运筹学与控制论

DOI:10.11721/cqnuj20250112

# 多任务场景下一种新的大语言模型算法*

王博文[1], 吴至友[1], 郑显达[2], 高 桓[1]
(1. 重庆师范大学 数学科学学院, 重庆 401331; 2. 奥克兰大学 计算机科学学院, 新西兰 奥克兰 1010)

**摘要**:大型语言模型(large language model(s),LLM(s))在多数自然语言处理任务中表现出了卓越的性能。然而,直接应用通用 LLM 往往无法满足特定领域的应用需求。为解决此问题,通常需要通过从头开始训练模型或微调通用模型来定制。从头训练能实现高度定制化,确保与需求匹配并保护数据隐私,但存在成本过高且技术难度大的问题;因此现有方法多通过对通用模型进行微调来提升模型性能,但全参量微调会面临 GPU 内存限制的挑战;现有的参数高效微调技术虽然能够缓解内存限制,但该技术难以同时在多个任务中保持性能,而且在持续微调过程中也可能会出现灾难性遗忘现象。为了解决该问题,提出了一种既能维持多个领域性能又能缓解灾难性遗忘现象的新方法,即基于优化算法的逐层自适应高效合并方法(A layer-wise adaptive and efficient merging method based on black-box optimization,简称 LAEM)。该方法以 LoRA 模块合并的形式进行:首先对多种特定任务中微调过的 LoRA 模块进行去冗余操作;其次,通过引入共享 LoRA 模块的思想,并利用逐层自适应加权平均的方法,将去冗余后的不同任务所对应的 LoRA 模块与共享模块进行合并,LAEM 可以根据模型内部不同层的具体表现和对最终结果的贡献,灵活设定权重,从而更精准地融合多个模型的优势,充分释放模型在各层的潜能,达到更佳的整体性能表现。实验结果表明,LAEM 不仅使模型具备了多种能力,而且在一定程度上缓解了灾难性遗忘的现象,同时避免了传统方法在模型合并时忽略层间特征差异的问题。

**关键词**:大语言模型;参数高效微调;模型合并
**中图分类号**:TP391.1 **文献标志码**:A **文章编号**:1672-6693(2025)01-0026-10

大型语言模型(large language model(s),LLM(s))拥有精心设计的架构和不断增长的规模[1],如 OpenAI GPT[1]、LLaMA[2]、Qwen[3]等大规模预训练语言模型,在语言理解和生成方面拥有强大能力,因此,能够在多种场景下提供高质量的文本生成、问答、摘要等服务,这些模型通过在大规模语料库上进行预训练,积累了丰富的语言知识和通用模式,这使得它们成为解决许多自然语言处理(natural language processing,NLP)任务的范式[4]。然而,尽管通用的 LLM 在广泛的任务上取得了显著成效,但被直接应用于特定领域或任务时,往往难以达到最佳效果,特别是在一些专业领域如医疗、法律等,它们可能缺乏对特定术语和规则的深入理解[5-6],或者在处理敏感数据时存在安全和隐私方面的顾虑。

为了解决这一问题,研究者们采取了 2 种主要策略:从头开始训练模型[7]和微调通用模型[8]。从头训练是指基于特定领域的大量数据,从零开始训练一个全新的模型。这种方法适用于需求极为特定、数据隐私要求极高或希望拥有完全自主知识产权的场景。例如,Bloomberg 公司从头训练了专门用于金融领域任务的大语言模型 BloombergGPT[9],该模型在金融文本处理和分析方面表现出色,能够满足金融行业对准确性和安全性的高要求;佛罗里达大学的研究团队从头训练了专门用于医疗诊断和研究的大型语言模型 GatorTron[10],能够处理复杂的医疗文本和数据,帮助医疗专业人士更高效地进行电子健康记录分析和疾病诊断。从头训练的优点在于完全定制化和确保数据安全,但根据 OpenAI 在 2018 年发布的关于 AI 训练应用的算例需求报告,以及关于 GPT-3 的论文数据[1]可知,训练 GPT-3 需要数千个 A100 GPU 持续 30~60 d,训练成本高达数百万美元;且从头训练还需要对数据进行收集和清理,而高质量的数据集需要去重、清洗、过滤,涉及人工审核和自动化筛选,成本也可达数百万美元;除此之外,还需要投入大量的时间成本和人力成本,因此从头训练大规模语言模型的成本十分高昂。

* **收稿日期**:2024-10-30 **网络出版时间**:2025-03-19T09:07
**资助项目**:国家自然科学基金面上项目(No. 12371258);重庆市自然科学基金创新发展联合基金项目(No. CSTB2023NSCQ-LZX0056)
**第一作者简介**:王博文,女,研究方向为最优化理论与算法,E-mail:2022110510033@stu.cqnu.edu.cn;**通信作者简介**:吴至友,女,教授,博士生导师,E-mail:zywu@cqnu.edu.cn
**网络出版地址**:https://link.cnki.net/urlid/50.1165.N.20250318.1527.006

鉴于上述局限性，研究者们多采用微调策略。微调是指在已有的预训练模型基础上，使用特定领域的少量数据进行进一步训练，以优化模型在该领域的表现。其中，全参数微调[11]是最直接的方法，它通过对模型的所有参数进行更新，以便更好地适应特定任务。然而，这种方法也需要大量的计算资源和 GPU 内存，对于大规模模型而言尤其具有挑战性。近些年来，LoRA 等参数高效微调技术[12-13]已成为流行的解决方案。LoRA 引入了低阶矩阵来调整特定层的模型参数，在实现有效微调的同时减少了内存使用。然而，随着多任务学习需求的增加，LoRA 等参数高效微调技术也面临着新的挑战。如何在多个任务中同时保持高性能，避免灾难性遗忘现象，以及实现模型参数的灵活整合，仍然是一项重大挑战[14]。

