LoRA: Low-Rank Adaptation of Large Language Models

Edward Hu* **Yelong Shen*** **Phillip Wallis** **Zeyuan Allen-Zhu**
Yuanzhi Li **Shean Wang** **Lu Wang** **Weizhu Chen**
Microsoft Corporation
{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(Version 2)

Abstract

自然语言处理的一个重要范式是在通用领域数据上进行大规模预训练，然后适配特定任务或领域。随着我们预训练的模型规模越来越大，对所有模型参数进行重新训练的全量微调变得越来越不可行。以 1750 亿参数的 GPT-3 为例，部署多个独立的微调模型实例，每个实例都有 1750 亿参数，成本高得令人却步。我们提出**低秩适配（Low - Rank Adaptation，简称 LoRA）**，它冻结预训练模型的权重，并在 Transformer 架构的每一层中注入可训练的低秩分解矩阵，极大地减少了下游任务的可训练参数数量。与使用 Adam 优化器进行微调的 GPT-3 175B 相比，LoRA 可以将可训练参数数量减少到原来的万分之一，将 GPU 内存需求减少到原来的三分之一。在 RoBERTa、DeBERTa、GPT-2 和 GPT-3 等模型上，LoRA 的模型质量与全量微调相当甚至更优，尽管其可训练参数更少、训练通量更高，并且与适配器方法不同，*LoRA 不会增加额外的推理延迟*。我们还对语言模型适配中的低秩性进行了实证研究，这有助于理解 LoRA 的有效性。我们发布了一个便于将 LoRA 与 PyTorch 模型集成的软件包，并在`https://github.com/microsoft/LoRA`上提供了针对 RoBERTa、DeBERTa 和 GPT-2 的实现代码及模型检查点。

1 引言

在自然语言处理的许多应用中，需要将一个大规模预训练语言模型适配到多个下游应用。这种适配通常通过*微调*完成，微调会更新预训练模型的所有参数。微调的主要缺点是新模型包含与原始模型相同数量的参数。随着更大的模型每隔几个月就被训练出来，这从 GPT-2 (Radford et al., b) 或 RoBERTa large (Liu et al., 2019) 的一个小小的" 不便" 变成了对于拥有 1750 亿可训练参数的 GPT-3 (Brown et al., 2020) 来说是一个关键的部署挑战。[1] 许多人试图通过仅适配部分参数或学习针对新任务的外部模块来缓解这一问题。这样，对于每个任务，我们只需要存储和加载少量特定任务的参数，并附加在预训练模型之上，极大地提高了部署时的操作效率。然而，现有技术常常会通过扩展模型深度引入推理延迟 (Houlsby

*Equal contribution.

[0]与 V1 相比，本草稿包括更好的基线、GLUE 上的实验以及更多关于适配器延迟的内容。

[1]虽然 GPT-3 175B 通过少样本学习实现了非平凡的性能，但微调显著提高了其性能，如Appendix A所示。

et al., 2019; Rebuffi et al., 2017)，或减少模型的可用序列长度 (Li & Liang, 2021; Lester et al., 2021; Hambardzumyan et al., 2020; Liu et al., 2021)（参见Section 3）。更重要的是，这些方法往往无法匹配微调基线，造成效率和模型质量之间的权衡。我们从 Li et al. (2018a); Aghajanyan et al. (2020) 的研究中获得启发，他们表明学习到的过参数化模型实际上位于低内在维度上。我们假设模型适配过程中权重的变化也具有低" 内在秩"，这导致了我们提出的 **Lo**w-**R**ank **A**daptation（LoRA）方法。LoRA 使我们能够通过优化密集层适配过程中变化的秩分解矩阵间接地训练神经网络中的某些密集层，同时保持预训练权重冻结，如Figure 1所示。以 GPT-3 175B 为例，我们展示即使全秩（即Figure 1中的 d）高达 12,288，非常低的秩（即 r 可以是 1 或 2）也已足够，使 LoRA 在存储和计算上都非常高效。

