive Inference, RAIN)^[59]通过综合自我评价和回卷机制,模型模拟人类的思考方式进行自对齐,无须额外的计算资源来进行微调,这些都是对离散提示的很好探索。

目前,无参数微调还面临着上下文长度受限、鲁棒性不足、可扩展性差和效率低等问题^[129]。随着大模型规模的不断扩大,大模型能力未来还会提升,因此如何高效地利用无参数微调唤醒大模型、使模型自对齐等都是值得探索的方向。

3 微调方法分析

典型的 Transformers 架构包含 Attention 模块和 FFN 模块,其中 Attention 模块由 Self Attention、Linear、Dropout 组成,FFN 包含两个 Linear 计算、GeLU、Dropout。

3.1 微调参数量

Transformer 模型由 n_{layer} 个相同的层组成,每层中的 Attention 模块的参数包括 Q 、 K 、 V 矩阵的权重矩阵和偏置,输出矩阵 O 的权重和偏置,这四个矩阵形状相同;FFN 模块由两个线性层组成,一般地,第一个线性层先将维度从 d_{model} 映射到 $4d_{\text{model}}$,第二个线性层再将维度从 $4d_{\text{model}}$ 映射到 d_{model} 。Attention 模块和 FFN 模块各有一个 layer

normalization,包含了缩放参数和平移参数,参数量为 $4d_{\text{model}}$ 。

$$\begin{aligned} W_{\text{emd}} &= n_{\text{vocab}} d_{\text{model}} \\ W_{\text{atten}} &= W_q + b_q + W_k + b_k + W_v + W_o + b_o \\ &= 4d_{\text{model}} d_{\text{model}} + 4d_{\text{model}} \\ W_{\text{ff}} &= W_{\text{up}} + b_{\text{up}} + W_{\text{down}} + b_{\text{down}} \\ &= 8d_{\text{model}}^2 + 5d_{\text{model}} \\ W_{\text{layernorm}} &= 4d_{\text{model}} \\ P &= W_{\text{emd}} + n_{\text{layer}}(W_{\text{atten}} + W_{\text{ff}} + W_{\text{layernorm}}) \\ &\approx 12n_{\text{layer}} d_{\text{model}}^2 \end{aligned} \quad (4)$$

其中, P 为总的参数量, n_{vocab} 是词表大小(LLaMa-7B 为 51 200), d_{model} 为模型的(隐藏)维度(LLaMa-7B 为 4 096), n_{layer} 是模型的层数(LLaMa-7B 为 32), W_{emd} 是嵌入层的参数量, W_{atten} 包含了计算 Q 、 K 、 V 的参数以及最后线性输出的参数量, W_{ff} 是两个线性计算(一个升维、一个降维)的参数量, $W_{\text{layernorm}}$ 是两个 Layer-Norm 操作的参数量。对于大语言模型来说,其参数量可以用 $12n_{\text{layer}} d_{\text{model}}^2$ 来近似计算。

对于 LLaMa-7B^[130] 模型:

$$\begin{aligned} P &= W_{\text{emd}} + n_{\text{layer}}(12d_{\text{model}}^2 + 13d_{\text{model}}) \\ &= 6.7B \end{aligned} \quad (5)$$

3.2 内存消耗分析

在模型混合精度(mixed-precision, fp16/fp32)微调期间,内存消耗主要由模型参数(parameters, fp16 类型)、优化器(optimizer)状态(fp32 类型)、模型梯度(gradients, fp16 类型)、中间激活值(activations, 基本为 fp16)^[131]四部分组成,其中“中间激活值”是指在前向计算中创建且是反向梯度计算所必需的中间值。这些内存的占用量如式(6)所示。

$$\begin{aligned} M_{\text{para}} &= 2P \\ M_{\text{gra}} &= 2P \\ M_{\text{opt}} &= KP \\ M_{\text{atc}} &= n_{\text{layer}} s b d_{\text{model}} \left(34 + 5 \frac{hs}{d_{\text{model}}} \right) \\ T_{\text{mem}} &= M_{\text{para}} + M_{\text{gra}} + M_{\text{atc}} \end{aligned} \quad (6)$$

其中, P 为模型参数个数, K 是随优化器变化而变化的系数,例如对于混合精度的 Adam 优化器 $K = 12$ ^[132];带有动量类的 SGD 优化器, $K = 8$;不带动量类的 SGD 优化器, $K = 4$ ^[133]。 M_{atc} 中, s 是输入序列的长度(Sequence Length), b 为批次大小(BatchSize), h 为多头注意力的头数(Number of Attention Heads), d_{model} 为模型的(隐藏)维度, n_{layer} 是模型的层数。

T_{mem} 是所需的总内存量。对于 LLaMa-7B 模型, $s=2048, b=1, h=32, n_{\text{layer}}=32, d_{\text{model}}=4096$, 使用 Adam 优化器进行全参数微调, 所需内存至少为 137.8GB。

在 LoRA 类微调方法中, 为提升微调效果, 把 LoRA 模块添加到 Transformer 的所有线性层中 Rank 参考值^[91] r 设为 16, LoRA 的微调参数量^[134] 为:

$$P_{\text{LoRA}} = n_{\text{layer}} 13 d_{\text{model}} r \quad (7)$$

LoRA 中为计算所有权重 $\mathbf{A}$ 的梯度, 需要存储的激活值(Activation)大小为:

$$M_{\text{act-A}} = n_{\text{layer}} 14 b s d_{\text{model}} \quad (8)$$

这部分是涵盖在 M_{act} 中, 不再重新计算。

LoRA 中为计算所有权重 $\mathbf{B}$ 的梯度, 需要存储的激活值大小为:

$$M_{\text{act-B}} = n_{\text{layer}} 12 b s r \quad (9)$$

使用 BitFit 微调方法时, 对所有的偏置(bias)进行微调, 包含 $b_q, b_k, b_v, b_{m1}, b_{\text{LN1}}, b_{m2}, b_{m3}, b_{\text{LN2}}$ 。参数量为:

$$P_{\text{bitfit}} = 11 n_{\text{layer}} d_{\text{model}} \quad (10)$$

使用 Adapter 方法进行微调, 在 Attention 和 MLP 后各加入了一个 Adapter, 其中的两个变换矩阵大小分别为 $d_{\text{model}} \times m$ 和 $m \times d_{\text{model}}$, 增加的参数量为:

$$P_{\text{adapter}} = n_{\text{layer}} (2 m d_{\text{model}} + d_{\text{model}} + m) \quad (11)$$

Adapters 方法中, Activation 内存量增加了两个 adapter 部分的消耗, 同时减少了第一层中 Attention 模块中的消耗, 总量为:

$$M_{\text{act-adapter}} = M_{\text{act}} + 2 b s d_{\text{model}} (n_{\text{layer}} - 1) \quad (12)$$