为应对这一挑战，常见方法是直接将所有的 LoRA 模块进行合并[15]。然而不同模型在学习过程中捕获了不同或重叠的特征表示，它们的输出空间可能不完全一致，会产生参数冲突，如果直接合并可能会忽略这些差异，导致 LoRA 的独特性丧失，从而使融合后的表示达不到预期效果[15]。

此外，直接合并的另一个问题是忽略了不同 Transformer 层的重要程度，不同的 LoRA 层对最终结果的贡献程度不同[16]。具体来说，不同的 LoRA 层提供了不同的效果，它们的组合构成了整体的可学习特征。因此直接合并会导致模型内部结构中不同层所学习的特征被忽略，从而限制模型潜力的充分发挥，无法针对特定任务需求进行最优化的调整。

基于上述的 2 个缺陷，本文认为应该为需要处理各种边缘情况和细粒度信息的层分配不同的权重，即不同 LoRA 模块的权重系数不必在所有 Transformer 层中都相同。为此提出了一种新的合并方法，即基于优化算法的逐层自适应高效合并方法（A layer-wise adaptive and efficient merging method based on black-box optimization，简称 LAEM）。与以往将 LoRA 模块直接合并的方法不同，本文的方法会根据不同的 Transformer 层以及不同 LoRA 模块的独特性，自适应的学习每个 Transformer 层内不同 LoRA 模块的融合权重并进行逐层合并，因此合并的方式会更灵活。此外，本文还在不同的 LoRA 模块进行逐层自适应加权合并之前，对每个 LoRA 模块进行了去冗余操作，通过消除模块中的冗余参数[17]，来避免合并时多个模块之间的参数冲突。本文使用 Qwen 为基座模型在 SST-5 数据集上进行了实验，以证明逐层自适应加权合并方法在不同配置下的有效性。

综上所述，本文的贡献如下：

1）在 Transformer 模型上提出了一种新的合并方法，即 LAEM。该方法具有灵活的分层 LoRA 分配功能，可自适应学习每个层中不同 LoRA 模块的融合权重系数，以进一步研究和拓展 LoRA 在现实生活场景中的应用。

2）在不同的 LoRA 模块合并之前，通过消除模块中参数影响力较小的一部分来达到去冗余的目的，以此来避免多个模块之间的参数冲突。

3）在部分典型 NLP 任务上进行了实验。实验结果表明，LAEM 可以提高 LoRA 混合性能，同时缓解现有高效微调方法在持续微调中所带来的灾难性遗忘的问题。

## 1　研究现状

在介绍本文的方法之前，首先简要回顾一下参数高效微调方法和模型合并的常见用法。

### 1.1　参数高效微调

在实际应用中往往需要利用特定任务的标注数据对大规模预训练模型进行微调，以进一步提高模型性能，这在 NLP 领域已经成为了标准范式。最直接的方法是全参量微调[11]，它使用特定领域的数据重新训练预训练模型的所有参数，需要过多的计算资源和训练时间，训练成本非常高。因此参数高效微调[13]成为目前比较主流的微调方案，它插入一个微小的可训练适配器模块，通过更新该模块参数并冻结预训练模型的主要参数来节省内存。最具代表性的方法是 Adapter[18]、AdaLoRA[19]、Prompt-Tuning[20]、Prefix-Tuning[21] 和 LoRA[12]。Adapter 旨在添加小的适配器模块在预训练模型的每一层后，通过微调这些模块来保持预训练模型权重不变。Prompt-Tuning 以及 Prefix-Tuning 旨在添加一个可训练的连续提示前缀序列来调整模型的行为，而不直接修改模型的参数。LoRA 在预训练模型的线性层之间添加一个低秩矩阵，旨在将模型权重的改变限制在一个低秩子空间内。

上述高效微调方法虽然都极大限度地减少了计算资源，但是经实验验证，这些方法在持续学习过程中均出

现了灾难性遗忘现象。

### 1.2 模型合并

模型合并涉及将2个或多个具备特定任务能力的模型结合成1个具备多种能力的模型,通过整合不同模型的优点来增强模型的鲁棒性和通用性。平均合并[22]是最经典的模型合并方法,它根据给定的权重进行简单的加权平均来合并模型。SLERP[23]方法通过对2个模型的参数进行球面线性插值,可以保持模型参数空间的几何特性,但是该方法1次只能合并2个模型。传统的模型合并在处理不同模型参数之间会受到不同的干扰。当合并的模型个数过多时,会导致模型性能的下降。Ties-Merging[24]方法通过识别并消除(稀疏化)特定任务模型中的冗余参数(非重要参数),并应用参数符号共识算法来解决模型之间的冲突和干扰。DARE[17]方法通过将微调过程中变化的delta参数随机设置为0,并通过重新调整权重来保证模型预期性能大致不变。passthrough[25]方法通过将模型的某些层或组件被视为"无操作"(no-op),采用非常规的方式将2个或多个预训练模型的部分结合在一起。LoRAHub[26]方法通过无梯度算法来优化LoRA模块的权重系数,从而高效地组合多种LoRA模块。

然而上述方法都是按照不同的比例对多个模型进行直接合并,会忽略模型内部结构中不同层学习的特征,从而可能会限制模型潜力的充分发挥,无法针对特定任务需求进行最优化的调整。

