大语言模型的高效参数微调技术

UNIkeEN

2024 年 4 月 6 日

摘要

由于较高的参数量，对大语言模型（LLMs）进行微调面临挑战。近年来，研究者提出了多种高效参数微调（PEFT）技术，以减少微调过程所需的计算与存储资源。本文基于 LoRA[1] 和 LLM-Adapters[2] 两篇论文，探讨两种主流的 PEFT 方法——低秩适应（LoRA）和基于适配器的方法——的原理、实验与效果对比，并介绍了实际微调应用中的经验观察。

关键词 大语言模型、高效参数微调、低秩适应、适配器

1 概述

1.1 应用背景

大语言模型（LLMs）在各种自然语言处理（NLP）任务和多模态任务中展现出惊人的性能，但对于具有一定专业性、包含私有数据的特定领域，需要将预训练的大规模语言模型通过微调适应于下游应用（如机器翻译、景区旅游导引等）。微调通常指全模型微调（FFT），这会更新预训练模型的所有参数，即新模型包含的参数数量与原始模型相同。随着近几年大语言模型的快速发展，模型参数不断增大，达到了超过百亿的规模（如 GPT-3 175B[3]）。对如此规模的参数进行全模型微调，在计算、存储和数据传输等方面都面临挑战。

许多方案通过只微调部分参数而不是整个主干模型、或为新任务学习外部模块来缓解这一点。这类方法被称为参数高效微调（PEFT），其中比较有代表性的有：低秩方法（如 LoRA[1]）、基于适配器的方法（如 LLM-Adapters[2]）、基于层前缀的方法、基于提示的方法等。将这些 PEFT 模块集成到主干模型中，使不同的下游应用开发者得以既利用主干模型的能力，又节省计算与存储资源，同时仍能达到与 FFT 相似甚至更优越的性能。

2 LoRA

LoRA[1] 是 Low-Rank Adaptation（低秩微调）的简称，其主要的方法是为大模型注入低秩矩阵并更新其参数（见 2.1.1 节）。特别地，许多现有 LLM 如 BERT[4] 采用了 Transformer 架构 [5]，本文也提出了将 LoRA 方法应用到 Transformer 层的方法（见 2.1.2 节）。此方法相较于之前基于适配器的方法，避免了因添加额外模型层产生的推理延迟；相较于基于提示的方法，减轻了训练难度、避免了因预留提示序列挤占下游任务输入产生的性能影响。

2.1 方法介绍

2.1.1 低秩矩阵注入与更新

神经网络包含许多进行矩阵乘法的密集层，这些层中的权重矩阵通常是满秩的。先前工作 [6] 显示大语言模型具有较低的“内在维度”，在为特定任务适配时，即使随机投影到更小的子空间，仍能较有效地学习。

本文由此假设，微调过程中的参数更新也存在“低秩”（实验见 2.2 节），给定预训练权重矩阵 $W_0 \in \mathbb{R}^{d\times k}$，目标权重矩阵

$$W = W_0 + \Delta W = W_0 + BA \tag{1}$$

其中 $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$。在训练期间，原始权重 W_0 固定，仅更新 A 和 B。训练开始时使用随机高斯分布初始化 A、使用全零矩阵初始化 B。网络结构示意如 Fig. 1。

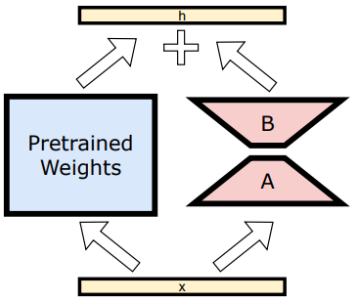


Fig. 1. LoRA 方法网络结构示意

设置秩 $r \ll \min(d, k)$，使这种方法大大减少了训练与存储时的参数量（对于不同的下游任务，只需要存储不同的 ΔW，便于快速切换），特别地，当 $r = k$ 时，LoRA 演变为全模型微调（FTT）。LoRA 的思想类似于残差网络中的残差连接，在原始权重外增加旁路降维后升维，模型的输入输出维度不变、推理过程不变（不增加延迟）。

2.1.2 在 Transformer 层应用 LoRA

LoRA 可以用于任意种类的权重矩阵。在现今模型尤其是 LLM 普遍运用的 Transformer 架构 [5] 中，自注意力模块包含四个权重矩阵 W_q, W_k, W_v, W_o，多层感知机（MLP）模块中包含两个。即使输出维度通常被分割为多个注意力头，本方法仍将 W_q（或 W_k、W_v）视为一个维度为 $d_{\text{model}} \times d_{\text{model}}$ 的单一矩阵，并应用上述方法。

