AdaLoRA 技术复现报告

摘要

本实验深入探讨了预训练语言模型（PLMs）在自然语言处理（NLP）任务中的优化策略，特别关注于提升模型的参数效率和性能。传统的全参数微调方法虽然能够提升模型性能，但同时也带来了巨大的内存消耗和计算成本。为了解决这一问题，研究者们提出了两种主要的研究方向：一是在 PLMs 中添加小型神经模块并仅对这些模块进行微调，二是以参数高效的方式模拟预训练权重的增量更新，即 LoRA 方法，它通过将增量矩阵参数化为两个小矩阵的乘积来实现低秩分解，从而减少了训练开销。然而，LoRA 方法在预设增量矩阵的秩时存在局限性，未能充分考虑不同模块和层的权重矩阵在微调过程中的重要性差异。

为了克服这一局限性，有论文提出了 AdaLoRA 方法，该方法包含两个关键组件：基于 SVD 的适应和重要性感知的秩分配。AdaLoRA 通过奇异值分解来参数化增量矩阵，并根据新设计的重要性度量来修剪冗余奇异值。这种方法不仅减少了计算开销，而且通过迭代修剪和全局预算调度器，实现了对参数预算的精细控制，从而在保持或提升 PLMs 性能的同时，显著降低了内存消耗。

实验结果表明，AdaLoRA 在多个 NLP 任务上均展现出了优于 LoRA 的性能，尤其是在 MNLI、SST-2、CoLA 和 QNLI 等任务上。此外，AdaLoRA 在训练初期就能快速降低损失值，尽管达到稳定状态所需的训练轮次可能多于 LoRA，但这并不影响其在参数效率和模型性能之间取得的良好平衡。

总的来说，AdaLoRA 方法为预训练语言模型的微调提供了一种新的视角，通过参数化的低秩适应和重要性感知的秩分配，有效地提升了模型的实用性和效率，为未来的 NLP 任务优化提供了有价值的参考。

关键词：大语言模型；参数高效微调；LoRA

1 引言

预训练语言模型（PLMs）在多种自然语言处理任务中表现出色。最常见的方法是对所有参数进行微调（全参数微调）。然而，预训练模型通常占用大量内存。例如，BERT 模型 [2] 包含高达 3 亿个参数；GPT-3 [1] 包含高达 1750 亿个参数。在构建基于这些预训练模型的 NLP 系统时，我们通常需要同时处理多个任务。面对大量下游任务，全参数微调需要每个任务维护一个大型模型的独立副本，这导致内存消耗非常昂贵。为解决这一问题，研究人员提出了两种主要研究方向来减少微调参数，同时保持甚至提高 PLMs 的性能。一种研究方向是向 PLMs 添加小型神经模块，并仅对这些模块进行微调——基础模型保持冻结并在任务间共享。这样，只引入和更新少量特定于任务的参数，大大提高了大型模型的实用性。另一种研

究方向提出以参数高效的方式模拟预训练权重的增量更新，而不修改模型架构。本文复现了一种新方法——AdaLoRA（自适应低秩适应）[4]，它在类似 LoRA 的微调过程中动态分配权重矩阵间的参数预算。具体来说，AdaLoRA 调整增量矩阵的秩以控制其预算。关键增量矩阵被赋予高秩，以便捕捉更细粒度和特定于任务的信息。较不重要的矩阵则被修剪为低秩，以防止过拟合并节省计算预算。在矩阵近似的现有文献中，有一些方法可以控制矩阵的秩，其中大多数直接计算矩阵的奇异值分解（SVD），然后截断最小的奇异值。这样的操作可以显式地操纵秩，更重要的是，最小化结果矩阵与原始矩阵之间的差异。此外，AdaLoRA 还提出了一个全局预算调度器以促进训练。具体来说，它从一个略高于最终预算的初始参数预算开始，然后逐渐减少，直到匹配目标。这样的调度器可以提高训练稳定性和模型性能。