不同微调方法, 其内存消耗如表 5 所示。

表 6 7 种典型微调方法的内存消耗分析

方法	微调参数量	内存消耗/GB				
		Params	Gradients	Optimizer	Activation	Total
LoMo ^[56]	6.7B	13.4	0.24	0	30.6	44.2
AdaLoMo ^[73]	6.7B	13.4	0.24	0	30.6	44.2
LoRA ^[53]	37.8M	13.48	0.075	0.45	30.61	44.6
LoRA-FA ^[119]	18.9M	13.44	0.038	0.23	30.61	44.3
BitFit ^[55]	1.4M	13.4	0.002 8	0.016 8	30.6	44.02
Adapters ^[50]	4.3M	13.4	0.008 6	0.052	31.7	45.16
Full-fineTuning ^[105]	6.7B	13.4	13.4	80.4	30.6	137.8

从表 6 的对比中可以看到, 相比于全参数微调, 各类参数高效的微调方法均能大幅度降低内存占用, 对内存占用的优化集中在保存梯度和优化器参数所使用的内存, 激活值的内存消耗占比较大。

降低激活值占用内存的主流方法是激活值重计算(Activation Recomputation, Activation Checkpointing, Gradient Checkpointing)^[135], 通过只存储网络激活的一部分而不是所有中间激活值(通常的做法)来减少激活内存的使用, 在反向传播过程中, 未存储的激活必须重新计算, 因此, 检查点会增加 30% 左右的计算量。

近期有论文提出用“可逆模型”^[85] 方法来大幅降低激活内存, 它通过引入一种特殊的参数初始化策略和反向传播算法, 使得在训练过程中, 模型的内部状态(主要是激活值)可以通过计算重建。它可以显著减少微调阶段激活内存需求, 从而使得在大规

模数据集上进行微调成为可能, 同时不会改变原始模型的参数, 因此不会影响到模型的泛化性能。这种方法在降低激活内存上与激活值重计算本质上是一样的, 均通过计算来换取内存空间, 计算量的增加也在 30% 左右。

3.3 计算量分析

Transformer 模型微调的计算量 FLOPs (Floating point Operations Per second) 包含前向计算量和反向传播计算量。

前向计算中, 每个 token 的计算量包含 Attention 模块计算和 MLP 部分的计算如式(3)所示。

$$\begin{aligned}
 C_{\text{attn}} &= C_{\text{QKV}} + C_{\text{QK}} + C_V + C_{\text{out}} \\
 &= 8 d_{\text{model}}^2 + 4 d_{\text{model}} \\
 C_{\text{mlp}} &= 16 d_{\text{model}}^2 \\
 C_{\text{forward}} &= n_{\text{layer}} (C_{\text{attn}} + C_{\text{mlp}}) \quad (13)
 \end{aligned}$$

上面的 C_{forward} 是一个 token 在 Transformer 模型中进行一次前向计算的总计算量, 这里 $4d_{\text{model}}$ 可以忽略, 对比模型的总参数量 $P \approx 12n_{\text{layer}} d_{\text{model}}^2$, 可以发现前向计算中每个 token 的计算量 $C_{\text{token}} \approx 2P$ 。

反向传播计算量为前向计算量的两倍^[136], 所以, 单个 token 总的计算量为:

$$C = C_{\text{forward}} + C_{\text{back}} = 3C_{\text{forward}} \approx 6P \quad (14)$$

前向计算量与参数量线性相关, 对于参数高效的各类微调方法 (包括 LoRA 类、Prompting 类、Adapter 类), 因其引入的新参数量一般较少 (低于 1%), 对前向计算量影响不大, 可以认为各类微调方法的前向计算量与全参数微调相当。

在反向传播计算过程中, 可以将模型梯度计算量分为两部分, 一部分是用来反向传播的局部梯度 (Local Gradient) 计算 dx , 这一部分所有的微调方法都需要且计算量相同; 另一部分是损失函数对待更新参数的梯度计算 dW , dW 是对该层参数的梯度, 用来进行梯度更新, 这部分的计算因需要微调的参数量而异。在全参数微调中, 计算 dx 和 dW 的计算量均与前向计算量相当。

对于 LoRA 类方法, 在所有的线性层中均加入微调参数的情况下, 假定预训练的矩阵为 $W_0 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, LoRA 中待更新参数可表示为:

$$W = W_0 + \Delta W = W_0 + AB \quad (15)$$

其中, $A \in \mathbb{R}^{d_{\text{model}} \times r}$, $B \in \mathbb{R}^{r \times d_{\text{model}}}$, $r \ll d_{\text{model}}$, 在 LoRA 微调过程中, 仅微调权重 A 和权重 B 。假设模型的损失函数为 C , 根据链式法则微调过程中待更新参数 A 和 B 的梯度计算如式 (16) 所示。

$$\begin{aligned} \frac{\partial C}{\partial A} &= \frac{\partial C}{\partial W} \frac{\partial W}{\partial A} = B^T \frac{\partial C}{\partial W} \\ \frac{\partial C}{\partial B} &= \frac{\partial C}{\partial W} \frac{\partial W}{\partial B} = \frac{\partial C}{\partial W} A^T \end{aligned} \quad (16)$$

对全参数微调, 需计算梯度 $\frac{\partial C}{\partial W}$, 而 LoRA 则需要计算梯度 $\frac{\partial C}{\partial A}$ 和 $\frac{\partial C}{\partial B}$ 。对于所有线性层均增加 LoRA 旁路的模型来说, 两者在待更新参数梯度的计算量上差别不大。如图 7 所示, 所有线性层均增加旁路的 LoRA 类微调方法在计算量上与全参数微调差别不大, LoRA 类微调方法的优势在于能够降低模型微调内存占用, 可以通过增加 batchsize 大小来提高微调的吞吐量, 进而提升微调训练速度。

对于 Prompt tuning 微调方法, 因为它只在 Embedding 层即输入层一层有少量的可微调参数,

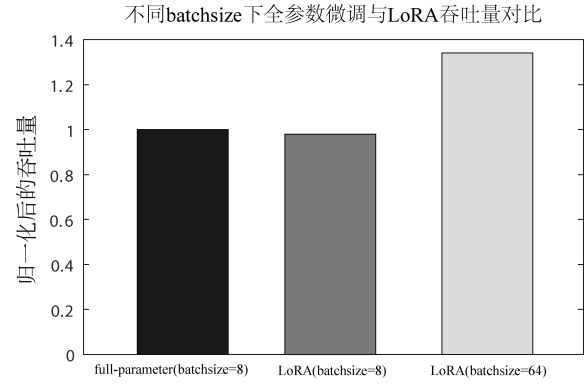


图 7 LoRA 与全参数微调对比^[153]

所以 Prompt tuning 微调方法中对待更新参数的梯度计算远远小于全参数微调方法, 可以忽略。Prompting tuning 总的计算量约为全参数微调的 2/3 左右。