LoRA 具有几个关键优势。

- 一个预训练模型可以被共享并用于为不同任务构建许多小型 LoRA 模块。我们可以冻结共享模型，并通过替换Figure 1中的矩阵 A 和 B 来高效地切换任务，显著降低存储需求和任务切换开销。
- 使用自适应优化器时，LoRA 使训练更加高效，并将硬件准入门槛降低最多 3 倍，因为我们不需要为大多数参数计算梯度或维护优化器状态。相反，我们只优化注入的、小得多的低秩矩阵。
- 我们简单的线性设计允许我们在部署时将可训练矩阵与冻结权重合并，相比于完全微调的模型，不会引入推理延迟，这是其设计的固有特性。
- LoRA 与许多先前的方法正交，可以与许多方法组合，如前缀调优。我们在Appendix E中提供了一个示例。

术语和约定 我们频繁引用 Transformer 架构，并使用其维度的常规术语。我们称 Transformer 层的输入和输出维度大小为 d_{model}。我们使用 W_q、W_k、W_v 和 W_o 指代自注意力模块中的查询/键/值/输出投影矩阵。W 或 W_0 指预训练权重矩阵，ΔW 指适配期间的累积梯度更新。我们用 r 表示 LoRA 模块的秩。我们遵循 (Vaswani et al., 2017; Brown et al., 2020) 设定的约定，使用 Adam (Loshchilov & Hutter, 2019; Kingma & Ba, 2017) 进行模型优化，并使用 Transformer MLP 前馈维度 $d_{ffn} = 4 \times d_{model}$。

2 问题陈述

虽然我们的提议与训练目标无关，但我们以语言建模作为动机用例。下面是语言建模问题的简要描述，特别是给定特定任务提示的条件概率最大化。假设我们有一个由 Φ 参数化的预训练自回归语言模型 $P_\Phi(y|x)$。例如，$P_\Phi(y|x)$ 可以是基于 Transformer 架构 (Vaswani et al., 2017) 的通用多任务学习器，如 GPT (Radford et al., b; Brown et al., 2020)。考虑将此预训练模型适配到下游条件文本生成任务，如摘要、机器阅读理解（MRC）和自然语言到 SQL（NL2SQL）。每个下游任务由上下文-目标对的训练数据集表示：$\mathcal{Z} = \{(x_i, y_i)\}_{i=1,..,N}$，其中 x_i 和 y_i 都是标记序列。例如，在 NL2SQL 中，x_i 是自然语言查询，y_i 是对应的 SQL 命令；在摘要中，x_i 是文章内容，y_i 是其摘要。在全面微调期间，模型从预训练权重 Φ_0 初始化并

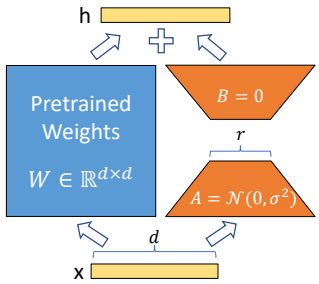

Figure 1: 重参数化。仅训练 A 和 B。

更新为 $\Phi_0 + \Delta\Phi$，通过重复遵循梯度以最大化条件语言建模目标：

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_{\Phi}(y_t|x, y_{<t})\right) \tag{1}$$

全面微调的主要缺点是对于每个下游任务，我们学习一个不同的参数集 $\Delta\Phi$，其维度 $|\Delta\Phi|$ 等于 $|\Phi_0|$。因此，如果预训练模型很大（如 GPT-3，$|\Phi_0| \approx 175$ 十亿），存储和部署许多独立的微调模型实例可能具有挑战性，甚至难以实现。在本文中，我们采用更加参数高效的方法，其中特定任务的参数增量 $\Delta\Phi = \Delta\Phi(\Theta)$ 由一个大小更小的参数集 Θ 进一步编码，满足 $|\Theta| \ll |\Phi_0|$。找到 $\Delta\Phi$ 的任务因此变为优化 Θ：