2 相关工作

此部分对课题内容相关的工作进行简要的分类概括与描述，二级标题中的内容为示意，可按照行文内容进行增删与更改，若二级标题无法对描述内容进行概括，可自行增加三级标题，后面内容同样如此，引文的 bib 文件统一粘贴到 **refs.bib** 中并采用如下引用方式。

2.1 基于 transformer 的语言模型

Transformer 模型是一种深度学习架构，广泛应用于自然语言处理领域。一个典型的 Transformer 模型由 L 个堆叠的块组成，每个块包含两个子模块：多头注意力（MHA）和全连接前馈网络（FFN）。给定输入序列 $X \in R^{(n\times d)}$，其中 n 是序列长度，d 是特征维度，MHA 在 h 个头上并行执行注意力函数：

$$MHA(X) = Concat(head_1, ..., head_h)W_o$$

其中，Concat 表示将所有头的输出拼接在一起，$W_o \in R^{(d\times d)}$ 是输出投影矩阵。对于第 i 个头，$head_i$ 的计算如下：

$$head_i = Softmax(XW_{q_i}(XW_{k_i})^T/\sqrt{d_h})XW_{v_i}$$

这里，$W_{q_i}, W_{K_i}, W_{v_i} \in R^{(d\times d_h)}$ 分别是第 i 个头的查询（query）、键（key）和值（value）的投影矩阵，d_h 通常是 d/h，即每个头的特征维度。

FFN 模块由两个线性变换组成，中间有一个 ReLU 激活函数：

$$FFN(X) = ReLU(XW_{f1} + b_1)W_{f2} + b_2$$

其中，$W_{f1} \in R^{(d\times d_m)}$ 和 $W_{f2} \in R^{(d_m\times d)}$ 是两个线性变换的权重矩阵，b_1 和 b_2 是偏置项，d_m 是中间维度，通常大于 d。

最后，Transformer 模型使用残差连接（residual connection）和层归一化（layer normalization）来提高训练的稳定性和效果。残差连接将输入 X 直接加到每个子模块的输出上，然后应用层归一化。这种设计允许模型在处理非常深的网络时仍然保持有效，因为它缓解了梯度消失和梯度爆炸的问题。

2.2 参数高效微调 PEFT

2.2.1 添加额外的模块

这种方法向预训练语言模型（PLMs）中添加小型神经模块，并仅对这些模块进行微调以适应每个任务——基础模型保持冻结并在任务间共享。这样，只引入和更新少量特定于任务的参数，大大提高了大型模型的实用性。例如，适配器微调在基础模型的层之间插入称为适配器的小型神经模块。前缀微调和提示微调则在基础模型的输入或隐藏层附加额外的可训练前缀标记。这些方法在只更新不到 1% 的原始模型参数的情况下，已显示出与全参数微调相当的性能，显著降低了内存消耗.

2.2.2 LoRA

Hu 等人 [3] 提出了一种名为 LoRA 的方法，通过将增量矩阵 参数化为两个较小矩阵的乘积来实现低秩分解：

$$W = W^{(0)} + \Delta = W^{(0)} + BA$$

，其中 $W^{(0)}, \Delta \in R^{(d_1 \times d_2)}$，$A \in R^{(r \times d_2)}$，$B \in R^{(d_1 \times r)}$，且 r 远小于 $d1$ 和 $d2$。在微调过程中，只更新 A 和 B。例如，当 $d1 = d2 = 1024$ 时，可以选择 $r = 8$。LoRA 在只增加不到 0.5% 的可训练参数的情况下，可以将训练开销减少多达 70%。然而，与全参数微调相比,LoRA 仍然存在局限性，因为它预设了每个增量矩阵 的秩 r 相同，忽略了在微调预训练模型时，不同模块和层的权重矩阵的重要性差异。