对于 Adapter 类方法, 它在每一层后面都加了一个小的可训练层, 因此对于待更新参数数量的计算量与全参数微调相当。所以, 这类方案并不能显著降低计算量。

通过上述分析可知, 各类参数高效的微调方法对计算量的减少远不如对内存的节约大, 它们之所以能有很好的效果, 得益于以下几个方面: ①更新的参数量少; ②内存占用降低, batchsize 大小增加; ③降低了多卡的数据传输量; ④采用了各种低精度加速技术。

4 总结

本文围绕大语言模型微调方法, 首先介绍了微调的相关概念, 然后基于一个新的分类视角对现有微调方法进行了整理归纳, 分析了每类方法的原理及特点, 并从计算的角度出发, 着重对微调方法进行了内存占用分析。

4.1 微调策略建议

结合微调实践, 从微调工具、微调方法以及微调技巧方面, 对大语言模型微调策略提出建议。

4.1.1 微调工具

当前, 大语言模型微调相关的开源框架数量众多, 从支持的微调方法和开源模型、加速训练以及易用性方面, 这些框架对模型的微调可以起到很大的支持作用, 方便用户基于这些框架开展领域模型的微调实践。例如, LLaMa-Factory 提供基于 Web 界面的模型微调、评估和推理等, 无须用户使用复杂的命令行, 支持 LLaMa、BLOOM、Baichuan 和 ChatGLM 等常见

大语言模型。实践中,也可以调用 HuggingFace PEFT、Adapters 等微调工具包,直接加载可选择的微调方法。截至 2024 年 7 月,HuggingFace PEFT 微调工具包中已经集成 LoRA、Prefix tuning、P-tuning 和 Prompt tuning 等 19 种实践中常用的微调方法,支持对 HuggingFace 库中的模型微调。Adapters 工具包支持 AdapterFusion、Paralleladapter 和 IA³ 等 12 种微调方法,也支持 LoRA 和 Prefix tuning 等方法。这些微调工具包仍然在持续更新。对于开发面向用户的大语言模型驱动的应用程序,可以使用一些基于 LangChain 方式的框架如 Chinese-LLaMa-Alpaca,结合本地知识库的问答和聊天机器人等应用程序。

4.1.2 微调方法

对于微调方法的选择,全参数微调一般会比其他方法的效果更好,但对计算资源要求较高,并且微调时间较长。LOMO 和 AdaLOMO 方法可以减少内存,支持全参数微调。除此之外,也可以根据实际需求侧重考虑选择哪种微调方法。例如:①在计算资源有限的条件下,可以选择一些轻量化后的微调方法,如 QLoRA。LST 方法在更新 Adapter 时不经过主权重,也减少计算资源。原模型上的稀疏方法也可以应用在新增参数上以减少梯度更新的计算资源,如 Sparse adapter。MEFTs 使用内存高效的方法减少计算资源。②对于推理时间消耗要求较高的下游应用,Adapter 系列会增加推理时间,而 LoRA 系列在推理时使用权重合并后的模型,因此相对于基础模型来说,不会增加额外的推理计算开销,可以优先考虑 LoRA 系列方法。③对于长文本的微调,一些微调方法在这方面做了额外拓展,以增加模型可输入的文本长度。例如,LLaMa-2-7B 的上下文长度是 4 096 tokens,通过 LongLoRA 微调,可以将其文本输入长度提升至 100k tokens,极大地增加了模型可处理文本的长度,使模型微调可以应用到对论文和著作等长文本应用。这种情况下,微调方法不仅能针对下游任务,还可以弥补基础模型本身存在的一些限制。

4.1.3 微调技巧

对于基础的 LoRA 方法超参数设置,在计算资源有限的条件下,优先考虑微调模型的 Q 和 V 矩阵,选择默认的秩,也可以达到一个不错的效果。Flash Attention、FSDP 以及 DeepSpeed 分布式训练等技术也可以进一步加速模型微调。另外,根据对模型隐含层及激活的分析,通过模型可解释性的指导进行细粒度的面向任务微调,这一点对于用户来说技术性要求较高。

4.2 未来展望

(1) **更加内存高效的微调方法** 如图 5 所示,现有的部分参数和新增参数微调方法已对 Transformer 架构进行了全面的“修补”,未来针对新的结构性微调空间变得十分有限。随着 LLM 规模的扩大,在资源有限的情况下微调 LLM 变得更加困难,因此微调方法是否更加内存高效,其优先级应比减少微调参数量更大。可以进一步利用跨学科交叉的思想,借助边缘计算中量化和剪枝等技术以及 LLM 预训练中使用的新的内存高效利用技术,加速参数高效微调的创新和进步。

(2) **可解释性指导大模型微调** 现有的微调方法对 LLM 微调位置、参数数量等大都还是凭借经验、实验对比以及前人方法的继承,可解释性不足。已经有一些文献 LLM 的可解释性^[137-138],使用可解释性成果指导模型微调实践,构建具有针对不同类型下游任务自适应每层参数数量的微调系统,是一个值得研究的方向。

(3) **更统一的集成微调框架** LLaMa-Factory 和 HuggingFace PEFT 等微调工具包已经集成了一些参数微调方法,但方法的覆盖面还不全,没有更统一的集成框架也不利于微调方法的推广使用。研究人员在推出新的 LLM 微调方法时,除公开代码外,还应尽量适配集成框架,争取尽快收录以便推广,在交叉碰撞中有助于产生新的高效微调方法。此外,统一的微调框架也有助于不同微调方法的结合使用以及微调可解释性研究,可实现微调方法发展的正向反馈。

(4) **更真实智能的评价数据集** 随着新出 LLM 已经在 GLUE 和 SuperGLUE 等数据集上超过了普通人的水平,再在这些数据集上对比新的微调方法,已不能反映其在真实世界数据上的表现,因此需要发展更贴近真实生活的更智能、更有区分度的数据集或结构化知识图谱,微调方法在对比时也可展示自己下游数据的表现,确保微调方法能使 LLM 应对现实世界的挑战,并提供下游任务更实际的解决方案。

(5) **使用不同规模 LLM 进行评估** 现有的微调方法文献大多沿着前人实验对比的设置,使用小规模 LLM 进行对比分析,较少有方法在新出的大规模公开 LLM 上(例如 LLaMa)做实验,这不利于方法的推广使用。微调方法除了对比推理效果、参与微调的数据量外,还应加入对内存等开销的对比,明确方法的适用范围和适用场景。

(6) **提高 ICL 的稳定性** 随着 LLM 规模的增

大,模型本身蕴含的知识量也逐渐增大,未来推出的 LLM 很可能已经能够基本适配下游任务,在 LongLoRA 等方法进一步拓宽了 LLM 对上下文的接受长度的情况下,只需使用 ICL 等无参数微调方法,即可实现 LLM 对自己下游任务的推理需求。现有 ICL 性能还不稳定,因此对 ICL 的鲁棒性研究是一个十分有意义但具有挑战的问题。