为了简化和提高参数效率，本文提出的原始 LoRA 方法仅限于适应下游任务的注意力权重，冻结 MLP 模块（因此它们在下游任务中不会被训练）。

2.2 实验效果

2.2.1 计算资源、效率优化与准确度

作为一种高效参数微调方法，LoRA 的核心优化目标在于减少微调所需的时空资源。以用 Adam 优化器训练的大型 Transformer 模型——GPT-3 175B 模型为例：

实验显示，在空间效率方面，当 $r << d_{\text{model}}$ 时，LoRA 可以将 VRAM 使用量减少到全模型微调的 1/3（其不需要为冻结参数存储优化器状态），从 1.2TB 减少到 350GB，减少了所需的计算资源。当 $r = 4$ 且仅调整查询和值投影矩阵时，训练时的中间权重存储大约减少了 10,000 倍（从 350GB 减少到 35MB），避免了 I/O 瓶颈。

在时间效率方面，训练速度较全模型微调提升了 25%，这是因为 LoRA 不需要为绝大多数参数计算梯度。

节省时空资源的同时也需保证微调的质量。Fig. 2 是全微调与各种 PEFT 微调方案（如基于适配器的方案 [7]，即表中的 $\textbf{Adapter}^{\textbf{H}}$）在 GPT-3 175B 模型上的性能评估。下游任务使用 WikiSQL[8]（自然语言转换 SQL 语句）、MNLI[9]、SAMSum[10]（对话总结）并分别测量验证准确率与 ROUGE 指标（一种用于评估文摘与机器翻译的性能指标）。可以看出 LoRA 在保证较低的训练参数量时，在效果上均达到最佳，甚至在大部分情况下优于全模型微调。

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

Fig. 2. 不同微调方案在 GPT-3 175B 模型上的效果比较

文中也对 BERT 系列模型、GPT-2 模型开展了实验，结果均为 LoRA 参数量较低，

性能指标大部分占优，在此不再详述。

2.2.2 秩参数 r 的选择

LoRA 最核心的超参数即低秩参数 r 的选择，作者以 GPT-2 作为预训练模型、WikiSQL[8] 和 MNLI[9] 为微调任务，选择 Transformer 层的不同权重矩阵组合进行微调并评估验证准确率，以探究秩 r 对模型性能的影响。实验结果如 Fig. 3。

	Weight Type	$r=1$	$r=2$	$r=4$	$r=8$	$r=64$
WikiSQL(±0.5%)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI (±0.1%)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

Fig. 3. 不同秩 r 在 GPT-2 模型上的微调效果比较

即使是非常小的 r，LoRA 的性能已经非常具有竞争力，继续增加 r，验证准确率不再上升，代表不同随机种子下所学习子空间并没有被覆盖更多，实验结果支持了更新矩阵 ΔW 具有非常小的“内在秩”这一假设，即一个低秩的适应矩阵对于微调已然足够。

此外，本文还进行了更多的补充实验，证明 ΔW 与 W 具有很强的相关性，ΔW 只放大了 W 中拥有较小奇异值的方向且放大因子极大，这表明低秩适应矩阵可能显著放大了已经学习过但未曾强调的重要特征。

3 LLM-Adapters

基于适配器的方法是另一类 LLM 常用的高效参数微调（PEFT）方法，主旨是向基座预训练模型的已有子层添加新的模块并训练之。主要包含串行适配器（见 3.1.1 节）和并行适配器（见 3.1.2 节）。本文提出了一个易于使用的框架——LLM-Adapters[2]，向 LLM 集成了各类适配器模块，并对不同任务、不同预训练模型下适配器类型、放置位置和超参数对基于适配器方法的影响展开实验。