3 本文方法

我们的方法包含两个重要组成部分：(i) 基于 SVD 的适应，将增量矩阵表示为奇异值分解的形式；(ii) 重要性感知的秩分配，根据新设计的重要性度量修剪冗余奇异值。

3.1 基于 SVD 的适应

我们提出将预训练权重矩阵的增量更新参数化为奇异值分解的形式：

$$W = W^{(0)} + \quad = W^{(0)} + P\Lambda Q$$

，其中 $P \in R^{(d_1 \times r)}$ 和 $Q \in R^{(r \times d_2)}$ 表示 Δ 的左/右奇异向量，对角矩阵 $\in R^{(r \times r)}$ 包含奇异值 $\lambda_i \leq i \leq r$，且 r 远小于 $min(d1, d2)$。我们进一步将 $G_i = \{P_{*i,}, Q_{i*}\}$ 表示为包含第 i 个奇异值和向量的三元组。在实践中，由于 Λ 是对角矩阵，我们只需将其保存为 R^r 中的向量。Λ 初始化为零，而 P 和 Q 采用随机高斯初始化，以确保在训练开始时 $\Delta = 0$。为了强制 P 和 Q 的正交性，即 $P^T P = QQ^T = I$，我们使用以下正则化项：

$$R(P, Q) = |P^T P - I|_F^2 + |QQ^T - I|_F^2$$

.

在我们的方法中，Λ 在每次梯度下降步骤后迭代修剪以调整秩。与直接对每个 Δ 进行 SVD 计算相比，我们的参数化避免了密集的 SVD 计算，大大减少了计算开销。

3.2 重要性感知的秩分配

我们将基于 SVD 的适应应用于每个 Transformer 层的权重矩阵，包括 W_q, W_k, W_v, W_{f1} 和 W_{f2}。为了控制预算，我们在训练过程中根据重要性分数迭代修剪奇异值。我们使用 k 来索引增量矩阵，即 $\Delta k = P_k \Lambda_k Q_k$，其中 n 是适应的权重矩阵的数量。我们用 $G_{k,i} = P_{k,*i}, {}_{k,i}, Q_{k,i*}$ 表示 Δk 的第 i 个三元组，并用 $S_{k,i}$ 表示其重要性分数。

我们进一步定义参数集 $P = \{P_k\}_{k=1}^n, E = \{\Lambda_k\}_{k=1}^n, Q = \{Q_k\}_{k=1}^n$ 和训练成本为 $C(P, E, Q)$。在第 t 步，我们首先进行随机梯度步骤来更新 $P_k^{(t)}$, $\Lambda_k^{(t)}$ 和 $Q_k^{(t)}$。具体来说，对于 $\Lambda_k^{(t)}$，我们有：

$$\Lambda_k^{(t+1)} = \mathcal{T}(_k^{(t)}, S_k^{(t)})$$

,

其中 $\mathcal{T}(_k^{(t)}, S_k^{(t)})_{ii} = \Lambda_{k,ii}^{(t)}$ 如果 $S(t)k, i$ 在 $S^{(t)}$ 的前 $b^{(t)}$ 中，否则为 0。这里 $b^{(t)}$ 是第 t 步剩余奇异值的预算。

我们提出了一种新的重要性度量标准，考虑了三元组 $G_{k,i}$ 中的奇异值和向量：

$$S_{k,i} = s(_{k,i}) + \frac{1}{d_1} \sum_{j=1}^{d1} s(P_{k,ji}) + \frac{1}{d_2} \sum_{j=1}^{d2} s(Q_{k,ij})$$

