

精简大型语言模型而不损失其智能

Qingda (Michael) Mai

University of Waterloo
Department of Management Science and Engineering
(e-mail: michael.mai@uwaterloo.ca)

Abstract. 本文研究并验证了微调对大型语言模型性能的影响, 重点关注参数高效的方法 (LoRA 和 QLoRA) [1, 4, 11]。我们评估了模型在三个关键领域的的能力: (1) 常识推理 (HellaSwag [16]), (2) 数学推理 (GSM8K² [2]), 和 (3) 多领域知识 (MMLU-CS³ [9])。我们的研究表明: (1) 基于 LoRA 的方法在保持计算效率的同时有效地提高了特定任务的性能, 并且 (2) 性能在很大程度上依赖于微调数据集与基准任务之间的对齐。该研究为参数高效机制提供了理论见解, 并为开发者在资源有限的情况下实施高效的大型语言模型适应提供了实用指导。

关键词: Fine-tuning large language model, LoRA (Low-Rank Adaptation), Alignment

1 介绍

训练拥有数百亿参数的大型神经网络在计算上是非常昂贵且资源密集的。随着模型规模的增长, 训练它们所需的成本也在增加。诸如 LoRA [1, 11] 和 QLoRA [4] 等参数高效微调技术成为减少这些成本的有效解决方案, 同时还能保持甚至提升模型性能。

虽然本文没有研究 LoRA 的内部运作, 但我们利用其核心原则来检验在大型语言模型中仅更新一小部分参数是否能在不降低性能的情况下提高计算效率。

- 微调是否能够显著提升大型语言模型的性能? 如果是, 提升的是哪种类型的任务? 微调过程可以如何设计? (例如, 微调所用数据集的类型是否重要? 微调是否有可能保持其他大型语言模型的能力, 比如“常识推理”或“数学推理”?)

通过回答这些问题, 这项工作旨在使个体开发者和研究人员能够有效地利用有限的数据集微调较小规模的语言模型。

1.1 目标 & 文献综述

本研究的主要目标是:

- 解释 QLoRA (量化 + LoRA) 在数学/工程直觉上的概念, 以及它如何促进对大型语言模型的高效微调。

¹ HellaSwag: A dataset for measuring commonsense natural language inference with challenging sentence completion tasks.

² Grade School Math 8K (GSM8K): A dataset of 8.5K high-quality grade school math word problems requiring multi-step reasoning.

³ MMLU (Massive Multitask Language Understanding): A benchmark testing knowledge across 57 subjects. CS refers to the Computer Science subset.

⁴ Source: default precision of Stata output number, see <https://www.stata.com/manuals13/ddatatypes.pdf>

- 设计一个实验并评估参数高效微调对模型性能在常识推理（HellaSwag）、数学推理（GSM8K）和领域知识（MMLU）基准测试中的影响

1.1.1 关键概念

在我们了解大型语言模型微调之前，我们必须熟悉一些概念，比如：“精度” & “量化”，因为它们是实现减小大型语言模型尺寸的重要技术，并且在 GPU 上运行更快（加载数据所需时间更少）。

- 精度：指的是数值表示的位宽（例如，32 位浮点）。更高的精度保留了更多的信息，但也增加了内存使用和计算时间 [15]。最近的研究者已经克服了大型语言模型内存加载的瓶颈，并开发了各种优化器，这些优化器使用 8 位统计数据，同时保持使用 32 位优化器状态的性能水平 [3]。众所周知，不同类型的神经网络可以表示为不同的架构，这些架构的基本单元是权重。而这些权重是浮点数（即 Python 中的 float32）。考虑一下软件 Stata，输出数字的默认设置是 7 位精度⁴。它提供的输出位数越长，所需的数据类型大小/计算机内存越大。
- 量化：减少数值精度（例如，从 32 位减少到 4 位）的过程，以减少模型大小和加速推理，同时尽量减少准确性损失 [5]。模型大小如下：

$$\text{Model size} = \text{size}_{\text{datatype}} \times \text{num weights} \quad (1.1)$$