参考文献

- [1] OpenAI. ChatGPT: Optimizing language models for dialogue[EB/OL]. <https://openai.com/research/chatgpt/> [2024-05-31].
- [2] OpenAI. GPT-4 technical report[J/OL]. arXiv: 2303.08774, 2023.
- [3] HOULSBY N, GIURGIU A, JASTRZEBSKI S, et al. Parameter-efficient transfer learning for NLP[C]//Proceedings of the International Conference on Machine Learning. PMLR, 2019: 2790-2799.
- [4] DING N, QIN Y, YANG G, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models[J]. Nature Machine Intelligence, 2023, 5(3): 220-235.
- [5] XU L, XIE H, QIN S Z J, et al. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment[J/OL]. arXiv preprint arXiv: 2312.12148, 2023.
- [6] LIALIN V, DESHPANDE V, RUMSHISKY A. Scaling down to scale up: A guide to parameter-efficient fine-tuning[J/OL]. arXiv preprint arXiv: 2303.15647, 2023.
- [7] DONG Q, LI L, DAI D, et al. A survey for in-context learning[J/OL]. Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2024: 1107-1128.
- [8] ZHANG S, DONG L, LI X, et al. Instruction tuning for large language models: A survey[J/OL]. arXiv preprint arXiv: 2308.10792, 2023.
- [9] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017: 6000-6010.
- [10] KRIZHEVSKY A, SUTSKEVER I, HINTON G. Imagenet classification with deep convolutional neural networks[C]//Proceedings of the 26th Annual Conference on Neural Information Processing Systems, 2012: 1106-1114.
- [11] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 1-9.
- [12] SAK H, SENIOR A, BEAUFAYS F. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition[J/OL]. arXiv preprint arXiv: 1402.1128, 2014.
- [13] STAUDEMEYER R C, MORRIS E R. Understanding LSTM tutorial into long short-term memory recurrent neural networks[J/OL]. arXiv preprint arXiv: 1909.09586, 2019.
- [14] BA J L, KIROS J R, HINTON G E. Layer normalization[J/OL]. arXiv preprint arXiv: 1607.06450, 2016.
- [15] DEVLIN J, CHANG M W, LEE K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[C]//Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019: 4171-4186.
- [16] BROWN T, MANN B, RYDER N, et al. Language models are few-shot learners[C]//Proceedings of the 34th International Conference on Neural Information Processing Systems, 2020: 1877-1901.
- [17] RADFORD A, WU J, CHILD R, et al. Language models are unsupervised multitask learners[EB/OL]. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf #: ~: text = This% 20paper% 20shows% 20that% 20language% 20models% 20can. [2024-05-31].
- [18] DING N, HU S, ZHAO W, et al. Openprompt: An open-source framework for prompt-learning[J/OL]. arXiv preprint arXiv: 2111.01998, 2021.
- [19] LIU P, YUAN W, FU J, et al. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing[J]. ACM Computing Surveys, 2023, 55(9): 1-35.
- [20] SUNDAR P. Introducing Gemini: Our largest and most capable AI model [EB/OL]. <https://blog.google/technology/ai/google-gemini-ai/#sundar-note>. [2024-05-31].
- [21] HENDRYCKS D, BURNS C, BASART S, et al. Measuring massive multitask language understanding [C]//Proceedings of the ICLR, 2021.
- [22] XU D, YEN I E H, ZHAO J, et al. Rethinking Network pruning--under the pre-train and fine-tune paradigm [C]//Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021: 2376-2382.
- [23] MIKHAILOV D I. Optimizing national security strategies through LLM-Driven artificial intelligence integration [J/OL]. arXiv preprint arXiv: 2305.13927, 2023.
- [24] WU S, IRSOY O, LU S, et al. Bloomberggpt: A large language model for finance[J/OL]. arXiv preprint arXiv: 2303.17564, 2023.
- [25] THIRUNAVUKARASU A J, TING D S J, ELANGOVAN K, et al. Large language models in medicine [J]. Nature Medicine, 2023, 29(8): 1930-1940.
- [26] WEN R, WANG T, BACKES M, et al. Last one standing: A comparative analysis of security and privacy of soft prompt tuning, LoRA, and In-context learning[J/OL]. arXiv preprint arXiv: 2310.11397,

- 2023.
- [27] LAMBDA LABS. OpenAI's GPT-3 language model: A technical overview[EB/OL]. <https://lambdalabs.com/blog/demystifying-gpt-3#1>. [2024-05-31].
 - [28] WEN H, LI Y, LIU G, et al. Empowering llm to use smartphone for intelligent task automation[J/OL]. arXiv preprint arXiv: 2308.15272, 2023.
 - [29] FLORIDI L, CHIRIATTI M. GPT-3: Its nature, scope, limits, and consequences[J]. *Minds and Machines*, 2020, 30: 681-694.
 - [30] LIU Y, OTT M, GOYAL N, et al. Roberta: A robustly optimized bert pretraining approach[J/OL]. arXiv preprint arXiv: 1907.11692, 2019.
 - [31] YANG J, JIN H, TANG R, et al. Harnessing the power of llms in practice: A survey on chatgpt and beyond[J]. *ACM Transactions on Knowledge Discovery from Data*, 2023, 18(6): 1-32.
 - [32] TAY Y, DEHGHANI M, TRAN V Q, et al. UL2: Unifying language learning paradigms[C]//Proceedings of the 11th International Conference on Learning Representations, 2022.
 - [33] MUKANS E, BARZDINS G. RIGA at SemEval Task 2: NER enhanced with GPT-3[C]//Proceedings of the 17th International Workshop on Semantic Evaluation, 2023: 331-339.
 - [34] WANG X, AITCHISON L, RUDOLPH M. LoRA ensembles for large language model fine-tuning[J/OL]. arXiv preprint arXiv: 2310.00035, 2023.
 - [35] ZHANG X, RAJABI N, DUH K, et al. Machine translation with large language models: Prompting, few-shot learning, and fine-tuning with QLoRA[C]//Proceedings of the 8th Conference on Machine Translation, 2023: 468-481.
 - [36] DE CURTÒ J, DE ZARZÀ I, ROIG G, et al. LLM-informed multi-armed bandit strategies for non-stationary environments[J]. *Electronics*, 2023, 12(13): 14-28.
 - [37] CHANG K W, TSENG W C, LI S W, et al. Speech-prompt: An exploration of prompt tuning on generative spoken language model for speech processing tasks[J/OL]. arXiv preprint arXiv: 2203.16773, 2022.
 - [38] LIU Y, HAN T, MA S, et al. Summary of chatgpt-related research and perspective towards the future of large language models[J]. *Meta-Radiology*, 2023: 100-107.
 - [39] WANG A, SINGH A, MICHAEL J, et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding[C]//Proceedings of the EMNLP Workshop Blackbox NLP: Analyzing and Interpreting Neural Networks for NLP, 2018: 353-355.
 - [40] SARLIN P E, DETONE D, MALISIEWICZ T, et al. Superglue: Learning feature matching with graph neural networks[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020: 4938-4947.
 - [41] RAJPURKAR P, ZHANG J, LOPYREV K, et al. Squad: 100,000+ questions for machine comprehension of text[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2016: 2383-2392.
 - [42] HASAN T, BHATTACHARJEE A, ISLAM M S, et al. XL-sum: Large-scale multilingual abstractive summarization for 44 languages[C]//Proceedings of the Association for Computational Linguistics: ACL-IJCNLP, 2021: 4693-4703.
 - [43] GUILLOU L, HARDMEIER C, NAKOV P, et al. Findings of the WMT shared task on cross-lingual pronoun prediction[C]//Proceedings of the 1st Conference on Machine Translation, Shared Task Papers, 2016: 525-542,.
 - [44] DODGE J, SAP M, MARASOVIĆ A, et al. Documenting large webtext corpora: A case study on the colossal clean crawled corpus[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2021: 1286-1305.
 - [45] LIAO B, TAN S, MONZ C. Make your pre-trained model reversible: From parameter to memory efficient fine-tuning[C]//Proceedings of the 37th International Conference on Neural Information Processing Systems, 2023: 15186-15209.
 - [46] SUN Y, WANG S, FENG S, et al. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation[J/OL]. arXiv preprint arXiv: 2107.02137, 2021.
 - [47] 王俊, 李石君, 杨莎等. 一种新的用于跨领域推荐的迁移学习模型[J]. *计算机学报*, 2017, 40(10): 2367-2380.
 - [48] CLARK K, KHANDELWAL U, LEVY O, et al. What does bert look at? An analysis of bert's attention[C]//Proceedings of the ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, 2019: 276-286.
 - [49] HUMZA NAVEED, ASAD ULLAH KHAN, SHI QIU, et al. A comprehensive overview of large language models[J/OL]. arXiv preprint arXiv: 2307.06435, 2023.
 - [50] HOULSBY N, GIURGIU A, JASTRZEBSKI S, et al. Parameter-efficient transfer learning for NLP[C]//Proceedings of the 36th International Conference on Machine Learning, 2019: 2790-2799.
 - [51] LI, X.L., P. LIANG. Prefix-tuning: Optimizing continuous prompts for generation[C]//Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, 2021: 4582-4597.
 - [52] LESTER, B., R. AL RFOU, N. CONSTANT, The power of scale for parameter-efficient prompt tuning[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2021: 3045-3059.
 - [53] HU E J, SHEN Y, WALLIS P, et al. Lora: Low-rank adaptation of large language models[J/OL].