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right) \tag{2}$$

在随后的章节中，我们提出使用低秩表示来编码 $\Delta\Phi$，该表示在计算和内存上都是高效的。当预训练模型是 GPT-3 175B 时，可训练参数的数量 $|\Theta|$ 可以小至 $|\Phi_0|$ 的 0.01%。

3 现有解决方案真的足够好吗？

我们要解决的问题绝非新问题。自迁移学习诞生以来，已有数十项工作致力于使模型适配更加参数和计算高效。关于一些知名工作的调查，请参见Section 6。以语言建模为例，在高效适配方面有两种突出策略：添加适配器层 (Houlsby et al., 2019; Rebuffi et al., 2017; Pfeiffer et al., 2021; Rücklé et al., 2020) 或优化某种形式的输入层激活 (Li & Liang, 2021; Lester et al., 2021; Hambardzumyan et al., 2020; Liu et al., 2021)。然而，这两种策略都有其局限性，尤其在大规模和延迟敏感的生产场景中。

适配器层引入推理延迟 适配器有多种变体。我们聚焦于 Houlsby et al. (2019) 的原始设计，每个 Transformer 块有两个适配器层，以及 Lin et al. (2020) 最近的设计，每个块只有一个适配器层，并额外包含一个 LayerNorm (Ba et al., 2016)。尽管可以通过剪枝层或利用多任务设置 (Rücklé et al., 2020; Pfeiffer et al., 2021) 来减少整体延迟，但没有直接方法绕过适配器层的额外计算。这看起来似乎不是问题，因为适配器层设计时参数很少（有时 $<$ 原始模型的 1%），通过使用小的瓶颈维度来限制可以添加的 FLOPs。然而，大型神经网络依赖硬件并行性来保持低延迟，而适配器层必须按顺序处理。这在批量大小通常只有 1 的在线推理场景中会造成明显差异。在没有模型并行性的通用场景中，如在单个 GPU 上运行 GPT-2 (Radford et al., b) medium 推理，我们发现使用适配器时延迟会有明显增加，即使瓶颈维度非常小（Table 1）。当我们需要像 Shoeybi et al. (2020); Lepikhin et al. (2020) 那样分片模型时，这个问题会变得更糟，因为额外的深度需要更多同步 GPU 操作，如 `AllReduce` 和 `Broadcast`，除非我们多次冗余存储适配器参数。

直接优化提示很困难 另一个方向，如前缀调优 (Li & Liang, 2021) 所示，面临不同的挑战。我们观察到前缀调优难以优化，其性能随可训练参数的变化呈非单调变化，这印证了原论文中的类似观察。更根本的是，为适配保留序列长度的一部分必然会减少处理下游任务可用的序列长度，我们怀疑这使得提示调优的性能比其他方法更差。关于任务性能的研究，我们推迟到Section 5。

批量大小 序列长度 $\lvert\Theta\rvert$	32 512 0.5M	16 256 11M	1 128 11M
全微调/LoRA	1449.4±0.8	338.0±0.6	19.8±2.7
适配器$^{\text{L}}$	1482.0±1.0 (+2.2%)	354.8±0.5 (+5.0%)	23.9±2.1 (+20.7%)
适配器$^{\text{H}}$	1492.2±1.0 (+3.0%)	366.3±0.5 (+8.4%)	25.8±2.2 (+30.3%)

Table 1: GPT-2 medium 单次前向传播的推理延迟（毫秒），100 次试验的平均值。我们使用 NVIDIA Quadro RTX8000。"$\lvert\Theta\rvert$" 表示适配器层中可训练参数的数量。适配器$^{\text{L}}$ 和 适配器$^{\text{H}}$ 是两种适配器调优变体，我们在Section 5.1中描述。在在线、短序列长度场景中，适配器层引入的推理延迟可能很显著。完整研究请参见Appendix B。