如公式 1.1 所示，模型大小取决于数据类型大小和权重的数量。因此，在这种情况下，人们提出了使用量化技术的想法，该技术只需要更少的内存和较低的精度，但仍保持模型性能。通常，这是一种通过降低精度来使模型大小更小而不影响模型性能的技术 [15]。这里的重点是，当加载大型语言模型时，使用 4 位量化几乎普遍是最佳选择 [5]。

- 低秩适应 (LoRA)：为了下游类似任务利用开源的预训练语言模型 (PLMs)（即 BERT 是一个基础模型，而微调 BERT 模型意味着使其适用于某些特定任务）现在是自然语言处理中的流行范式。最常见的方法是微调所有模型参数（完全微调）来使通用 PLMs 适应下游任务。[8] 然而，最近的研究提出了各种参数高效的迁移学习方法，这些方法仅微调必要的参数以获得强性能。一般有 4 种微调类型：(1) 迁移学习（微调下游任务的最后一层），(2) 适配器层（在预训练模型层内添加新的中间模块），(3) 前缀微调（提示工程），(4) LoRA（秩分解）。这些不同的方法将在方法部分中更详细地讨论。[10, 11, 12, 14] 考虑计算效率；我们希望尽可能多地减少计算密集型参数的数量。这里有一个 LoRA 的基本示例（是一种参数高效的方法）：假设你有一个权重矩阵大小为 1000×1000 的预训练语言模型。直接微调这个模型需要更新 W 中全部 1,000,000 个参数，这在计算上是昂贵的。LoRA 方法 LoRA 并不更新 W 中的所有参数，而是引入了两个低秩矩阵 A 和 B ：

- 设 A 的大小为 1000×10 （秩为 $r = 10$ ）。
- 令 B 的大小为 10×1000 （秩为 $r = 10$ ）。

现在，你只需要更新：

$$\text{Number of trainable parameters} = (1000 \times 10) + (10 \times 1000) = 20,000 \quad (1.2)$$

这显著减少了可训练参数的数量。

给定一个预训练的权重矩阵 $W \in \mathbb{R}^{d \times d}$ ，LoRA 学习：

- $A \in \mathbb{R}^{d \times r}$ 和 $B \in \mathbb{R}^{r \times d}$ 其中 $r \ll d$ （通常为 $r = 8$ ）
- 更新变为 $\Delta W = AB$ 而不是完整的 d^2 参数
- 前向传递计算：

$$y = (W + AB)x \quad (1.3)$$

QLoRA 通过将 W 量化到 4 位精度 [4] 来扩展这一点，在保持低秩适应性优势的同时进一步减少内存需求。

2 方法

2.1 大模型训练和微调的相关工作

有几种不同的微调技术：

迁移学习（微调最后一层用于下游任务）：传统的迁移学习方法是简单地冻结预训练模型中的所有参数和权重，然后在右侧添加一个新组件。所以在这种情况下，预训练模型的输出将成为新组件的输入（可能是一个层或新定义的用于特定下游任务的神经网络架构）。这种方法的缺点是我们的下游模型只能访问预训练模型的“输出嵌入”，而在进行反向传播时无法调整预训练模型的权重。[13]

适配器层（在预训练模型的层中添加新的中间模块）：谷歌的研究论文提出，微调可以在原始预训练模型内部进行，并且可以在层之间插入新模块。然而，缺点是这个新添加的层会增加模型推理过程中的延迟，并且整体的计算效率会降低。[10]

前缀微调（提示工程）：另一种微调解决方案是修改预训练模型每层的前缀，这类似于进行特征工程，简单地优化每层的输入。[12]

LoRA（秩分解）：这是最常用的方法。它对更新的权重矩阵进行秩分解。采用这种优化方向，该方法实际上非常适合于与 transformer 相关的架构。这些注意力机制总是需要权重矩阵来计算嵌入。秩分解总是将应用于这些更新后的权重矩阵。[11,14]

在本文中，我们将重点使用最常见的一种：QLoRA，这是一种量化和“低阶自适应”的结合。

LoRA 的全称是“低秩适应”，这并不是一个新的概念，早已广泛应用于许多传统的机器学习领域：例如推荐领域的“稀疏矩阵问题”和图像压缩中的“感知压缩”。Facebook 研究团队 [1] 提供了一个将这项技术应用于大型语言模型进行微调的例子。为了更好地理解，“秩”是一个数学概念，表示独立行/列的最小数量。这是在处理来自各个领域应用的矩阵时的重要数学特征。