,

其中 $s(\cdot)$ 是单个条目的特定重要性函数。我们可以采用敏感度作为 $s(\cdot)$，定义为梯度-权重乘积的幅度：

$$I(w_{ij}) = |wij \ w_{ij} L|$$

,

这种方法在低预算设置下表现尤为突出。

4 复现细节

4.1 与已有开源代码对比

此部分为必填内容。如果没有参考任何相关源代码，请在此明确申明。如果复现过程中引用参考了任何其他人发布的代码，请列出所有引用代码并详细描述使用情况。同时应在此部分突出你自己的工作，包括创新增量、显著改进或者新功能等，应该有足够差异和优势来证明你的工作量与技术贡献。在本实验中，我们基于论文中提供的开源代码进行了复现和改进。具体来说，我们成功地复现了原代码中的 AdaLoRA 层部分，并对其进行了优化。通过去除与本实验无关的冗余判断部分，我们不仅使代码更加简洁，还显著提高了代码的运行速度。对于其他功能模块，如基于重要性感知的秩分配，我们保留了原代码的实现方法。这部分功能的进一步优化和改进将是未来工作的一个重点方向。

4.2 实验环境搭建

系统为 Ubuntu20.0.2，使用 1 张 4090 24G 显卡，cuda 版本 12.2，python 环境为 3.7，pytorch 版本 1.9.1，训练模型为 DeBERTaV3-base，数据集为 GLUE。

4.3 创新点

本实验在完成了 AdaLoRA 代码复现后进行了一系列的测试,除了像论文中聚焦于 AdaLoRA 于 LoRA 等各种微调技术的性能对比，本实验还额外测试了其他的指标，如使用 AdaLoRA 技术的模型的 loss 曲线收敛速度。

5 实验结果分析

5.1 性能表现

Method	**MNLI**	**SST-2**	**CoLA**	**QQP**	**QNLI**	**RTE**	**MRPC**	**STS-B**	**All**
	m/mm	acc	acc	Acc/F1	Acc	Acc	Acc	Corr	Avc.
$\text{LoRA}_{\text{r=8}}$	90.65/90.69	94.95	69.82	91.99/89.38	93.87	85.20	89.95	91.60	88.34
AdaLoRA	90.97/90.79	96.10	71.45	92.23/89.74	94.55	88.09	90.69	91.84	89.31

表 1. LoRA 和 AdaLoRA 技术的性能表现对比

从1中，我们可以观察到 AdaLoRA 方法在多个自然语言处理任务上相较于 LoRA 方法有轻微的性能提升。在 MNLI 任务上，AdaLoRA 在匹配和不匹配的子任务中都展现出了略高的平均准确率，这表明它在处理自然语言推理问题时可能更有效。在 SST-2 任务，即情感分析任务中，AdaLoRA 的准确率显著高于 LoRA，这可能意味着 AdaLoRA 在捕捉文本情感方面更为敏感和准确。

在 CoLA 任务，即语法正确性评估上，AdaLoRA 同样展现出了比 LoRA 更高的准确率，这可能表明它在理解语言的语法结构方面更为出色。在 QQP 任务，即问答对的语义相似性评估中，AdaLoRA 在准确率和 F1 分数上都略胜一筹，这可能意味着它在识别问题间的语义关系方面更为精确。

在 QNLI 任务，即问题自然语言推理中，AdaLoRA 的准确率也略高于 LoRA，这进一步证实了 AdaLoRA 在自然语言推理任务上的优势。在 RTE 任务，即识别文本蕴含关系中，AdaLoRA 的表现同样优于 LoRA，这可能表明它在判断文本间的逻辑关系上更为准确。

在 MRPC 任务，即识别文本对的语义等价性中，AdaLoRA 的准确率也略高，这可能意味着它在处理语义相似性问题时更为有效。最后，在 STS-B 任务，即评估文本对的语义相关性中，AdaLoRA 的相关性分数也略高于 LoRA，这可能表明它在量化文本间的语义相似度方面更为精确。