4 我们的方法

我们描述 LoRA 的简单设计及其实际优势。此处概述的原则适用于深度学习模型中的任何密集层，尽管在实验中我们仅关注 Transformer 语言模型中的某些权重作为动机用例。

4.1 低秩参数化更新矩阵

神经网络包含许多执行矩阵乘法的密集层。这些层中的权重矩阵通常具有满秩。在适配特定任务时，Aghajanyan et al. (2020) 表明预训练语言模型具有低" 内在维度"，即使投影到更小的子空间也能有效学习。受此启发，我们假设权重更新在适配过程中也具有低" 内在秩"。对于预训练权重矩阵 $W_0 \in \mathbb{R}^{d\times k}$，我们通过低秩分解 $W_0 + \Delta W = W_0 + BA$ 约束其更新，其中 $B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}$，且秩 $r \ll \min(d, k)$。在训练期间，W_0 被冻结且不接收梯度更新，而 A 和 B 包含可训练参数。注意 W_0 和 $\Delta W = BA$ 与相同输入相乘，其各坐标输出向量相加。对于 $h = W_0x$，我们修改的前向传播产生：

$$h = W_0x + \Delta Wx = W_0x + BAx \tag{3}$$

我们在Figure 1中说明了我们的重参数化。我们对 A 使用随机高斯初始化，对 B 使用零初始化，因此训练开始时 $\Delta W = BA$ 为零。然后我们按 $\frac{\alpha}{r}$ 缩放 ΔWx，其中 α 是 r 中的常数。使用 Adam 优化时，调整 α 大致等同于调整学习率（如果适当缩放初始化）。因此，我们简单地将 α 设置为尝试的第一个 r，不对其进行调整。这种缩放有助于在改变 r 时减少重新调整超参数的需要 (Yang & Hu, 2021)。

全面微调的泛化. 更通用的微调形式允许训练预训练参数的子集。LoRA 更进一步，不要求权重矩阵的累积梯度更新在适配期间具有满秩。这意味着当将 LoRA 应用于所有权重矩阵并训练所有偏置时[2]，通过将 LoRA 秩 r 设置为预训练权重矩阵的秩，我们大致恢复全面微调的表达能力。换句话说，随着可训练参数数量的增加[3]，训练 LoRA 大致收敛到训练原始模型，而基于适配器的方法收敛到 MLP，基于前缀的方法收敛到无法处理长输入序列的模型。

无额外推理延迟. 在生产部署时，我们可以显式计算并存储 $W = W_0 + BA$，并照常执行推理。注意 W_0 和 BA 都在 $\mathbb{R}^{d\times k}$ 中。当需要切换到另一个下游任务时，我们可以通过减去

[2]与权重相比，它们代表可忽略数量的参数。
[3]在适配困难任务时不可避免。

BA 并添加不同的 $B'A'$ 来恢复 W_0，这是一个快速操作，内存开销极小。关键是，这保证了我们不会通过设计引入任何额外的推理延迟，与微调模型相比。

4.2 在 Transformer 中应用 LoRA

原则上，我们可以将 LoRA 应用于神经网络中任何权重矩阵的子集，以减少可训练参数的数量。在 Transformer 架构中，自注意力模块有四个权重矩阵（W_q, W_k, W_v, W_o），MLP 模块有两个。我们将 W_q（或 W_k、W_v）视为 $d_{model} \times d_{model}$ 维度的单个矩阵，尽管输出维度通常被切片到注意力头。为了简单和参数效率，我们将研究限制为**仅适配下游任务的注意力权重**并冻结 MLP 模块（因此不在下游任务中训练）。我们在**??**中进一步研究了在 Transformer 中适配不同类型注意力权重矩阵的效果。我们将对 MLP 层、LayerNorm 层和偏置进行经验性研究留待未来工作。