基本上，工程师想要使用一个更小的矩阵来近似未确定的矩阵，从而降低计算复杂度。

本文并不是研究 LoRA 的工作原理，而是通过利用这种方法的理念，我们希望看看这是否真的有助于“瘦身”大型语言模型，同时提高稀疏矩阵的计算效率；下游任务不需要调整整个参数空间中的所有参数以适应特定模型。相反，它可以转化为一个更小的权重集合来实现性能良好的模型。他们发现模型越大，固有维度 [1] 越低。

在传统的微调中，层的权重矩阵 W 是直接更新的：

$$W_{\text{new}} = W_{\text{old}} + \Delta W$$

，其中 ΔW 是全秩的权重更新。

LoRA 使用两个低秩矩阵 A 和 B 来逼近权重更新 ΔW ：

$$\Delta W = A \cdot B$$

，其中：

- $A \in \mathbb{R}^{d \times r}$ 是一个秩为 r 的低秩矩阵。
- $B \in \mathbb{R}^{r \times d}$ 是另一种秩为 r 的低秩矩阵。
- $r \ll d$ ，意味着秩 r 远小于原始维度 d 。

2.1.1 使用 LoRA 的最终权重更新

更新后的权重矩阵 W_{new} 计算为：

$$W_{\text{new}} = W_{\text{old}} + A \cdot B$$

。在这里，只有 A 和 B 是可训练的，而 W_{old} 保持不变。

2.1.2 带有 LoRA 的前向传播

在前向传播过程中，输出 y 计算为：

$$y = (W_{\text{old}} + A \cdot B) \cdot x$$

其中 x 是层的输入。结合量化优势，这就是我们如何利用 QLoRA 来微调我们的大型语言模型的方法。[4]