## 2 LoRA参数高效微调

给定一个由 $\boldsymbol{W}_0\in\mathbf{R}^{d\times k}$ 参数化的密集神经网络,为了使之适应下游任务,这里用 $\Delta\boldsymbol{W}\in\mathbf{R}^{d\times k}$ 来更新它,可得到一个由 $\boldsymbol{W}=\boldsymbol{W}_0+\Delta\boldsymbol{W}$ 参数化的更新层,对于全参量微调,$\Delta\boldsymbol{W}$ 是基于该层所有 $d\times k$ 参数进行梯度计算,这需要耗费过多的计算资源和训练时间,训练成本非常高。为了降低计算成本、提高计算效率并使得大语言模型能够在短时间内完成训练,LoRA将 $\Delta\boldsymbol{W}$ 分解成2个秩为 $r$ 的低秩矩阵 $\boldsymbol{B}\in\mathbf{R}^{d\times r}$ 和 $\boldsymbol{A}\in\mathbf{R}^{r\times k}$,维度小于 $d$ 和 $k$,即 $\boldsymbol{W}=\boldsymbol{W}_0+\boldsymbol{BA}$,其中 $r\ll\min\{d,k\}$。这里矩阵 $\boldsymbol{A}$ 使用的是随机高斯分布初始化,矩阵 $\boldsymbol{B}$ 使用的是全零初始化。通过这种低秩分解,LoRA将模型更新的参数量降低为 $r\times(d+k)$,远小于 $d\times k$。

## 3 主要内容

为最小化上述高效微调方法在持续学习过程中出现灾难性遗忘现象的影响,使模型能同时掌握多个领域的特定知识,本文采用模型融合策略,旨在避免持续微调过程中新任务的学习对旧任务的影响,通过整合的方式更新模型,以确保旧知识得以保留。同时,为了避免传统模型合并方法中因权重均等化分配而忽视各层特征差异,导致模型的性能受损,本文为需要处理各种边缘情况和细粒度信息的层分配不同的权重,即不同LoRA模块的权重系数不必在所有Transformer层中都相同。

因此,本文将LoRA与自适应的层级权重分配机制相结合,引入了LAEM。由于目前大多数大语言模型均采用Transformer的架构,因此本文研究如何将LAEM应用于Transformer模型。不同于传统的模型合并方法在Transformer的所有层上对LoRA分配相同的权重,本文的方法采用自适应机制动态调整不同层上多个LoRA模块的权重系数。

### 3.1 LAEM流程

本文提出的LAEM框架如图1所示,具体步骤如下。

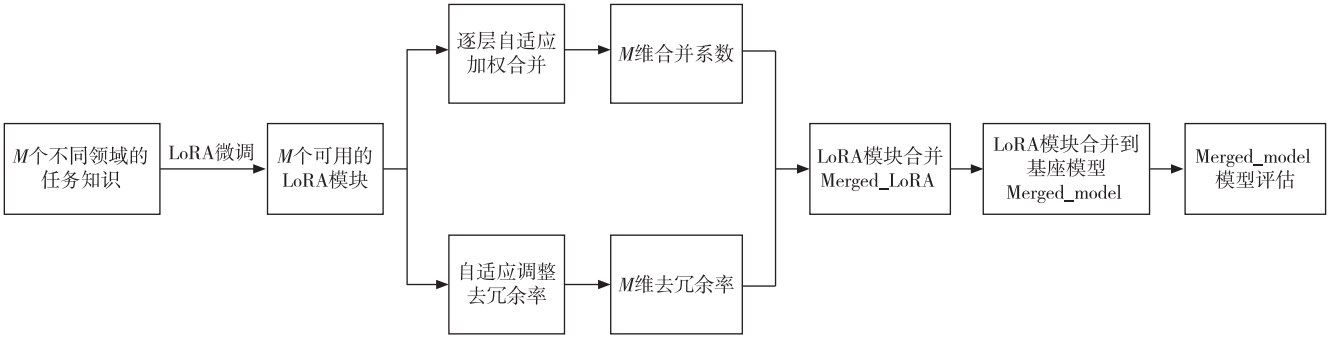


**图1 LAEM框架图**

**Fig. 1 LAEM diagram**

步骤 1，利用给定的一组任务$\{T_1,T_2,\cdots,T_M\}$，使用已有的 LoRA 参数高效微调方法对 Qwen 模型进行微调，得到利用不同任务微调所对应的 $M$ 个 LoRA 模块$\{\text{LoRA}_1,\text{LoRA}_2,\cdots,\text{LoRA}_M\}$。

步骤 2，根据本文提出的 LAEM，计算将多个 LoRA 模块融合为一个统一的、功能更加强大的 LoRA 模块的权重系数。具体来说，引入了 $M$ 个权重系数，标记为$\{\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_M\}$，每一个权重系数 $\boldsymbol{w}_i$ 对应一个原先独立的 LoRA 模块。此外，这里的每一个权重系数 $\boldsymbol{w}_i$ 本身并非单一数值，而是一个向量，它包含了针对第 $i$ 个 LoRA 模块中每一层的独立权重。本文将 LoRA 模块和对应的权重系数进行加权合并，在全部任务的少数样本上评估性能，并用无梯度优化算法来更新权重系数，经过 $N_w$ 个步骤迭代后，得到最优的权重系数。

步骤 3，根据本文提出的自适应调整去冗余率算法，对于每个 LoRA 模块都进行去除不同冗余参数的操作。具体来说，本文使用去冗余率$\{k_1,k_2,\cdots,k_M\}$共 $M$ 个参数，得到经过去除冗余参数后的 LoRA 模块，即$\{\text{LoRA}'_1,\text{LoRA}'_2,\cdots,\text{LoRA}'_M\}$，结合上述阶段所寻找的最优权重系数对经过去冗余操作后的 LoRA′模块进行合并，在全部任务的少数样本上评估性能，并用无梯度优化算法来更新去冗余率，经过 $N_k$ 个步骤迭代后，得到最优的去冗余率。

步骤 4，根据步骤 2 和步骤 3 得到的最优权重系数和最优去冗余率，对 $M$ 个可用的 LoRA 模块进行逐层加权合并，得到一个具有综合性能的 LoRA 模块。随后，将综合的 LoRA 模块合并到 Qwen 基座模型中，以增强 LoRA 模块在特定任务上的表现，同时保持它在其他任务上的泛化能力。