- arXiv preprint arXiv: 2106.09685, 2021.
- [54] DETTMERS T, PAGNONI A, HOLTZMAN A, et al. Qlora: Efficient finetuning of quantized llms[C]//Proceedings of the 37th International Conference on Neural Information Processing System, 2023: 10088-10115.
 - [55] ZAKEN E B, RAVFOGEL S, GOLDBERG Y. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models[C]//Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, 2021: 1-9.
 - [56] LV K, YANG Y, LIU T, et al. Full parameter fine-tuning for large language models with limited resources[C]//Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024: 8187-8198.
 - [57] LIAO B, MENG Y, MONZ C. Parameter-efficient fine-tuning without introducing new latency [C]//Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, 2023: 4242-4260.
 - [58] XIE E, YAO L, SHI H, et al. DiffFit: Unlocking transferability of large diffusion models via simple parameter-efficient fine-tuning[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023: 4230-4239.
 - [59] LI Y, WEI F, ZHAO J, et al. Rain: Your language models can align themselves without finetuning[J/OL]. arXiv preprint arXiv: 2309.07124, 2023.
 - [60] ZHAO J, ZHANG Z, CHEN B, et al. Galore: Memory-efficient llm training by gradient low-rank projection[C]//Proceedings of the 41st International Conference on Machine Learning, 2024: 61121-61143.
 - [61] AN S, LI Y, LIN Z, et al. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models[J/OL]. arXiv preprint arXiv: 2203.03131, 2022.
 - [62] ZHU Y, FENG J, ZHAO C, et al. Counter-interference adapter for multilingual machine translation [C]//Proceedings of the Association for Computational Linguistics; EMNLP, 2021: 2812-2823.
 - [63] SUNG Y L, CHO J, BANSAL M. Lst: Ladder side-tuning for parameter and memory efficient transfer learning[J]. Advances in Neural Information Processing Systems, 2022, 35: 12991-13005.
 - [64] LIU X, ZHENG Y, DU Z, et al. GPT understands, too[J/OL]. arXiv preprint arXiv: 2103.10385, 2023.
 - [65] LI X L, LIANG P. Prefix-tuning: Optimizing continuous prompts for generation[C]//Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, 2021: 4582-4597.
 - [66] LIU X, JI K, FU Y, et al. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks[C]//Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, 2022.
 - [67] GUO D, RUSH A M, KIM Y. Parameter-efficient transfer learning with diff pruning[C]//Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, 2021: 4884-4896.
 - [68] ZHAO M, LIN T, MI F, et al. Masking as an efficient alternative to finetuning for pretrained language models[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2020: 2226-2241.
 - [69] SUNG Y L, NAIR V, RAFFEL C A. Training neural networks with fixed sparse masks[J]. Advances in Neural Information Processing Systems, 2021, 34: 24193-24205.
 - [70] HE S, DING L, DONG D, et al. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters[C]//Proceedings of the Association for Computational Linguistics; EMNLP, 2022: 2184-2190.
 - [71] DING N, LV X, WANG Q, et al. Sparse low-rank adaptation of pre-trained language models[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2023: 4133-4145.
 - [72] LIU H, TAM D, MUQEETH M, et al. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning[J]. Advances in Neural Information Processing Systems, 2022, 35: 1950-1965.
 - [73] LV K, YAN H, GUO Q, et al. AdaLomo: Low-memory Optimization with Adaptive Learning Rate [C]//Proceedings of the Association for Computational Linguistics; ACL, 2024: 12486-12502.
 - [74] MALLADI S, GAO T, NICHANI E, et al. Fine-tuning language models with just forward passes[J]. Advances in Neural Information Processing Systems, 2023, 36: 53038-53075.
 - [75] SARKAR S, SARATHI D, RANRAN H, et al. Unified low-resource sequence labeling by sample-aware dynamic sparse finetuning [C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2023: 6998-7010.
 - [76] LIU S Y, WANG C Y, YIN H, et al. DoRA: Weight-decomposed low-rank adaptation [J/OL]. arXiv preprint arXiv: 2402.09353, 2024.
 - [77] MENG X, DAI D, LUO W, et al. PeriodicLoRA: Breaking the low-rank bottleneck in LoRA optimization[J/OL]. arXiv preprint arXiv: 2402.16141, 2024.
 - [78] PFEIFFER J, KAMATH A, RÜCKLÉ A, et al. AdapterFusion: Non-destructive task composition for transfer learning[C]//Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, 2021: 487-503.
 - [79] RÜCKLÉ A, GEIGLE G, GLOCKNER M, et al. Adapterdrop: On the efficiency of adapters in transformers[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2021: 7930-7946.