2.2 Transformer 架构 & 参数高效微调

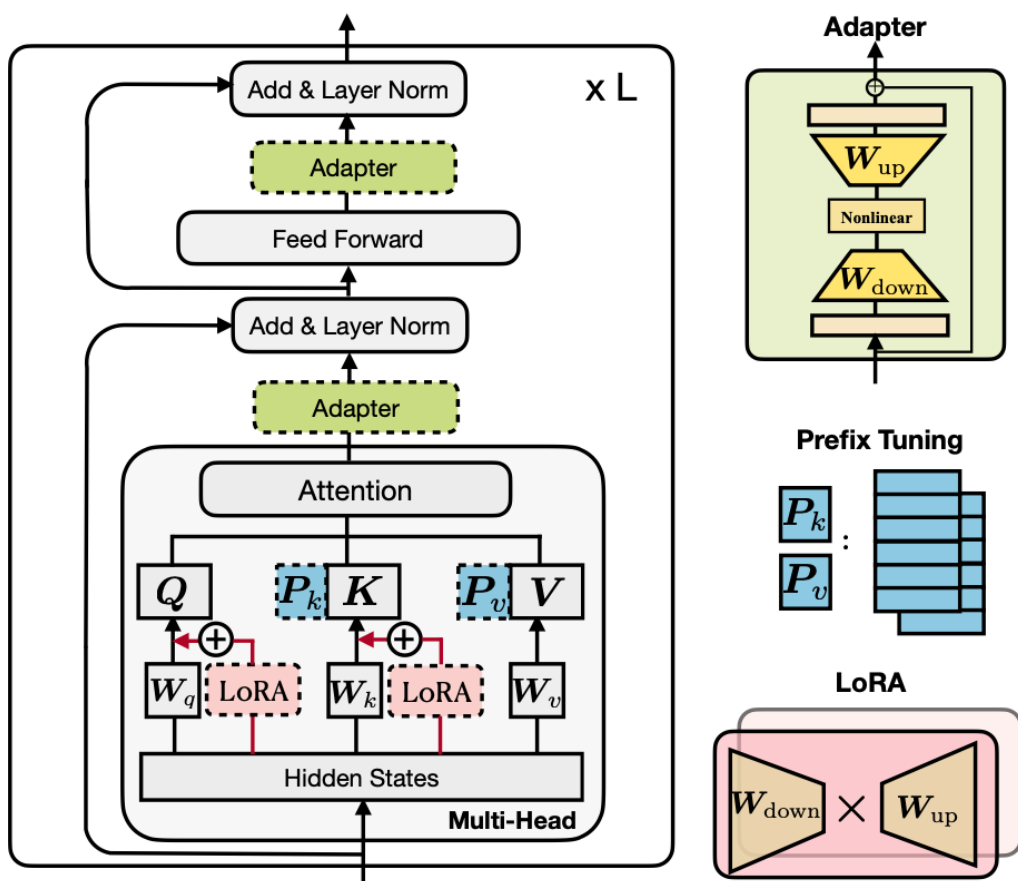


Figure 1. 具有参数高效调优方法的 Transformer 架构。虚线块代表添加的模块：(1) LoRA 通过低秩分解更新权重矩阵，(2) Adapters 插入投影层，(3) Prefix Tuning 将可学习向量添加到注意力的键/值之前。图片来源自 [8]

现代基于变压器的 LLM 通过包含两个核心组件的层叠处理输入：

2.2.1 Transformer 基础架构回顾

- 多头注意力 [6,14]：通过以下方式计算上下文关系：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

，其中查询 (Q)、键 (K) 和值 (V) 是隐藏状态的投影。

- 前馈网络 (FFN): 应用非线性变换:

$$\text{FFN}(x) = \text{GeLU}(W_2 \cdot \text{GeLU}(W_1 x + b_1)) + b_2$$

- 残差连接: 通过以下方式稳定训练:

$$x_{out} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

2.2.2 高效调优机制

有三种参数高效的策略 (图 1)，在本文中，我们只应用 QLoRA (低秩适应的量化版本):

1. LoRA (低秩适应):

- 分解权重更新: $\Delta W = AB$ ，其中 $A \in \mathbb{R}^{d \times r}$ ， $B \in \mathbb{R}^{r \times d}$
- 应用于注意力头中的 Q/K/V 投影
- 仅训练 A 和 B (可训练的% 的总参数取决于超参数 “rank”)

2. 适配器:

- 在注意力/FFN 后插入瓶颈层:

$$x_{adapted} = x + W_{up} \cdot \text{GeLU}(W_{down} \cdot x)$$

其中 $W_{down} \in \mathbb{R}^{d \times r}$ ， $W_{up} \in \mathbb{R}^{r \times d}$

3. 前缀调优:

- 在注意力键/值前添加可学习的向量 P_k, P_v :

$$K' = [P_k; K], \quad V' = [P_v; V]$$

2.2.3 合并过程

训练后，适配器参数通过以下方式集成到基础权重中:

$$W_{merged} = W_{base} + \Delta W_{adapter}$$

保持原始模型尺寸 (12 亿参数)，同时编码新的功能。这使得在部署时无需额外的计算开销。

2.3 用于微调目的的基础模型 & 训练数据

在本研究中，我们比较了两个基础模型的性能: Llama 3.2 1B 和 TinyLlama 1.1B。这两个模型均使用 Alpaca 数据集⁶ (高质量的指令跟随数据) 进行预训练和微调，其摘要如表 1 所示。这些模型从 Hugging Face⁵ 下载，并通过认可获得了这些模型的访问权限。

2.3.1 模型选择的合理性

Llama 3.2⁹ 提供了一个更强的基线 (更先进的模型架构)，而 TinyLlama 测试了效率极限 (较不先进的模型)。

我们比较 Llama 3.2 1B (1.24B 参数) 和 TinyLlama 1.1 1B，因为:

- 控制缩放：相似的数量（1.24B vs 1.1B）将架构效果与纯粹的规模差异分开
- 实际可访问性：二者均适合消费级硬件（T4 GPU），验证了我们方法的实际应用性
- 架构光谱：将 Llama 3.2 的优化注意力机制与 TinyLlama 的简化设计进行对比，测试 QLoRA 微调是否能提高较不先进模型的能力（超过较先进的模型（未进行微调））。
- QLoRA 方法的稳健性：对比不同的架构突显了 QLoRA 微调方法的稳健性。