步骤 5，进行 SST-5 任务，对分类得到的语义预测结果进行评估，计算评估指标。

本文主要增加了逐层自适应加权合并算法和自适应调整去冗余率算法，因此下面将重点介绍这 2 个算法。

### 3.2　逐层自适应加权合并算法

本文在合并的每个 LoRA 模块中分离出了一个共享 LoRA 模块，该模块是用所有任务的少量样本训练而成的，可以学习整个数据集中的一个普遍规律[27]，而其他 LoRA 专家模块则专注于学习特定数据子集。

在合并过程中，本文将每层不同的 LoRA 专家模块的权重系数视为一个 $M$ 维参数，逐层自适应加权合并算法可以通过最小化任务数据的交叉熵损失函数来学习该参数，它针对不同任务数据来优化每一层 LoRA 的合并系数，即第 $L$ 层的组合 LoRA 模块可以表示为：

$$\boldsymbol{A}^L_{\text{merge}}=\sum_{i=1}^{|M|}\boldsymbol{w}_i^L\boldsymbol{A}_i^L,\boldsymbol{B}^L_{\text{merge}}=\sum_{i=1}^{|M|}\boldsymbol{B}_i^L,$$

则最终合并得到的 LoRA 模块可以表示为：

$$\boldsymbol{U}=(\boldsymbol{w}_1\boldsymbol{A}_1+\boldsymbol{w}_2\boldsymbol{A}_2+\cdots+\boldsymbol{w}_M\boldsymbol{A}_M)(\boldsymbol{B}_1+\boldsymbol{B}_2+\cdots+\boldsymbol{B}_M)。$$

这里每个 $\boldsymbol{w}_i$ 都是一个向量，在该向量中每个元素代表对应层的 LoRA 模块在合并过程中所分配的特定权重系数。确保了各层能根据重要性和贡献度在模型融合时得到最优的权重分配。

在优化阶段，本文的目标是找到一组最优的权重集$\{\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_M\}$，以创建一个具备多种特定任务能力的单一模型。黑盒优化技术可以被用来寻找最优权重[28]，本文以最小化交叉熵损失函数为导向，建立如下优化模型：

$$\arg\min_{\boldsymbol{w}} f(\boldsymbol{w})=E_{(x,y)\sim T}[\ell(M^*(x;\boldsymbol{w}),y)],$$

$$\text{s.t. } 1\geqslant\boldsymbol{w}_i^L\geqslant 0,i=1,2,\cdots,M;L=1,2,\cdots,N。$$

其中：$\ell$ 是损失函数，用于衡量模型 $M^*$ 在输入 $x$ 上的预测与真实标签之间的差距 $y$；$E$ 表示期望，即模型在任务 $T$ 上的平均损失；$M$ 是需要合并的模型个数；$N$ 是模型的总层数。

在求解过程中，本文采用无梯度优化的方法来寻找最优权重集。在无梯度优化方面，主要使用(1+1) Evolution Strategy (OneplusOne)[29]，这是一种逐步改进解的简单进化算法。算法开始时，随机初始化一个解，然后在每一次迭代中，通过在当前解上应用一个变异操作(通常是加上一个随机扰动，如高斯噪声)来生成一个新的候选解。如果新解的目标函数值优于当前解，那么新解就取代旧解，成为下一次迭代的起点；否则，算法继续使用当前解，并可能调整变异操作的尺度(如减小变异的步长)，以便更细致地搜索。相较广泛使用的 CMA-ES[30]算法，本文使用的 OneplusOne 每次迭代只涉及一个解的评估和一个变异操作，这就大大减少了计算资源的需求，这在目标函数评估非常昂贵、计算资源十分有限的情况下非常有用。

### 3.3　自适应调整去冗余率算法

对于每个经过 LoRA 参数高效微调的模型，本文用 $\boldsymbol{\theta}$ 表示微调参数，$\boldsymbol{\theta}_{\text{init}}$ 表示预训练参数。Yu 等人[17]的研

究结果表明,监督微调过程中的 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 参数值大多都是冗余的,删除这些值可以产生一个低秩和极为稀疏的 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 参数集,且几乎不会影响模型的性能,因此在合并过程中忽略这些参数值可以在不影响任务性能的情况下避免对有影响的参数造成干扰。基于这一观察,本文对不同 LoRA 模块在每一层的参数向量均进行"修剪",只保留最大值的前 $k\%$,并将其余 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 参数值重置为初始值 0。本文改进了 DARE 方法中的随机替换的做法,并利用无梯度优化算法,最小化特定任务数据的交叉熵损失函数来优化 top-$k$ 的 $k$ 值。具体"修剪"步骤如下。

首先,定义一个与张量 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 形状相同但全为零的张量 $\boldsymbol{X}$,并展平为一维向量 $\boldsymbol{m}$。即 $\boldsymbol{m}=\text{vec}(\boldsymbol{X})=\{m_i=0\}_{i=1}^{N}$,其中 $N$ 是张量 $\boldsymbol{X}$ 的元素总数,vec(·)是向量化操作,即将矩阵展平为列向量。

其次,根据定义的密度参数 $k\%$和张量元素总数 $N$,计算要保留的元素数量 $K$。即 $K=\lfloor k\cdot N\rfloor$,其中$\lfloor\cdot\rfloor$表示向下取整。

接着,计算 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 的绝对值,然后将该值展平并找到绝对值最大的前 $K$ 个元素的索引集 $\boldsymbol{V}$,即$(\boldsymbol{V},\boldsymbol{I})=S(|\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}|_{\text{flat}},K)$,其中 S(·)表示选择$|\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}|_{\text{flat}}$ 中绝对值最大的前 $K$ 个元素,$\boldsymbol{V}$ 是包含 $K$ 个最大绝对值的张量,$\boldsymbol{I}$ 是这些值在展平后的 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 中的索引位置。

然后,设置二进制掩码,将 $\boldsymbol{X}$ 中对应于 $\boldsymbol{I}$ 的元素设置为 1,其他元素保持为 0。即 $\boldsymbol{X}[i]=\begin{cases}1, i\in\boldsymbol{I}\\0,\text{其他}\end{cases}$。