实际优势和局限性. 最显著的优势来自内存和存储使用的减少。对于用 Adam 训练的大型 Transformer，如果 $r \ll d_{model}$，我们可以将显存使用减少多达 2/3，因为我们不需要为冻结参数存储优化器状态。在 GPT-3 175B 上，我们将训练期间的显存消耗从 1.2TB 降低到 350GB。使用 $r = 4$ 并且仅适配查询和值投影矩阵时，检查点大小约减少 10,000 倍（从 350GB 降至 35MB）[4]。这使我们能够使用显著更少的 GPU 进行训练，并避免 I/O 瓶颈。另一个优势是我们可以在部署时仅通过交换 LoRA 权重，以更低的成本在任务间切换，而不是交换所有参数。这允许创建许多可以在存储预训练权重的 VRAM 机器上快速交换的定制模型。与全面微调相比，我们还观察到在 GPT-3 175B 上训练期间加速 25%[5]，因为我们不需要为绝大多数参数计算梯度。

LoRA 也有其局限性。例如，如果选择将 A 和 B 吸收到 W 中以消除额外推理延迟，那么在单次前向传播中批处理具有不同 A 和 B 的不同任务输入并不直接。尽管在延迟不重要的场景中，仍然可以不合并权重，并动态选择批次样本要使用的 LoRA 模块。

5 实验研究

我们评估 LoRA 在 RoBERTa (Liu et al., 2019)、DeBERTa (He et al., 2021) 和 GPT-2 (Radford et al., b) 上的下游任务性能，并最终扩展到 GPT-3 175B (Brown et al., 2020)。我们的实验涵盖从自然语言理解（NLU）到自然语言生成（NLG）的广泛任务。具体而言，我们在 GLUE (Wang et al., 2019) 基准上评估 RoBERTa 和 DeBERTa。对于 GPT-2，我们遵循 Li & Liang (2021) 的设置以进行直接比较，并在 GPT-3 上增加了 WikiSQL (Zhong et al., 2017)（自然语言到 SQL 查询）和 SAMSum (Gliwa et al., 2019)（对话摘要）进行大规模实验。关于我们使用的数据集的更多细节，请参见 Appendix C。所有实验均使用 NVIDIA Tesla V100 进行。

5.1 基线方法

为了广泛比较其他基线方法，我们复现了先前工作的实验设置，并尽可能复用其已报告的结果。然而，这意味着某些基线方法可能仅在特定实验中出现。

微调（FT）是一种常见的模型适配方法。在微调过程中，模型初始化为预训练权重和偏置，并对所有模型参数进行梯度更新。一个简单的变体是仅更新部分层，而冻结其他层。

[4]部署时我们仍然需要 350GB 模型；然而，存储 100 个适配模型只需要 350GB + 35MB * 100 ≈ 354GB，而不是 100 * 350GB ≈ 35TB。

[5]对于 GPT-3 175B，全面微调的训练吞吐量为每 V100 GPU 32.5 个标记/秒；使用相同数量的权重分片进行模型并行时，LoRA 的吞吐量为每 V100 GPU 43.1 个标记/秒。

模型 & 方法	可训练参数数量	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	平均值
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} (LoRA)	0.3M	$87.5_{\pm.3}$	**95.1**	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	**93.3**	$90.8_{\pm.1}$	**86.6**	**91.5**	**87.2**

Table 2: GLUE 基准上 RoBERTa 和 DeBERTa 的适配方法表现。MNLI 采用整体准确率，CoLA 采用 Matthew 相关系数，STS-B 采用 Pearson 相关系数，其他任务采用准确率。* 表示来自先前研究的结果。

我们包括了先前研究中报告的一种基线方法 (Li & Liang, 2021)，其在 GPT-2 上仅适配最后两层（$\textbf{FT}^{\textbf{Top2}}$）。

仅训练偏置或 BitFit 是一种仅训练偏置向量而冻结所有其他参数的基线方法。同时，BitFit (Zaken et al., 2021) 也研究了这一基线方法。