3.1 方法介绍

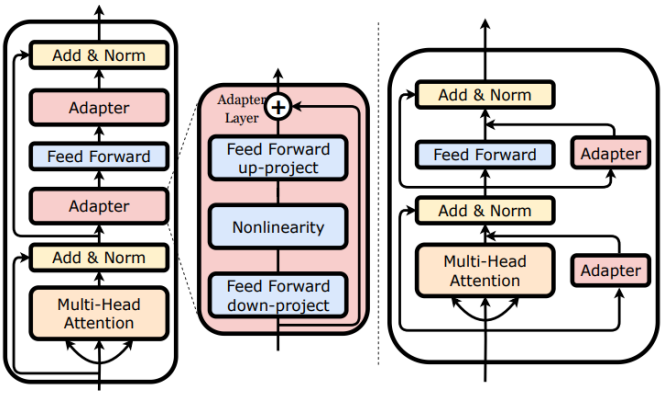


Fig. 4. 串行与并行适配器网络结构示意

3.1.1 串行适配器

串行适配器涉及在特定子层后以序列方式整合额外的可学习模块：如在 Transformer 模型 [5] 的注意力和 FFN 层之后整合全连接网络。同 LoRA 类似，这些可学习模块的计算复杂性通常小于原有子层（经历降维再升维的过程），原有层的参数在微调时被冻结。Compacter[11] 利用 Kronecker 积、低秩矩阵和跨层的参数共享来生成适配器权重。这种技术旨在减少与适配器相关的计算复杂性，同时保持它们的性能。

串行适配器可以如下公式化：

$$H_o \leftarrow H_o + f(H_o W_{\text{down}}) W_{\text{up}} \tag{2}$$

特定层的**输出** H_o 首先通过 $W_{\text{down}} \in \mathbb{R}^{d \times r}$ 向下投影到较低维度 r，然后通过 $W_{\text{up}} \in \mathbb{R}^{r \times d}$ 向上投影回原始维度 d。ΔW，f 是一个非线性函数。具体的网络结构如 Fig. 4 左侧。

3.1.2 并行适配器

并行适配器 [12] 旨在与骨干模型内的不同子层并行整合额外的可学习模块。其可以如下公式化：

$$Ho \leftarrow Ho + f(HiW_{\text{down}}) W_{\text{up}} \tag{3}$$

特定层的**输入** H_i 经过额外的旁路——包含升降维层、非线性函数与特定层——后与特定层原本的输出相加。具体的网络结构如 Fig. 4 右侧。

3.2 实验效果

3.2.1 适配器位置的选择

本文以 LLaMA-7B[13] 作为基座模型，评估数学推理任务背景下不同适配器放置位置的效果。对于串行适配器，分别放置在自注意力层后、MLP 层后和两者之后；对于并行适配器，分别整合到自注意力层、MLP 层和两者兼有。特别地，LoRA 原始论文[1] 仅对自注意力层权重引入低秩适应矩阵，本文也分组实验了将 LoRA 方法分别应用于自注意力层、MLP 层和两者兼有并比较其验证准确率。

以数学推理数据集为例，结果如 Fig. 5。

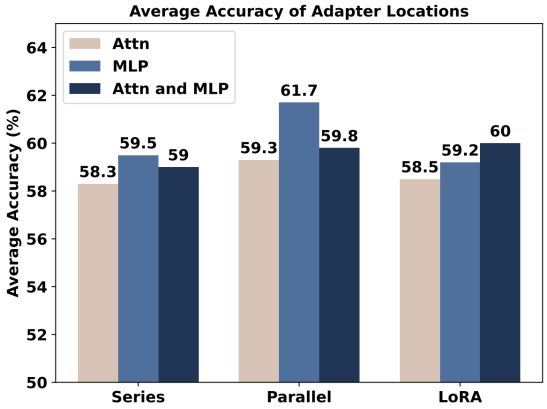
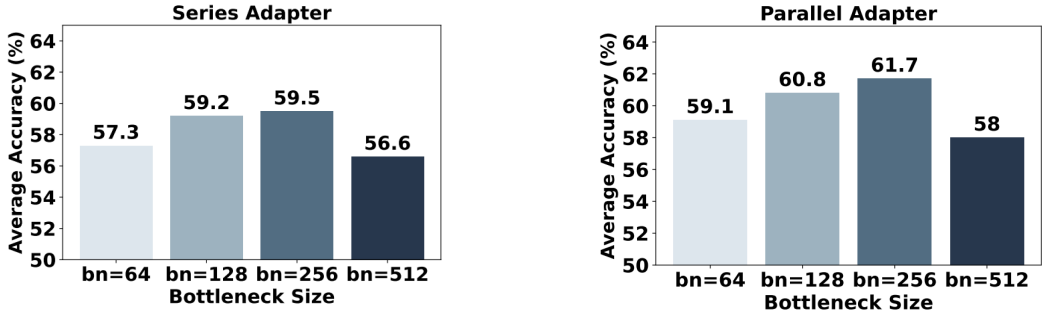