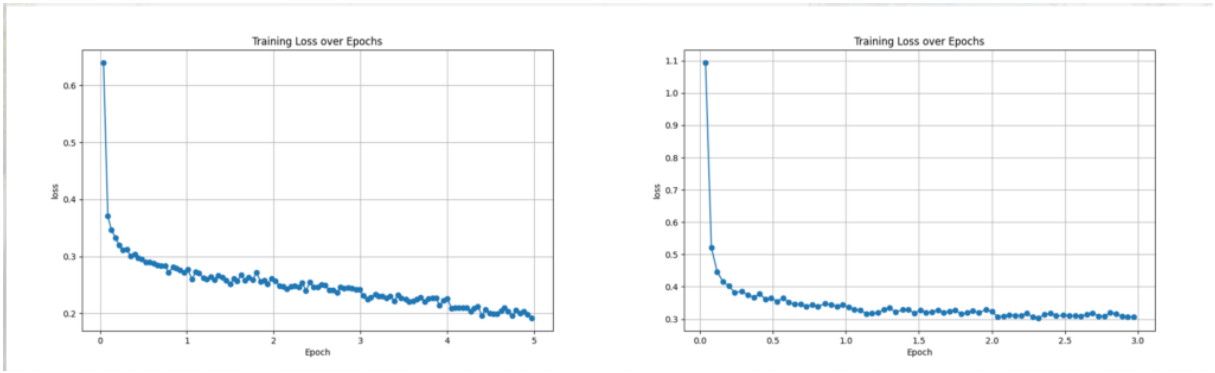

图 1. 不同微调技术使用 QQP 数据集对模型进行训练时收敛速度对比，左图为 AdaLoRA 损失曲线，右图为 LoRA 损失曲线

综合所有任务的平均准确率，AdaLoRA 以 89.31% 的平均准确率略高于 LoRA 的 88.34%，这表明 AdaLoRA 在这些自然语言处理任务上的整体性能略优于 LoRA。这些结果可能意味着 AdaLoRA 在参数效率和模型适应性方面有所改进，从而在不同的自然语言处理任务中都能保持较好的性能。

5.2 loss 曲线收敛速度

从2中可以清晰地观察到，在训练的初始阶段，AdaLoRA 方法的损失值（loss）显著低于 LoRA 方法，这表明 AdaLoRA 在训练的早期阶段就能够更快地收敛并展现出较好的学习效果。随着训练轮次的增加，AdaLoRA 的损失值依然保持在较低水平，这进一步证实了其在优化过程中的稳定性和优越性。

然而，值得注意的是，AdaLoRA 达到稳定损失值所需的训练轮次显著多于 LoRA。这一现象可能源于 AdaLoRA 在训练过程中涉及的参数数量相对较少。由于参数数量的减少，AdaLoRA 在训练初期能够快速降低损失值，但同时也意味着它需要更多的训练轮次来充分挖掘和利用有限的参数信息，以达到与 LoRA 相当的稳定状态。

这种训练参数数量的差异可能导致 AdaLoRA 在训练过程中需要更长的时间来调整和优化模型，以确保所有参数都能够有效地参与到学习过程中，从而实现模型性能的最大化。尽管如此，AdaLoRA 在训练效率和最终性能上的优势仍然表明，它是一种在参数效率和模型表现之间取得良好平衡的方法。