前缀嵌入调优（PreEmbed）在输入 token 之间插入特殊 token。这些特殊 token 具有可训练的词嵌入，并通常不在模型的词汇表中。这些 token 的放置位置会影响模型性能。我们关注两种放置方式：“前缀（prefixing）”，即将这些 token 置于提示词前；以及“中缀（infixing）”，即将其置于提示词后。二者均在 Li & Liang (2021) 中有所讨论。我们用 l_p（或 l_i）表示前缀（或中缀）token 的数量。可训练参数的数量为 $|\Theta| = d_{model} \times (l_p + l_i)$。

前缀层调优（PreLayer）是前缀嵌入调优的扩展。它不仅学习某些特殊 token 的词嵌入（等价于嵌入层后的激活值），还学习 Transformer 每一层后的激活值。前一层计算的激活值将被可训练参数替换。其可训练参数的数量为 $|\Theta| = L \times d_{model} \times (l_p + l_i)$，其中 L 为 Transformer 层数。

适配器调优（Adapter tuning）由 Houlsby et al. (2019) 提出，在自注意力模块（以及 MLP 模块）和后续残差连接之间插入适配器层。适配器层包含两个带有偏置的全连接层，并在其中加入非线性变换。我们称该原始设计为 $\textbf{Adapter}^{\textbf{H}}$。最近，Lin et al. (2020) 提出了更高效的设计，即仅在 MLP 模块之后以及 LayerNorm 之后应用适配器层。我们称其为 $\textbf{Adapter}^{\textbf{L}}$。该设计与 Pfeiffer et al. (2021) 提出的另一个设计相似，我们称之为 $\textbf{Adapter}^{\textbf{P}}$。此外，我们还包括另一种基线方法 AdapterDrop (Rücklé et al., 2020)，该方法通过去除部分适配器层以提高效率（$\textbf{Adapter}^{\textbf{D}}$）。为了尽可能扩大基线方法的比较范围，我们引用了先前工作的数据；这些数据在表格中第一列标有星号（*）。在所有情况下，可训练参数的数量为：

$$|\Theta| = \hat{L}_{Adpt} \times (2 \times d_{model} \times r + r + d_{model}) + 2 \times \hat{L}_{LN} \times d_{model}$$

其中，$\hat{L}_{Adpt}$ 为适配器层的数量，$\hat{L}_{LN}$ 为可训练的 LayerNorm 层数（例如在 $\text{Adapter}^{\text{L}}$ 中）。

LoRA 在现有权重矩阵的并行路径中添加可训练的秩分解矩阵对。如 Section 4.2 所述，为了简化实验，我们在大多数实验中仅对 W_q 和 W_v 应用 LoRA。可训练参数的数量取决于秩 r 以及原始权重的形状：

$$|\Theta| = 2 \times \hat{L}_{LoRA} \times d_{model} \times r$$

其中 $\hat{L}_{LoRA}$ 为应用 LoRA 的权重矩阵数量。

5.2 RoBERTa base/large

RoBERTa (Liu et al., 2019) 优化了最初在 BERT (Devlin et al., 2019a) 中提出的预训练方案，并提升了后者在任务上的表现，且没有引入更多可训练的参数。尽管近年来 RoBERTa

模型 & 方法	# 可训练参数 参数数量	E2E NLG 挑战 BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M (Adapter^{L})*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M (Adapter^{L})*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M (Adapter^{H})	11.09M	$67.3_{\pm.6}$	$8.50_{\pm.07}$	$46.0_{\pm.2}$	$70.7_{\pm.2}$	$2.44_{\pm.01}$
GPT-2 M (FT^{Top2})*	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (PreLayer)*	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.85}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{71.8}_{\pm.1}$	$\mathbf{2.53}_{\pm.02}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (Adapter^{L})	0.88M	$69.1_{\pm.1}$	$8.68_{\pm.03}$	$46.3_{\pm.0}$	$71.4_{\pm.2}$	$\mathbf{2.49}_{\pm.0}$
GPT-2 L (Adapter^{L})	23.00M	$68.9_{\pm.3}$	$8.70_{\pm.04}$	$46.1_{\pm.1}$	$71.3_{\pm.2}$	$2.45_{\pm.02}$
GPT-2 L (PreLayer)*	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.89}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{72.0}_{\pm.2}$	$2.47_{\pm.02}$