最后,将二进制掩码 $\boldsymbol{X}$ 重塑回原始张量 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 的形状,并与 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 进行逐元素乘法。即$\{\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}\}'=\{\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}\}\odot\boldsymbol{X}_{\text{reshaped}}$,其中 $\boldsymbol{X}_{\text{reshaped}}$ 表示将 $\boldsymbol{X}$ 从一维重塑回 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 原始形状的结果。

最终结果$\{\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}\}'$即为经过修剪的张量,其中只有 $K$ 个最大绝对值的元素被保留,其余元素被设置为 0。

# 4 实验结果及分析

## 4.1 实验设置

### 4.1.1 基座模型

本文使用 Qwen1.5-1.8B-Chat[31]作为基座模型,它是一种基于 Transformer 的纯解码器语言模型,在大量数据上对模型进行了预训练,并使用监督微调和直接偏好优化对模型进行了后训练。

### 4.1.2 数据集

为了评估方法的有效性,采用 SST-5[32]数据集对模型进行评估。该数据集是一个用于情感分析的基准数据集,由斯坦福大学的研究者开发。这个数据集是从电影评论中收集句子,旨在进行细粒度的情感分类,即不仅仅是积极或消极的二元分类,而是分为非常消极、消极、中性、积极、非常积极 5 类不同的情感等级,SST-5 数据规模如表 1 所示,数据集分为 3 个部分:8 544 条用于训练,1 101 条用作验证集,2 210 条用于评估模型的泛化能力。本文将该数据集的非常消极和消极这 2 类合并,称为消极类数据集;将中性、积极和非常积极合并,称为积极类数据集。

**表 1 SST-5 数据集规模**

**Tab.1 Size of SST-5 data set**

| SST-5 | 训练集 | 验证集 | 测试集 |
|---|---|---|---|
| 消极类数据集 | 3 310 | 428 | 912 |
| 积极类数据集 | 5 234 | 673 | 1 298 |
| 合计 | 8 544 | 1 101 | 2 210 |

### 4.1.3 实验测度

在自然语言处理领域,准确率作为一种基准评估指标,被广泛应用于分类任务的性能衡量中[33-34]。因此本文中同样采用准确率作为主要的评估指标。准确率定义为正确分类的样本数占总样本数的比例,它直观地反映了模型在 SST-5 分类任务中的性能。

### 4.1.4 基线

使用上述 SST-5 消极类训练数据集和积极类训练数据集对 Qwen1.5-1.8B-Chat 基座模型进行微调,并对微

调或合并后的模型在这 2 类的测试数据集上进行评估。在所有的实验中，为了评估 LAEM 算法的有效性，比较了各种方法调整后的模型性能：Full-Tuning 方法对模型的所有参数进行全面的微调；QLoRA 方法对模型权重进行量化并在低秩分解矩阵上进行梯度更新；Average merging 方法对多个模型的权重进行平均，这些模型由相同的预训练模型初始化；LoRAHuB 方法利用黑盒优化工具合并多个现有的 LoRA 模块并生成一个新的 LoRA 模块。此外，测试了使用本文方法进行调整的模型，其中包括 2 个变体：1）粗粒度权重融合：在对应的 LoRA 微调模块之间整合权重，实现较高抽象层次的表示学习合并；2）细粒度权重融合：侧重于在单个 LoRA 微调模块内的层间进行权重融合，实现参数学习的更细致、更丰富的组合。

#### 4.1.5　超参数设置

在 LoRA 方法中，将 LoRA 模块与 Qwen 的前馈神经网络中的线性层结合在一起。具体来说，即在 Qwen 的 gate_proj，up_proj，down_proj 模块中插入 LoRA 模块。在 LoRA 适配器的设置中，LoRA r 设置为 128，LoRA alpha 设置为 128，LoRA dropout 设置为 0.05，学习率为 2e－5。由于所用的数据集较小，因此用于微调的每个节点的批次大小设置为 4。本文所有实验都是在配备了 4 个 A100-40G GPU 的服务器上进行，使用 Python 中的 Pytorch 实现代码。

### 4.2　LAEM 实验结果

在大型语言模型中，特别是现在的多层 Transformer 架构中，每一层负责不同的信息处理和特征提取，早期层可能更多关注于低级特征（如语法结构、词频等），而后期层则更侧重于高层语义（如情感、主题等），不同模型在这些层次上的表现都存在差异。本文的 LAEM 充分挖掘了基于 Transformer 模型多层架构的潜力，为模型在复杂任务上的表现提供了更精准的控制和更高效的执行路径。

在表 2 的结果中，LoRA 消极、LoRA 积极分别是利用 SST-5 所划分的消极、积极子类训练出来的 LoRA 模块。后面所使用的合并方法均是在这 2 种模块和共享 LoRA 模块上进行操作的。由于本文关注模型在多任务上的整体表现，因此在整个实验过程中，最终对比的均是综合性能的准确率结果，准确率数值越大，表示模型处理多任务的性能越好。

**表 2　不同任务训练的 LoRA 模块实验结果**

**Tab. 2　Experimental results of LoRA module with different task training**

| | 积极子类任务 | 消极子类任务 | 综合性能 |
|---|---|---|---|
| LoRA 消极 | 0 | 69.51 | 29.50 |
| LoRA 积极 | 65.87 | 0 | 38.00 |

表 3 展示了 LAEM、微调方法以及通过 Average merging、LoRAHub 方法调整的模型的性能，其中每个数字是每个数据集的准确率。从表中结果可以看出，本文的 LAEM 优于另 2 个微调方法 Full tuning 和 QLoRA 的基线方法。并且，LAEM 在总的综合性能测试任务上的性能比 Full tuning 高出 3.56%，比 LoRAhub 高出 6.82%。这证实了 LAEM 中所涉及的逐层权重融合策略在增强模型跨任务表现能力方面的有效性。通过 LAEM 合并不同任务的模型权重，能够提升模型在处理多样化任务时的性能和适应性。