Fig. 5. 不同适配器位置在 LLaMA-7B 模型上的效果比较

可以观察到，对于串行适配器，最佳位置是在 MLP 层之后，平均准确率达到了 59.5%。对于并行适配器，当我们将其放置在 MLP 层内时，表现最佳，准确率达到 61.7%。关于 LoRA，我们需要同时将其插入多头注意力层和 MLP 层中，以达到最佳性能 60%。



(a) (b)

Fig. 6. 不同适配器瓶颈大小在 LLaMA-7B 模型上的效果比较

本文也同样对不同的适配器瓶颈大小进行了实验，结果如 Fig. 6。在 LLaMA-7B、数学推理数据集上，将瓶颈维度配置为 256，串行和并行适配器展示了最高水平的性能。

然而，当瓶颈大小增加到 512 时，串行和并行适配器的准确率都有所下降。

3.2.2 效果评估

本文的主要创新点在于提出高效、组件化的 PEFT 框架、建立私有数据和多种微调效果的量化评估标准，对已有的 PEFT 方法（串/并行适配器、LoRA 方法、基于前缀的方法等）开展广泛实验。

使用 3.2.1 节得到的最佳参数设置，在 LLaMA-7B 等参数量中等的 LLM 上开展实验。不同模型、不同下游任务下，适配器方法与 LoRA 的效果各有千秋，但并行适配器通常优于串行适配器，基于前缀的方法在所有任务上均劣于其余方法。考虑到推理延迟，存储与任务切换的便捷性，LoRA 在大多数情况下是更好的选择。

4 比较与思考

基于适配器的方法和 LoRA 方法都是大语言模型的常见微调方法。通过两篇文章的介绍，两类方法都能在相比于全模型微调显著减少训练参数量的情况下达到较好的性能。观察前文所述实验数据，在中型 LLM（如 LLaMA-7B）上，并行适配器略强于 LoRA 方法，强于串行适配器；在大型 LLM（如 GPT-3 175B）上，LoRA 在下游任务的验证准确度高于适配器方法。效果评估受评测标准、基座模型、具体下游任务不同而有所差异。但由于 LoRA 只需存储低秩矩阵、没有修改模型结构；而适配器无论串并行都增加了特定子层，引入了推理延迟，且不同适配器结构可能不同、切换任务时效率低于 LoRA，故我个人认为 LoRA 是更好的选择。

这与我之前在相关微调实践的观察类似，在 ChatSJTU 和另一个比赛项目中，分别对 LLaMA 2-7B 模型和 ChatGLM-6B 模型，在校园信息与“交我算”用户文档、中小学教材的数据上进行微调（问答对数据集由我整理，具体实现使用开源框架）。通过简单的 case study，我们发现，开源框架提供的默认配置下，适配器微调的效果差于 LoRA 方法。另外，我们还使用检索增强生成方法（RAG）与上两种 PEFT 微调方法作比较，观察得到：在进行小规模私有数据集的微调时，如不进行数据增广，微调效果一般；如进行数据增广（复制问答对数量，修改问法），容易过度微调，LLM 产生“幻觉”。而 case study 显示 RAG 方法效果强于微调方法。

此外，在本次论文阅读中，我发现低秩矩阵与并行适配器的概念容易产生混淆。两者看似都为原有层添加了“旁路”，而前者侧重于“权重”，其只包含两个低秩矩阵，前向传播过程不变，仅存储矩阵参数，不会引入推理延迟且存储成本低；后者侧重于“层”，其旁路上可包含多个层、旁路接入的原有层数量也不确定，并且通常会新增非线性激活函数；前向传播过程变化，会引入推理延迟且可变性强。

参考文献

[1] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[2] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. K.-W. Lee, "Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models," 2023.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[6] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," 2020.

[7] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," 2019.

[8] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," 2017.

[9] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," 2018.

[10] B. Gliwa, I. Mochol, M. Biesek, and A. Wawer, "Samsum corpus: A human-annotated dialogue dataset for abstractive summarization," in *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, Association for Computational Linguistics, 2019.

[11] R. K. Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," 2021.

[12] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, “Towards a unified view of parameter-efficient transfer learning,” 2022.

[13] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, “Llama: Open and efficient foundation language models,” 2023.