Table 3: 不同适配方法下的 GPT-2 medium（M）和 large（L）在 E2E NLG 挑战中的表现。所有指标越高越好。LoRA 在可训练参数相当或更少的情况下超越了多个基线。实验结果显示了置信区间。* 表示先前工作中发布的数字。

被许多更大规模的模型所超越，特别是在 GLUE 基准测试 (Wang et al., 2019) 上，但它仍然是一个在实际应用中因其规模而具有竞争力和受欢迎的预训练模型。我们从 HuggingFace Transformers 库 (Wolf et al., 2020) 中获取了预训练的 RoBERTa base（125M）和 RoBERTa large（355M），并评估了不同高效适配方法在 GLUE 基准测试任务上的表现。我们还根据 Houlsby et al. (2019) 和 Pfeiffer et al. (2021) 的设置进行了复制。为了确保公平比较，我们在与适配器比较时对 LoRA 评估方法进行了两项关键修改。首先，我们对所有任务使用相同的批量大小，并使用序列长度 128 来匹配适配器基线。其次，我们将模型初始化为预训练模型，用于 MRPC、RTE 和 STS-B，而不是已经适配 MNLI 的模型，这一点与微调基线不同。根据 Houlsby et al. (2019) 的更严格设置进行的实验使用 † 标记。结果见 Table 2（前三部分）。有关使用的超参数的详细信息，请参见 Section D.1。

5.3 DeBERTa XXL

DeBERTa (He et al., 2021) 是 BERT 的一个更新版本，经过大规模训练，在 GLUE (Wang et al., 2019) 和 SuperGLUE (Wang et al., 2020) 等基准测试中表现非常有竞争力。我们评估了 LoRA 是否能在 GLUE 上与完全微调的 DeBERTa XXL（1.5B）匹敌。结果见 Table 2（底部部分）。有关使用的超参数的详细信息，请参见 Section D.2。

5.4 GPT-2 medium/large

在展示 LoRA 在 NLU 上作为完整微调的竞争替代方案后，我们希望回答 LoRA 是否仍然能在 NLG 模型（如 GPT-2 medium 和 large (Radford et al., b)）中占据优势。我们尽可能将设置与 Li & Liang (2021) 保持一致，以进行直接比较。由于篇幅限制，本节仅展示 E2E NLG Challenge（Table 3）上的结果。有关 WebNLG (Gardent et al., 2017) 和 DART (Nan et al., 2020) 上的结果，请参见 Section F.1。我们在 Section D.3 中列出了使用的超参数。

模型 & 方法	# 可训练参数 参数数量	WikiSQL 准确率 (%)	MNLI-m 准确率 (%)	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

Table 4: 不同适配方法在 GPT-3 175B 上的表现。我们报告了 WikiSQL 的逻辑形式验证准确率、MultiNLI-matched 的验证准确率以及 SAMSum 的 Rouge-1/2/L。LoRA 在多个方法中表现最佳，甚至超过了完整微调的效果。WikiSQL 的结果波动范围约为 ±0.5%，MNLI-m 的波动范围约为 ±0.1%，SAMSum 的结果波动范围分别为 ±0.2/±0.2/±0.1。

5.5 扩展到 GPT-3 175B

作为对 LoRA 的最终压力测试，我们将规模扩展到具有 1750 亿参数的 GPT-3。由于训练成本较高，我们仅报告了每个任务在随机种子下的典型标准差，而不是为每个条目提供标准差。有关使用的超参数的详细信息，请参见 Section D.4。