- [80] KARIMI MAHABADI R, HENDERSON J, RUDER S. Compacter: Efficient low-rank hypercomplex adapter layers[J]. *Advances in Neural Information Processing Systems*, 2021, 34: 1022-1035.
- [81] ZHANG A, TAY Y, ZHANG S, et al. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters[C]//*Proceedings of ICLR*, 2021.
- [82] WANG Y, AGARWAL S, MUKHERJEE S, et al. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning[C]//*Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2022: 5744-5760.
- [83] WANG R, TANG D, DUAN N, et al. K-adapter: Infusing knowledge into pre-trained models with adapters[C]//*Proceedings of the Association for Computational Linguistics: ACL-IJCNLP*, 2021: 1405-1418.
- [84] CHRONOPOULOU A, PETERS M E, FRASER A, et al. Adaptersoup: Weight averaging to improve generalization of pretrained language models[C]//*Proceedings of the Association for Computational Linguistics: EACL*, 2023: 2054-2063.
- [85] ZHAO H, TAN H, MEI H. Tiny-attention adapter: Contexts are more important than the number of parameters[C]//*Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2022: 6626-6638.
- [86] LU X, BRAHMAN F, WEST P, et al. Inference-time policy adapters (ipa): Tailoring extreme-scale lms without fine-tuning[C]//*Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023: 6863-6883.
- [87] EDALATI A, TAHA EI M, KOBYZEV I, et al. Krona: Parameter efficient tuning with kronecker adapter[J/OL]. *arXiv preprint arXiv: 2212.10650*, 2022.
- [88] HU Y, XIE Y, WANG T, et al. Structure-aware low-rank adaptation for parameter-efficient fine-tuning[J]. *Mathematics*, 2023, 11(20): 4317.
- [89] KOPICZKO D J, BLANKEVOORT T, ASANO Y M. VeRA: Vector-based random matrix adaptation[C]//*Proceedings of ICLR*, 2024.
- [90] CHAVAN A, LIU Z, GUPTA D, et al. One-for-All: Generalized LoRA for parameter-efficient fine-tuning[J/OL]. *arXiv preprint arXiv: 2306.07967*, 2023.
- [91] ZHANG Q, CHEN M, BUKHARIN A, et al. Adaptive budget allocation for parameter-efficient fine-tuning[C]//*Proceedings of ICLR*, 2023.
- [92] VALIPOUR M, REZAGHOLIZADEH M, KOBYZEV I, et al. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation[C]//*Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023: 3274-3287.
- [93] CHEN Y, QIAN S, TANG H, et al. Longlora: Efficient fine-tuning of long-context large language models[J/OL]. *arXiv preprint arXiv: 2309.12307*, 2023.
- [94] SHENG Y, CAO S, LI D, et al. S-lora: Serving thousands of concurrent lora adapters[J/OL]. *arXiv preprint arXiv: 2311.03285*, 2023.
- [95] HAYOU S, GHOSH N, YU B. LoRA+: Efficient low rank adaptation of large models[J/OL]. *arXiv preprint arXiv: 2402.12354*, 2024.
- [96] ZHANG R, QIANG R, SOMAYAJULA S A, et al. AutoLoRA: Automatically tuning matrix ranks in low-rank adaptation based on meta learning[C]//*Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2024: 5048-5060.
- [97] FENG W, HAO C, ZHANG Y, ET al. Mixture-of-loras: An efficient multitask tuning for large language models[C]//*Proceedings of the Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, 2024: 11371-11380.
- [98] LI D, MA Y, WANG N, et al. MixLoRA: Enhancing large language models fine-tuning with LoRA based mixture of experts[J/OL]. *arXiv preprint arXiv: 2404.15159*, 2024.
- [99] ZHU W, TAN M. SPT: Learning to selectively insert prompts for better prompt tuning[C]//*Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023: 11862-11878.
- [100] SHI Z, LIPANI A. Dept: Decomposed prompt tuning for parameter-efficient fine-tuning[C]//*Proceedings of the 3rd Workshop on Efficient Natural Language and Speech Processing*, 2023.
- [101] BANDARA W G C, PATEL V M. Attention prompt tuning: Parameter-efficient adaptation of pre-trained models for spatiotemporal modeling[C]//*Proceedings of the 18th International Conference on Automatic Face and Gesture Recognition*, 2024.
- [102] CHEN J, ZHANG A, SHI X, et al. Parameter-efficient fine-tuning design spaces[J/OL]. *arXiv preprint arXiv: 2301.01821*, 2023.
- [103] MAO Y, MATHIAS L, HOU R, et al. Unipelt: A unified framework for parameter-efficient language model tuning[C]//*Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2021: 6253-6264.
- [104] LEWIS P, PEREZ E, PIKTUS A, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks[C]//*Proceedings of the 34th Conference on Neural Information Processing Systems*, 2020: 9459-9474.
- [105] BAKKER M, CHADWICK M, SHEAHAN H, et al. Fine-tuning language models to find agreement among humans with diverse preferences[J]. *Advances in Neural Information Processing Systems*, 2022, 35: 38176-38189.
- [106] FAN Y, JIANG F, LI P, et al. Grammargpt: Exploring open-source llms for native chinese grammatical error correction with supervised fine-tuning