Llama 3.2 1B 作为我们的性能上限——一个现代架构，在稀疏更新过程中我们测试其能力保留。TinyLlama 1.1B 代表效率前沿，我们测量参数减少是否能够通过重点适应微调数据集实现竞争性性能。

加上，

Llama 3.2 1B⁵ 是来自 Llama 家族的一个具有 12.4 亿参数的模型，以其性能和计算效率之间的平衡而闻名。在本研究中，它作为比较的基准。

Llama 1.1 1B 是 Llama 家族的一个较小变体，具有 11 亿个参数，专为资源受限的环境设计。尽管其规模较小，但它在更大子集的 Alpaca 数据集上进行了微调。（50,000 个样本）

所有实验均使用以下硬件和软件设置进行：

- 硬件：配有 Tesla T4 GPU 加速的 Google Colab。
- 软件：Python 3 环境。
- 微调框架：使用 Sloth 框架的 QLoRA⁸。

Table 1. 实验设置：基础模型、训练数据和微调方法

Model	Model Size	Training Data	Training Size	Method
Base Llama 3.2 1B	1.24B parameters	None (Pre-trained)	-	-
Fine-tuned Llama 3.2 1B	1.24B parameters	Alpaca Dataset	2,000 samples	QLoRA
TinyLlama 1.1 1B	1.1B parameters	Alpaca Dataset	50,000 samples	QLoRA

* limited to 2,000 due to hardware constraints, TinyLlama 1.1B takes around 14 hours to finetune

通过进行这个实验，我们可以确定：

- (1) 微调后，TinyLlama 1.1B 这种架构不太先进的模型是否能够超越 Llama 3.2 1B（Llama 家族的高级版本）；
- (2) 微调后，Llama 3.2 1B 是否能够超越其基础版本（未经过微调）；

2.4 基准数据集和评估指标

我们采用了三个已建立的基准测试，全面操作化了所有指标和变量。评估指标是通过利用 GitHub 开源框架 “lm-evaluation-harness” [7]⁷ 计算得出的：

- HellaSwag 数据集 [16]（常识推理）
 - 目的：评估日常推理的保存情况

⁵ meta-llama/Llama-3.2-1B, Hugging Face. <https://huggingface.co/meta-llama/Llama-3.2-1B>

⁶ Alpaca: A Strong, Replicable Instruction-Following Model, Stanford. <https://crfm.stanford.edu/2023/03/13/alpaca.html>

⁸ Finetune framework for LLM, Unsloth. <https://github.com/unslothai/unsloth>

⁹ Llama 3.2: Revolutionizing edge AI and vision with open, customizable models, Meta. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>

– 度量变量：

- * p_i : 模型对 i^{th} 样本的预测延续
- * t_i : 人类偏好的真实值延续
- * n : 总样本数 ($n = 10,000$)
- * $f(x)$: 归一化函数 = 小写 (去除标点 (x))
- * $A_{\text{norm}}^{\text{Base}}$: 基础模型的规范化准确率
- * $A_{\text{norm}}^{\text{FT}}$: 微调后的归一化准确率

– 度量标准：

$$A_{\text{norm}} = \frac{\sum_{i=1}^n \mathbb{I}(f(p_i)=f(t_i))}{n} \times 100\%$$

$$\Delta A = A_{\text{norm}}^{\text{FT}} - A_{\text{norm}}^{\text{Base}}$$

• GSM8K 数据集 [2] (数学推理)

– 目的：量化数学能力保留

– 度量变量：

- * p_i : 从模型输出中提取的数值答案
- * t_i : 真实数值
- * ϵ : 公差阈值 ($\epsilon = 0.01$)
- * $A_{\text{flex}}^{\text{Base}}$: 基础模型的灵活准确性
- * $A_{\text{flex}}^{\text{FT}}$: 微调模型的灵活准确性

– 指标：

$$A_{\text{strict}} = \frac{\sum \mathbb{I}(p_i=t_i)}{n} \times 100\% \quad (\text{Exact Match})$$

$$A_{\text{flex}} = \frac{\sum \mathbb{I}(|p_i-t_i|<\epsilon)}{n} \times 100\% \quad (\text{Flexible Extract})$$

$$\text{FR} = \left(1 - \frac{A_{\text{flex}}^{\text{FT}}}{A_{\text{flex}}^{\text{Base}}}\right) \times 100\% \quad (\text{Forgetting Rate})$$