此外，本文还在数学数据集 Math、推理数据集 Reasoning 和多选问答数据集 MMLU 验证了 LAEM 的有效性，本文对比了 LoRA 参数高效微调方法以及 Average merge 方法，结果见表 4。

由表 4 的结果可以验证在 Math、Reasoning、MMLU 数据集上，LAEM 的平均性能会比 LoRA 参数高效微调方法性能提升 3.11%，比传统的模型合并方法性能提升 0.71%。

### 4.3　消融实验

#### 4.3.1　逐层权重合并对模型性能的影响

由于 Transformer 中不同层在不同任务或数据集上的表现不同，本文通过对比粗粒度权重融合和细粒度权重融合的模型性能，来验证逐层权重合并的必要性。

**表3 LAEM与传统方法结果对比**

**Tab.3 Comparison of results between LAEM and traditional methods**

| 方法 | 积极子类任务 | 消极子类任务 | 综合性能 |
|---|---|---|---|
| Full-tuning | 49.00 | 58.55 | 54.48 |
| QLoRA | 52.31 | 63.37 | 57.84 |
| Average merging | 23.42 | 22.69 | 23.17 |
| LoRAHub | 0 | 69.19 | 29.14 |
| Average merging+shared expert | 55.47 | 57.24 | 56.06 |
| LoRAHub+shared expert | 52.00 | 48.57 | 51.22 |
| LAEM | 55.55 | 62.50 | 58.04 |

**表4 LAEM在推理数据集上与传统方法结果对比**

**Tab.4 Comparison of LAEM results with traditional methods on inferential datasets**

| 方法 | 数据集分类 | | | 平均性能 |
|---|---|---|---|---|
| | Math | Reasoning | MMLU | |
| LoRA | 12.21 | 75.05 | 46.22 | 44.49 |
| Average merging | 21.45 | 74.04 | 45.19 | 46.89 |
| LAEM | 23.35 | 73.88 | 45.57 | 47.60 |

在本文中,粗粒度权重融合与细粒度权重融合的主要区别在于它们在模型参数合并过程中的操作级别和细致程度不同。粗粒度融合侧重于模型整体间的权重合并,通过模型级别的策略来平衡不同部分的贡献;而细粒度融合更关注模型内部特定层内或跨层的权重合并,通过精细调整每一层的权重分配,根据各层在特定任务中的表现,实现资源的有效分配。例如,若模型A在处理噪声数据的某一层表现出色,而模型B在同层稍逊一筹,通过对模型不同层进行权重调整,可以强化模型A在该层的影响力,同时减少模型B在这一层的计算投入,从而提升模型整体的效能。

表5对比了粗粒度权重融合和细粒度权重融合所对应的模型性能,其中每个数字是每个数据集的准确率。

**表5 粗/细粒度权重融合实验结果对比**

**Tab.5 Comparison of coarse/fine-grained weight fusion experiment results**

| 实验结果 | 积极子类任务 | 消极子类任务 | 综合性能 |
|---|---|---|---|
| 粗粒度权重融合 | 55.16 | 63.05 | 57.78 |
| 细粒度权重融合 | 55.55 | 62.50 | 58.04 |

由表5可知,在SST-5任务上,基于细粒度权重融合方法比基于粗粒度权重融合方法的性能高出了0.26%。这一成果表明,通过深入到模型内部每一层的权重融合能够更精确更有效地均衡模型不同层对最终预测结果的贡献。通过这样的权重调整,能够更有效地平衡模型各层之间的信息流动,减少信息丢失或冗余,从而提升模型的泛化能力和预测准确性。

### 4.3.2 不同去冗余率对模型性能的影响

为了进一步探索模型参数去冗余率促进模型合并性能方面的实际效果,在SST-5数据集的基础上进行消融实验,从而研究模型参数不同去冗余率对模型粗粒度权重融合和细粒度权重融合的贡献程度,具体结果如表6所示,表中去冗余率15%表示将LoRA模块中最不重要的前15%的参数值重置为初始值0。这里设置了不同的去冗余率,以验证不同去冗余率对模型性能的影响。

**表 6　不同去冗余率对模型性能的影响**

**Tab. 6　Impact of different de-redundancy rates on model performance**

| 粗粒度去冗余率 | 综合性能 | 细粒度去冗余率 | 综合性能 |
|---|---|---|---|
| 0 | 57.60 | 0 | 57.65 |
| 15% | 57.73 | 15% | 57.82 |
| 20% | 57.64 | 20% | 57.87 |
| 30% | 57.64 | 30% | 57.69 |
| 40% | 57.37 | 40% | 57.55 |
| 50% | 56.56 | 50% | 56.78 |

从表 6 可以看出，通过去除不同比率的 $\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{init}}$ 参数，模型性能会受到较大的影响，因此可以证明对模型参数进行适当去冗余能促进合并性能的有效性。由于表 6 中粗粒度去冗余率 15%和细粒度去冗余率 20%为各自效果最好的情况，因此在下面的实验结果中，本文只对比这两者与下述所提算法的性能。

此外，为验证本文所提出的自适应优化去冗余率算法的有效性，本文做了如下实验。由表 7 可知，与未去冗余参数或上述所设的去冗余参数相比，本文所提出的自适应优化去冗余率算法，结合粗粒度权重融合和细粒度权重融合方法，使得 SST-5 的相关指标有提升；尤其是细粒度权重融合与自适应优化去冗余率算法的结合通过去除 10.1%的冗余参数，可以使 SST-5 的相关指标提升了 0.39%。这进一步表明了通过去除冗余参数，可以有效净化模型结构，减少不必要的计算负担，进而增强了模型在处理数据时的效率与准确性。