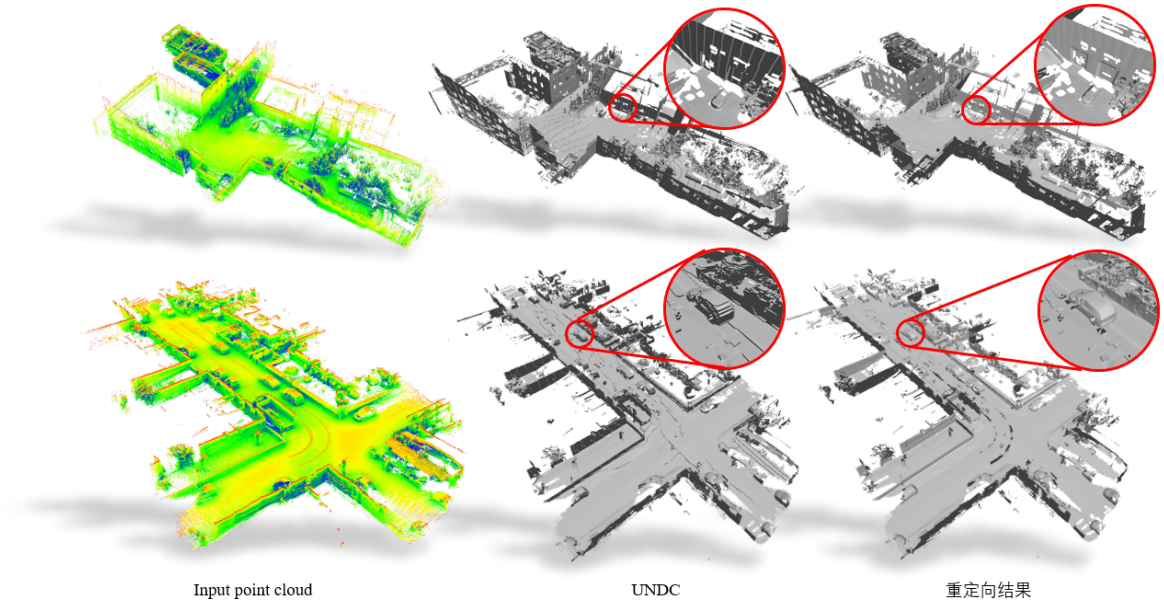

图 2. 实验结果示意

6 总结与展望

本实验首先深入探讨了预训练语言模型在自然语言处理任务中的优化策略，尤其是以 LoRA 为代表的参数高效微调方法，指出现有技术存在的问题，并引出基于 LoRA 的改进方法——AdaLoRA。该方法包含两个关键组件：基于 SVD 的适应和重要性感知的秩分配。AdaLoRA 通过奇异值分解来参数化增量矩阵，并根据新设计的重要性度量来修剪冗余奇异值。这种方法不仅减少了计算开销，而且通过迭代修剪和全局预算调度器，实现了对参数预算的精细控制，从而在保持或提升 PLMs 性能的同时，显著降低了内存消耗。

本实验成功复现并对 AdaLoRA 进行一系列测试，实验结果表明，AdaLoRA 在多个 NLP 任务上均展现出了优于 LoRA 的性能，尤其是在 MNLI、SST-2、CoLA 和 QNLI 等任务上。此外，AdaLoRA 在训练初期就能快速降低损失值，尽管达到稳定状态所需的训练轮次可能多于 LoRA，但这并不影响其在参数效率和模型性能之间取得的良好平衡。

在未来的研究工作中，我们计划深入探索 AdaLoRA 方法，以期实现更快速地达到稳定状态的目标。这将涉及到对算法的进一步优化，可能包括对学习率的调整、训练策略的改进以及对模型结构的精细化调整。我们相信，通过这些努力，AdaLoRA 将能够在保持其参数效率优势的同时，缩短训练周期，更快地收敛到最优性能。

同时，我们也将持续对代码进行改进和优化。这不仅包括提高代码的运行效率，减少内存消耗，还包括增强代码的可读性和可维护性，使其更加健壮和易于使用。我们致力于通过这些改进，为研究社区提供一个更加高效、可靠的工具，以促进自然语言处理领域的发展。

此外，我们也将考虑将 AdaLoRA 应用于更广泛的任务和数据集，以验证其通用性和适应性。通过在多样化的应用场景中测试和评估 AdaLoRA，我们希望能够进一步揭示其潜力，并为未来的研究和实践提供更丰富的洞见。我们期待 AdaLoRA 能够在未来的自然语言处理任务中发挥更大的作用，为该领域带来新的突破和进步。

参考文献

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[4] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023.