• MMLU 数据集 [9] (领域知识)

– 目的：衡量事实知识的保持情况

– 度量变量：

- * p_i : 模型选择的选项 (A/B/C/D)
- * t_i : 正确选项标识符
- * z : Z 值 ($z = 1.96$)
- * A^{Base} : 基础模型准确率
- * A^{FT} : 微调模型的准确性
- * CI_{Base} : 基本模型置信区间
- * CI_{FT} : 微调模型置信区间

– 度量指标：

$$A = \frac{\sum \mathbb{I}(p_i=t_i)}{n} \times 100\% \quad (\text{Accuracy})$$

$$\Delta K = A^{\text{Base}} - A^{\text{FT}} \pm \sqrt{\text{CI}_{\text{Base}}^2 + \text{CI}_{\text{FT}}^2} \quad (\text{Knowledge Loss})$$

⁷ Implementation Details:

3 结果

如表 2 所示，通过我们的正式指标量化后，参数高效微调在能力领域中产生了不同的结果：

Table 2. 基础模型和微调模型的性能比较

Task	Metric	Base Llama 3.2 1B	Fine-tuned Llama 3.2 1B (2k)	TinyLlama 1.1B (50k)
GSM8K ²	Flexible Accuracy	33.51 % $\pm$ 1.30 %	3.71 % $\pm$ 0.52 %	2.81 % $\pm$ 0.45 %
	Strict Accuracy	33.51 % $\pm$ 1.30 %	3.11 % $\pm$ 0.48 %	2.20 % $\pm$ 0.40 %
	Forgetting Rate (FR)	-	88.92 % *	-
HellaSwag ¹	Accuracy	45.23 % $\pm$ 0.50 %	45.61 % $\pm$ 0.50 %	45.84 % $\pm$ 0.50 %
	Normalized Accuracy	60.77 % $\pm$ 0.49 %	61.20 % $\pm$ 0.49 %	59.26 % $\pm$ 0.49 %
	Ability Augmentation (ΔA)	-	+0.43 %	-
MMLU CS ³	Accuracy	47.00 % $\pm$ 5.02 %	34.00 % $\pm$ 4.76 %	27.00 % $\pm$ 4.46 %
	Knowledge Loss (ΔK)	-	13.00 % $\pm$ 6.93 % *	-

1. 在数学推理中的灾难性遗忘

- 关键指标：遗忘率 (FR) = 88.92 % * (GSM8K 灵活提取)
- 基础模型 vs. 微调模型：33.51 % $\pm$ 1.30 % $\rightarrow$ 3.71 % $\pm$ 0.52 %
- 解释：FR 指标显示数学推理能力几乎完全丧失（相对减少 89 %），超过了典型的灾难性遗忘阈值（超过 50 % FR）。

2. 保持的常识推理能力

- 关键指标：能力增强 (ΔA) = +0.43 % (HellaSwag 归一化准确率)
- 基础模型 vs. 微调模型：60.77 % $\pm$ 0.49 % $\rightarrow$ 61.20 % $\pm$ 0.49 %
- 解释：稳定的 ΔA 表明尽管存在：
 - ，结构性知识仍得以保留优先考虑会话能力的架构约束 常识模式的共享表示空间

3. 领域知识退化

- 关键指标：知识损失 (ΔK) = 13.00 % $\pm$ 6.93 % * (MMLU CS)
- 基础与微调：47.00 % $\pm$ 5.02 % $\rightarrow$ 34.00 % $\pm$ 4.76 %
- 解释：显著但部分的遗忘（相对损失 28 %）表明：
 - 事实性知识对参数更新的中等稳健性

• 所有指标均使用 lm-evaluation-harness [7] 计算

• 统计显著性：通过成对 t 检验 $p < 0.05$

• 置信区间：95% 置信水平 ($\alpha = 0.05$)

¹ Ability Augmentation calculated as $\Delta A = A_{\text{norm}}^{\text{FT}} - A_{\text{norm}}^{\text{Base}}$