**表 7　自适应与手动设置去冗余率实验结果对比**

**Tab. 7　Comparison of experimental results between adaptive and manual de-redundancy rates**

| 分类 | 设置方式 | 去冗余率 | 综合性能 | 分类 | 设置方式 | 去冗余率 | 综合性能 |
|---|---|---|---|---|---|---|---|
| 粗粒度 | 手动设置 | 0 | 57.60 | 细粒度 | 手动设置 | 0 | 57.65 |
| | | 15% | 57.73 | | | 20% | 57.87 |
| | 自适应优化 | 6.2% | 57.78 | | 自适应优化 | 10.1% | 58.04 |

## 5　结论

本文提出的 LAEM 框架运用了逐层自适应加权融合机制，以整合多个针对特定任务训练的 LoRA 模块，从而在 LLMs 中实现高效且灵活的模块集成。LAEM 框架能够基于少量示例数据的引导，动态地、差异化地为 LoRA 模块内部的不同层分配权重系数，有效规避了传统模型合并方法中因权重均等分配而可能忽略的层间特征差异问题。LAEM 不仅展现了在快速适应多任务场景下的卓越能力，还通过促进 LoRA 模块的复用与灵活组合，降低了训练成本，推动了构建更加通用、适应性强的 LLMs 的发展。

**参考文献：**

[1] BROWN T B, MANN B, RYDER N, et al. Language models are few-shot learners[EB/OL]. (2020-07-22)[2024-10-30]. https://arxiv.org/abs/2005.14165.

[2] TOUVRON H, LAVRIL T, IZACARD G, et al. LLaMA: open and efficient foundation language models[EB/OL]. (2023-02-27)[2024-10-30]. https://arxiv.org/abs/2302.13971.

[3] BAI J Z, BAI S, CHU Y F, et al. Qwen technical report[EB/OL]. (2023-09-28)[2024-10-30]. https://arxiv.org/abs/2309.16609.

[4] YU L, CHEN Q, ZHOU J, et al. MELO: enhancing model editing with neuron-indexed dynamic LoRA[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2024, 38(17): 19449-19457.

[5] SINGHAL K, TU T, GOTTWEIS J, et al. Toward expert-level medical question answering with large language models[J]. Nature Medicine, 2025, 31: 943-950.

[6] YUAN W K, CAO J J, JIANG Z R, et al. Can large language models grasp legal theories? Enhance legal reasoning with insights

from multi-agent collaboration[EB/OL]. (2024-10-03)[2024-10-30]. https://arxiv.org/abs/2410.02507.

[7] HAN X,ZHANG Z Y,DING N,et al. Pre-trained models:past,present and future[J]. AI Open,2021,2:225-250.

[8] CHUNG H W,HOU L,LONGPRE S,et al. Scaling instruction-finetuned language models[EB/OL]. (2022-12-06)[2024-10-30]. https://arxiv.org/abs/2210.11416.

[9] WU S J,IRSOY O,LU S,et al. BloombergGPT:a large language model for finance[EB/OL]. (2023-12-21)[2024-10-30]. https://arxiv.org/abs/2303.17564.

[10] YANG X,CHEN A K,POURNEJATIAN N,et al. GatorTron:a large clinical language model to unlock patient information from unstructured electronic health records[EB/OL]. (202-12-21)[2024-10-30]. https://arxiv.org/abs/2203.03540.

[11] LÜ K,YANG Y Q,LIU T X,et al. Full parameter fine-tuning for large language models with limited resources[EB/OL]. (2024-06-06)[2024-10-30]. https://arxiv.org/abs/2306.09782.

[12] HU E J,SHEN Y L,WALLIS P,et al. LoRA:low-rank adaptation of large language models[EB/OL]. (2021-10-16)[2024-10-30]. https://arxiv.org/abs/2106.09685.

[13] HE J X,ZHOU C T,MA X Z,et al. Towards a unified view of parameter-efficient transfer learning[EB/OL]. (2022-02-02)[2024-10-30]. https://arxiv.org/abs/2110.04366.

[14] LI D C,MA Y Z,WANG N Z,et al. MixLoRA:enhancing large language models fine-tuning with LoRA-based mixture of experts[EB/OL]. (2024-07-20)[2024-10-30]. https://arxiv.org/abs/2404.15159.

[15] WU X,HUANG S H,WEI F R. Mixture of LoRA experts[EB/OL]. (2024-04-21)[2024-10-30]. https://arxiv.org/abs/2404.13628.

[16] GAO C Y,CHEN K Z,RAO J M,et al. Higher layers need more LoRA experts[EB/OL]. (2024-02-13)[2024-10-30]. https://arxiv.org/abs/2402.08562.

[17] YU L,YU B W,YU H Y,et al. Language models are super mario:absorbing abilities from homologous models as a free lunch[EB/OL]. (2024-06-13)[2024-10-30]. https://arxiv.org/pdf/2311.03099.

[18] HOULSBY N,GIURGIU A,JASTRZEBSKI S,et al. Parameter-efficient transfer learning for NLP[C]//CHAUDHURI K,SALAKHUTDINOV R. Proceedings of the 36th International Conference on Machine Learning,June 9-15,2019,Long Beach,California,USA. San Diego:PMLR,2019,97:2790-2799.

[19] ZHANG Q R,CHEN M S,BUKHARIN A,et al. AdaLoRA:adaptive budget allocation for parameter-efficient fine-tuning[EB/OL]. (2023-12-20)[2024-10-30]. https://arxiv.org/abs/2303.10512.

[20] LESTER B,AL-RFOU R,CONSTANT N. The power of scale for parameter-efficient prompt tuning[EB/OL]. (2021-09-02)[2024-10-30]. https://arxiv.org/abs/2104.08691.

[21] LI X L,LIANG P. Prefix-tuning:optimizing continuous prompts for generation[EB/OL]. (2021-01-01)[2024-10-30]. https://arxiv.org/abs/2101.00190.