- [C]//Proceedings of the CCF International Conference on Natural Language Processing and Chinese Computing. Cham: Springer Nature Switzerland, 2023: 69-80.
- [107] ROZIERE B, GEHRING J, GLOECKLE F, et al. Code LLaMa: Open foundation models for code[J/OL]. arXiv preprint arXiv: 2308.12950, 2023.
- [108] ZHENG H, SHEN L, TANG A, et al. Learn from model beyond fine-tuning: A survey[J/OL]. arXiv preprint arXiv: 2310.08184, 2023.
- [109] DAO T, FU D, ERMION S, et al. Flashattention: Fast and memory-efficient exact attention with io-awareness[J]. *Advances in Neural Information Processing Systems*, 2022, 35: 16344-16359.
- [110] ZHAO Y, GU A, VARMA R, et al. PyTorch FS-DP: Experiences on scaling fully sharded data parallel[J/OL]. arXiv preprint arXiv: 2304.11277, 2023.
- [111] TAN C, SUN F, KONG T, et al. A survey on deep transfer learning[C]//Proceedings of the Artificial Neural Networks and Machine Learning - ICANN; 27th International Conference on Artificial Neural Networks, 2018: 270-279.
- [112] KOVALEVA O, ROMANOV A, ROGERS A, et al. Revealing the dark secrets of BERT[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019: 4365-4374.
- [113] MICHEL P, LEVY O, NEUBIG G. Are sixteen heads really better than one? [C]//Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019: 14037-14047.
- [114] LEE J, TANG R, LIN J. What would else do? Freezing layers during transformer fine-tuning[J/OL]. arXiv preprint arXiv: 1911.03090, 2019.
- [115] ZHANG Q, CHEN M, BUKHARIN A, et al. Adaptive budget allocation for parameter-efficient fine-tuning[C]//Proceedings of ICLR, 2023.
- [116] WANG S, ZHU Y, LIU H, et al. Knowledge editing for large language models: A survey[J]. *ACM Computing Surveys*, 2023, 57(3): 1-37.
- [117] HE J, ZHOU C, MA X, et al. Towards a unified view of parameter-efficient transfer learning[C]//Proceedings of ICLR, 2022.
- [118] RUNWAL B, PEDAPATI T, CHEN P Y. Parameter efficient finetuning for reducing activation density in transformers[C]//Proceedings of the 37th Conference on Neural Information Processing System. 2023: 1-6.
- [119] ZHANG Q, CHEN M, BUKHARIN A, et al. Adaptive budget allocation for parameter-efficient fine-tuning[C]//Proceedings of ICLR, 2023.
- [120] HUANG C, LIU Q, LIN B Y, et al. LoraHub: Efficient cross-task generalization via dynamic lora composition[C]//Proceedings of the Workshop on Robustness of Zero/Few-Shot Learning in Foundation Models, 2023.
- [121] TOUVRON H, MARTIN L, STONE K, et al. LLaMa 2: Open foundation and fine-tuned chat models[J/OL]. arXiv preprint arXiv: 2307.09288, 2023.
- [122] Liu Z, Lin Y, Cao Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows [C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 10012-10022.
- [123] SU Y, WANG X, QIN Y, et al. On transferability of prompt tuning for natural language processing [C]//Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics; Human Language Technologies, 2021: 3949-3969.
- [124] ZHAO X, LU J, DENG C, et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey[J/OL]. arXiv preprint arXiv: 2305.18703, 2023.
- [125] PETRONI F, ROCKTÄSCHEL T, LEWIS P, et al. Language models as knowledge bases? [C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019: 2463-2473.
- [126] WEI J, WANG X, SCHUURMANS D, et al. Chain-of-thought prompting elicits reasoning in large language models[C]//Proceedings of the 36th Joint International Conference on Neural Information Processing Systems, 2022, 35: 24824-24837.
- [127] WEI J, TAY Y, BOMMASANI R, et al. Emergent abilities of large language models[J/OL]. arXiv preprint arXiv: 2206.07682, 2022.
- [128] SUN T, SHAO Y, QIAN H, et al. Black-box tuning for language-model-as-a-service [C]//Proceedings of the International Conference on Machine Learning. PMLR, 2022: 20841-20855.
- [129] LI Y. A practical survey on zero-shot prompt design for in-context learning[C]//Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing, 2023: 641-647.
- [130] TOUVRON H, LAVRIL T, IZACARD G, et al. LLaMa: Open and efficient foundation language models[J/OL]. arXiv preprint arXiv: 2302.13971, 2023.
- [131] KORTHIKANTI V A, CASPER J, LYM S, et al. Reducing activation recomputation in large transformer models[C]//Proceedings of the 6th MLSys Conference, 2023.
- [132] RAJBHANDARI S, RASLEY J, RUWASE O, et al. Zero: Memory optimizations toward training trillion parameter models[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020: 1-16.
- [133] Transformer Math 101 EleutherAI Blog.[EB/OL]. <https://blog.eleuther.ai/Transformer-math/>. [2024-05-31].
- [134] ZHANG L, ZHANG L, SHI S, et al. LoRA-FA:

- Memory-efficient low-rank adaptation for large language models fine-tuning [J/OL]. arXiv preprint arXiv: 2308.03303, 2023.
- [135] CHEN T, XU B, ZHANG C, et al. Training deep nets with sublinear memory cost[J/OL]. arXiv preprint arXiv: 1604.06174, 2016. <https://arxiv.org/abs/1604.06174>.
- [136] KAPLAN J, MCCANDLISH S, HENIGHAN T, et al. Scaling laws for neural language models[J/OL]. arXiv preprint arXiv: 2001.08361, 2020.
- [137] ZHAO H, CHEN H, YANG F, et al. Explainability for large language models: A survey[J/OL]. arXiv preprint arXiv: 2309.01029, 2023.
- [138] YIN F, VIG J, LABAN P, et al. Did you read the instructions? Rethinking the effectiveness of task definitions in instruction learning[C]//Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, 2023: 3063-3079.
- [139] KWON Y, WU E, WU K, et al. DataInf: Efficiently estimating data influence in LoRA-tuned LLMs and diffusion models[J/OL]. arXiv preprint arXiv: 2310.00902, 2023.
- [140] YANG Z, DAI Z, YANG Y, et al. Xlnet: Generalized autoregressive pretraining for language understanding[C]//Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019: 5753-5763.
- [141] SOLAIMAN I, BRUNDAGE M, CLARK J, et al. Release strategies and the social impacts of language models[J/OL]. arXiv preprint arXiv: 1908.09203, 2019.
- [142] HE P, LIU X, GAO J, et al. Deberta: Decoding-enhanced bert with disentangled attention[J/OL]. arXiv preprint arXiv: 2006.03654, 2020.
- [143] LEWIS M, LIU Y, GOYAL N, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020: 7871-7880.
- [144] CONNEAU A, KHANDELWAL K, GOYAL N, et al. Unsupervised cross-lingual representation learning at scale[C]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, 2019: 31-38.
- [145] SANH V, WEBSON A, RAFFEL C, et al. Multi-task prompted training enables zero-shot task generalization[C]//Proceedings of the ICLR, 2022.
- [146] ZHANG Z, GU Y, HAN X, et al. Cpm-2: Large-scale cost-effective pre-trained language models[J]. AI Open, 2021, 2: 216-224.
- [147] ZHANG S, ROLLER S, GOYAL N, et al. Opt: Open pre-trained transformer language models[J/OL]. arXiv preprint arXiv: 2205.01068, 2022.
- [148] ZENG A, LIU X, DU Z, et al. Glm-130b: An open bilingual pre-trained model[C]//Proceedings of the ICLR, 2023.
- [149] WORKSHOP B S, SCAO T L, FAN A, et al. Bloom: A 176b-parameter open-access multilingual language model[J/OL]. arXiv preprint arXiv: 2211.05100, 2022.
- [150] YANG A, XIAO B, WANG B, et al. Baichuan 2: Open large-scale language models [J/OL]. arXiv preprint arXiv: 2309.10305, 2023.
- [151] LI X, YAO Y, JIANG X, et al. Flm-101b: An open llm and how to train it with \$100 k budget[J/OL]. arXiv preprint arXiv: 2309.03852, 2023.
- [152] TAYLOR, M.E., P. STONE, Transfer learning for reinforcement learning domains: A survey[J]. Journal of Machine Learning Research, 2009. 10: 1633-1685.
- [153] ZHAO, W.X., et al., A survey of large language models[J]. ACM Transactions on Intelligent Systems and Technology, 2023, 15(3): 1-45.
- [154] CUI Y, YANG Z, YAO X. Efficient and effective text encoding for chinese LLaMa and alpaca[J/OL]. arXiv preprint arXiv: 2304.08177, 2023.
- [155] Fine-Tuning LLMs: LoRA or full-parameter? An in-depth analysis with LLaMa 2. [EB/OL]. <https://www.anyscale.com/blog/fine-tuning-llms-lora-or-full-parameter-an-in-depth-analysis-with-LLaMa-2>. [2024-05-31].