² Forgetting Rate (FR) = $\left(1 - \frac{A_{\text{flex}}^{\text{FT}}}{A_{\text{flex}}^{\text{Base}}}\right) \times 100\%$

³ Knowledge Loss $\Delta K = A^{\text{Base}} - A^{\text{FT}} \pm \sqrt{\text{CI}_{\text{Base}}^2 + \text{CI}_{\text{FT}}^2}$

⁴ - indicates metric not applicable/comparable due to architectural differences

- 尽管客观上不对齐，仍存在残余的语义关联

4. 架构效率分析

- 跨模型比较：尽管数据量减少了 25 倍，经过微调的 Llama 3.2 ($34.00 \% \pm 4.76 \%$) 在 MMLU CS 上表现优于 TinyLlama ($27.00 \% \pm 4.46 \%$)
- 关键洞察：准确率差距（7 个百分点）持续存在于：
 - TinyLlama 的简化架构（模型架构较简单）
 - 显著更大的微调数据集（50k 对比 2k 样本）

4 讨论

微调策略必须在赋予模型新能力与知识保留之间取得平衡，尤其是对于较小的模型。未来的研究应探索混合微调数据集、参数高效技术和针对性的评估。研究揭示了在对大型语言模型进行微调时必须考虑的关键权衡：

任务特定能力：在没有任务特定例子的情况下进行微调可能会完全消除专业能力（如表 2 所示 - “Base Llama 3.2 1B” 和 “微调的 Llama 3.2 1B (2k)” 在数学推理前后的变化，遗忘率 (FR) = *88.92 %)

表征竞争：神经资源是有限的——提高一种能力往往以其他能力为代价

数据效率与知识保留：更多的数据不一定能克服架构限制

References

1. A. Aghajanyan, L. Zettlemoyer, and S. Gupta, Intrinsic dimensionality explains the effectiveness of language model fine-tuning, 2020, <http://dx.doi.org/10.48550/arxiv.2012.13255>.
2. K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, Training verifiers to solve math word problems, 2021, <http://dx.doi.org/10.48550/arxiv.2110.14168>.
3. T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer, 8-bit optimizers via block-wise quantization, 2021, <http://dx.doi.org/10.48550/arxiv.2110.02861>.
4. T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, QLoRA: Efficient Finetuning of Quantized LLMs, 2023, <http://dx.doi.org/10.48550/arxiv.2305.14314>.
5. T. Dettmers and L. Zettlemoyer, The case for 4-bit precision: k-bit Inference Scaling Laws, 2022, <http://dx.doi.org/10.48550/arxiv.2212.09720>.
6. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805*, 2018, <http://dx.doi.org/10.48550/arXiv.1810.04805>.
7. Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou, A framework for few-shot language model evaluation, 07 2024, <http://dx.doi.org/10.5281/zenodo.12608602>, available from: <https://zenodo.org/records/12608602>.
8. J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, Towards a unified view of parameter-efficient transfer learning, 2021, <http://dx.doi.org/10.48550/arxiv.2110.04366>.

9. D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, Measuring massive multitask language understanding, 2020, <http://dx.doi.org/10.48550/arxiv.2009.03300>.
10. 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, <http://dx.doi.org/10.48550/arxiv.1902.00751>.
11. 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, <http://dx.doi.org/10.48550/arxiv.2106.09685>.
12. X. L. Li and P. Liang, Prefix-tuning: Optimizing continuous prompts for generation, 2021, <http://dx.doi.org/10.48550/arxiv.2101.00190>.
13. S. J. Pan and Q. Yang, A survey on transfer learning, *IEEE Transactions on Knowledge and Data Engineering*, **22**(10):1345–1359, 2010, <http://dx.doi.org/10.1109/TKDE.2009.191>.
14. C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi, and J. Zhong, Attention is all you need in speech separation, pp. 21–25, 2021, <http://dx.doi.org/10.1109/ICASSP39728.2021.9413901>.
15. N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, Training deep neural networks with 8-bit floating point numbers, 2018, <http://dx.doi.org/10.48550/arxiv.1812.08011>.
16. R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, HellaSwag: Can a Machine Really Finish Your Sentence?, 2019, <http://dx.doi.org/10.48550/arxiv.1905.07830>.