[22] WORTSMAN M,ILHARCO G,GADRE S Y,et al. Model soups:averaging weights of multiple fine-tuned models improves accuracy without increasing inference time[C]//CHAUDHURI K,JEGELKA S,SONG L,et al. Proceedings of the 39th International Conference on Machine Learning,July 17-23,2022,Baltimore,Maryland,USA. San Diego:PMLR,2022,162:23965-23998.

[23] SHOEMAKE K. Animating rotation with quaternion curves[C]//COLE P,HEILMAN R,BARSKY B A. SIGGRAPH'85:Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques,July 22-26,1985. New York:Association for Computing Machinery,1985,19(3):245-254.

[24] YADAV P,TAM D,CHOSHEN L,et al. TIES-merging:Resolving interference when merging models[C]//OH A,NAUMANN T,GLOBERSON A,et al. Advances in Neural Information Processing Systems 36,San Diego:NeurIPS,2024,36:1-23.

[25] KIM D,PARK C,KIM S,et al. SOLAR 10.7B:scaling large language models with simple yet effective depth up-scaling[EB/OL]. (2024-04-04)[2024-10-30]. https://arxiv.org/abs/2312.15166.

[26] HUANG C S,LIU Q,LIN B Y C,et al. LoraHub:efficient cross-task generalization via dynamic LoRA composition[EB/OL]. (2024-08-19)[2024-10-30]. https://arxiv.org/abs/2307.13269.

[27] DING Y F,LIU J W,WEI Y X,et al. XFT:unlocking the power of code instruction tuning by simply merging upcycled mixture-of-experts[EB/OL]. (2024-06-06)[2024-10-30]. https://arxiv.org/abs/2404.15247.

[28] SUN T X,SHAO Y F,QIAN H,et al. Black-box tuning for language-model-as-a-service[C]//CHAUDHURI K,JEGELKA S,SONG L,et al. Proceedings of the 39th International Conference on Machine Learning,July 17-23,2022,Baltimore,Maryland,

USA. San Diego:PMLR,2022,162:20841-20855.
[29] 公茂果,高原,王炯乾,等. 基于进化策略的自适应联邦学习算法[J]. 中国科学:信息科学,2023,53(3):437-453.
GONG M G,GAO Y,WANG J Q,et al. Adaptive federated learning algorithm based on evolution strategies[J]. Scientia Sinica: Informationis,2023,53(3):437-453.
[30] HANSEN N,OSTERMEIER A. Adapting arbitrary normal mutation distributions in evolution strategies:the covariance matrix adaptation[C]//1996 IEEE International Conference on Evolutionary Computation,May 20-22,1996,Symposion & Toyoda Auditorium,Nagoya University,Japan. Piscataway:IEEE,1996:312-317.
[31] YANG A,YANG B S,HUI B Y,et al. Qwen2 technical report[EB/OL]. (2024-09-10)[2024-10-30]. https://arxiv.org/abs/2407.10671.
[32] SOCHER R,PERELYGIN A,WU J Y,et al. Recursive deep models for semantic compositionality over a sentiment treebank [C]//2013 Conference on Empirical Methods in Natural Language Processing,Seattle,Washington,USA,October 18-21,2013. Kerrville:Association for Computational Linguistics,2013:1631-1642.
[33] WU Z Y,WANG Y X,YE J C,et al. Self-adaptive in-context learning:an information compression perspective for in-context example selection and ordering[EB/OL]. (2023-10-16)[2024-10-30]. https://arxiv.org/abs/2212.10375.
[34] DING N,LV X T,WANG Q S,et al. Sparse low-rank adaptation of pre-trained language models[EB/OL]. (2023-11-20)[2024-10-30]. https://arxiv.org/abs/2311.11696.

## Operations Research and Cybernetics

# A New Algorithm for Large Language Models in Multitasking Scenarios

WANG Bowen[1], WU Zhiyou[1], ZHENG Xianda[2], GAO Huan[1]
(1. School of Mathematical Science, Chongqing Normal University, Chongqing 401331, China;
2. School of Computer Science, the University of Auckland, Auckland 1010, New Zealand)

**Abstract**: Large language models (LLMs) have shown excellent performance in most natural language processing tasks. However, direct application of general-purpose LLMs often does not meet the application needs of a specific domain. To solve this problem, it is often necessary to customize the model by training the model from scratch or fine-tuning the generic model. *De novo* training can achieve a high degree of customization, ensure matching with requirements and protect data privacy, but there are problems of high cost and technical difficulty, so the existing methods mostly improve the model performance by fine-tuning the general model, but the full-parameter fine-tuning will face the challenge of GPU memory limitation, although the existing parameter efficient fine-tuning technology can alleviate the memory limit, but the technology is difficult to maintain performance in multiple tasks at the same time, and catastrophic forgetting may also occur in the process of continuous fine-tuning. In order to solve this problem, a new method that can maintain the performance of multiple domains and alleviate the catastrophic forgetting phenomenon is proposed, namely the layer-by-layer adaptive weighted merger (LAEM) algorithm. This method is carried out in the form of LoRA module merger: firstly, the fine-tuned LoRA modules in a variety of specific tasks are deredundant. Secondly, by introducing the idea of sharing LoRA modules, and using the layer-by-layer adaptive weighted average method, the LoRA modules corresponding to different tasks after redundancy are merged with the shared modules, and the LAEM algorithm can flexibly set the weights according to the specific performance of different layers within the model and the contribution to the final result, so as to more accurately integrate the advantages of multiple models, fully release the potential of the model in each layer, and achieve better overall performance. Experimental results show that the LAEM algorithm not only enables the model to have a variety of capabilities, but also alleviates the phenomenon of catastrophic forgetting to a certain extent, and avoids the problem that the traditional method ignores the difference of features between layers when merging models.

**Keywords**: large language models; efficient fine-tuning of parameters; model merging

(责任编辑 黄 颖)