吴春志(1991—), 共同第一作者, 博士, 讲师, 主要研究领域为航天发射装备保障、深度学习算法。

E-mail: chunzhi.wu@qq.com



刘鑫(1979—), 通信作者, 博士, 研究员, 主要研究领域为并行算法和应用。

E-mail: yyylx@263.net



赵玉龙(1984—), 共同第一作者, 博士研究生, 助理研究员, 主要研究领域为人工智能基础软件。

E-mail: zhaoyl04@163.com

附表 1 一些微调方法在 GLUE 数据集上的效果对比

Model	PEFT Method	CoLA	SST2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Avg.	
RoB _B	FT	59.07	92.89	88.24/91.58	90.87/90.61	90.81/87.72	86.27	91.07	72.20	84.00/84.00	[5]
	Adapter ^S	63.32	94.31	90.44/93.18	91.25/90.94	90.81/86.55	87.33	92.06	73.56	85.39/85.16	[5]
	Prompt-tuning	49.37	92.09	70.83/81.72	82.44/83.11	82.99/78.35	80.57	80.03	58.12	74.56/75.42	[5]
	Prefix-tuning	59.31	93.81	87.25/91.03	88.48/88.32	87.75/84.09	85.21	90.77	54.51	80.89/80.88	[5]
	(IA) ³	59.58	93.92	87.00/90.52	90.30/90.32	87.99/84.10	83.95	90.88	71.12	83.09/83.05	[5]
	BitFit	61.32	94.72	89.22/92.41	90.34/90.27	88.12/84.11	84.64	91.09	77.98	84.68/84.57	[5]
	Child-Tuning _b	60.33	93.58	89.22/92.20	91.14/90.93	90.98/88.04	87.40	92.20	77.62	85.31/85.29	[5]
	LoRA	62.09	94.04	87.50/90.68	90.66/90.83	88.83/85.21	86.54	92.02	72.92	84.33/84.29	[5]
	AdaLoRA	59.82	93.92	87.99/91.33	90.83/90.73	88.58/84.98	86.26	91.43	70.04	83.61/83.56	[5]
	MAM Adapter	61.42	94.87	89.31/92.21	90.74/90.42	88.31/83.20	86.63	90.19	72.62	84.26/83.95	[5]
	Prefix	47.45±4.47	91.65±0.44	88.09±1.37	87.49±0.63	85.98±0.11	82.58±0.24	85.91±0.44	60.47±0.54	78.7	[121]
	SaLoRA	63.39±1.79	95.14±0.73	90.20±0.74	91.13±0.17	91.46±0.09	87.83±0.04	92.99±0.21	81.01±0.87	86.64	[121]
	VeRA	64.1±1.7	94.5±0.3	89.7±0.8	90.3±0.2			91.9±0.2	75.8±1.8	84.4	[125]
	DyLoRA (best r)	63.32	94.72	92.79	91.36	90.17	87.17	93.56	80.14	86.66	[123]
	S4	63.20	94.80	90	90.4	91.6	87.80	93.40	85.80	87.1	[94]

续表

Model	PEFT Method	CoLA	SST2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Avg.	
RoB _L	FT	65.78	95.54	89.22/92.28	91.74/91.76	89.30/86.68	89.42	93.61	81.23	86.98/87.04	[5]
	Adapter ^S	67.03	96.37	89.94/92.54	92.58/92.42	92.19/88.50	91.00	94.31	85.25	88.58/88.43	[5]
	Prompt-tuning	61.13	94.61	73.04/81.29	78.51/78.99	80.74/75.16	68.15	89.13	60.29	75.70/76.09	[5]
	Prefix-tuning	59.01	95.76	88.24/91.37	90.92/91.07	88.88/85.45	89.30	93.32	74.01	84.93/84.91	[5]
	(IA) ³	61.15	<u>94.61</u>	86.52/90.33	92.22/92.03	89.45/86.25	88.63	94.25	81.23	86.00/86.06	[5]
	BitFit	68.01	96.10	90.93/93.38	91.93/91.77	89.48/86.43	89.98	94.47	87.73	88.57/88.47	[5]
	Child-Tuning _b	63.08	95.07	90.69/93.43	92.36/92.18	91.52/88.75	35.45	93.15	86.25	80.95/80.92	[5]
	LoRA	64.47	96.67	87.50/91.19	91.66/91.44	90.15/86.91	90.76	95.00	79.78	87.00/87.03	[5]
	AdaLoRA	65.85	94.95	89.46/92.34	92.05/91.80	89.60/86.30	90.36	94.62	77.98	86.86/86.78	[5]
	MAM Adapter	67.39	95.81	90.12/92.77	92.44/92.18	90.87/86.65	90.62	94.31	86.62	88.52/88.29	[5]
	Diff Pruning	65.1±2.39	96.4±0.21	90.2±1.11	92.0±0.17	90.3±0.17	90.3±0.08	94.6±0.22	84.5±1.18	87.9±0.69	[72]
	FISH Mask	66.3±1.89	96.1±0.38	88.7±0.92	92.5±0.08	89.8±0.17	90.2±0.08	94.1±0.26	86.3±1.07	88.0±0.61	[72]
	PaFi	70.2±0.46	96.7±0.19	91.4±0.14	91.9±0.24	90.3±0.05	90.2±0.05	94.6±0.08	88.8±0.63	89.3±0.23	[72]
	AdaMix	70.20	97.10	91.90	92.40	92.30	90.90	95.40	89.20	89.90	[92]
	Prefix	60.06±1.44	94.70±0.44	89.85±.88	89.97±3.37	87.57±0.25	88.61±0.12	89.60±0.37	77.33±1.37	84.71	[121]
	SaLoRA	68.37±0.34	96.63±0.28	91.81±1.57	92.43±0.18	91.95±0.08	90.67±0.07	94.98±0.09	87.80±1.29	89.33	[121]
	VeRA	68.0±0.8	96.1±0.1	90.9±0.7	91.7±0.8			94.4±0.2	85.9±0.7	87.80	[125]
	SoRA(r=16)	71.93	95.61	92.00	92.05	92.37/89.84	90.34/90.47	94.11	87.41	89.33	[87][87]
	S4	68.40	96.60	92.30	92.30	92.00	90.80	95.10	87.20	89.30	[94]