如 Table 4所示，LoRA 在所有三个数据集上与微调基线的表现相当或超过基线。注意，并非所有方法在增加更多可训练参数时都会表现得更加优越，如 Figure 2所示。我们观察到，当使用超过 256 个特殊标记进行前缀嵌入微调或超过 32 个特殊标记进行前缀层微调时，性能显著下降。这与 Li & Liang (2021) 中的类似观察结果一致。虽然对这一现象的深入调查超出了本工作的范围，但我们推测，增加更多的特殊标记会导致输入分布与预训练数据分布的差距进一步加大。另外，我们在 Section F.3 中研究了不同适配方法在低数据环境下的表现。

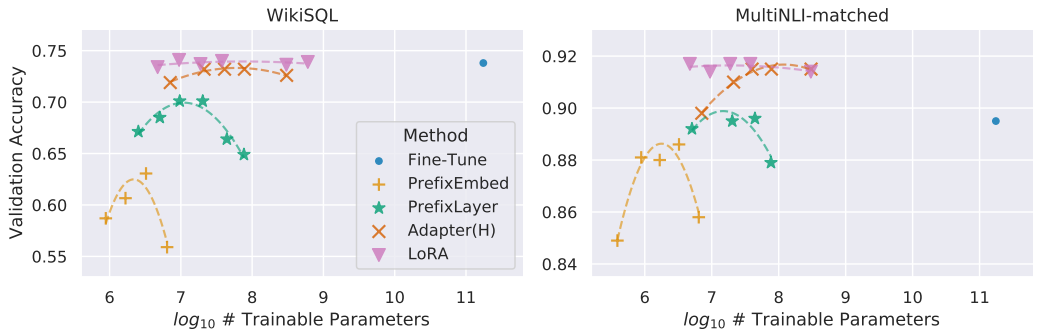

Figure 2: GPT-3 175B 在 WikiSQL 和 MNLI-matched 上的验证准确率与多种适配方法的可训练参数数量的关系。LoRA 展现了更好的可扩展性和任务性能。更多数据点的详细信息请参见 Section F.2。

6 相关工作

Transformer 语言模型。 Transformer (Vaswani et al., 2017) 是一种大量使用自注意力的序列到序列架构。Radford et al. (a) 通过使用 Transformer 解码器堆栈将其应用于自回归语言建模。此后，基于 Transformer 的语言模型在许多任务中占据主导地位，并取得了最先进的成果。随着 BERT (Devlin et al., 2019b) 和 GPT-2 (Radford et al., b)，一个新范式出现——这两者都是

在大量文本上训练的大型 Transformer 语言模型——在通用领域预训练后对特定任务数据进行微调，相比直接在特定任务数据上训练，能提供显著的性能提升。训练更大的 Transformer 通常会带来更好的性能，这仍是一个活跃的研究方向。GPT-3 (Brown et al., 2020) 是迄今为止最大的单一 Transformer 语言模型，拥有 1750 亿参数。

提示工程和微调. 尽管 GPT-3 175B 可以仅通过几个额外的训练示例适应其行为，但结果严重依赖输入提示 (Brown et al., 2020)。这需要在撰写和格式化提示以最大化模型在特定任务上的性能方面进行经验性探索，这被称为提示工程或提示黑客。微调将在通用领域预训练的模型重新训练到特定任务 Devlin et al. (2019b); Radford et al. (a)。其变体包括仅学习参数子集 Devlin et al. (2019b); Collobert & Weston (2008)，但从业者通常重新训练所有参数以最大化下游性能。然而，GPT-3 175B 的庞大规模使得以通常方式进行微调变得具有挑战性，因为它会产生大型检查点，并且由于具有与预训练相同的内存占用，硬件准入门槛很高。

参数高效适配. 许多人提出在神经网络现有层间插入*适配器层* (Houlsby et al., 2019; Rebuffi et al., 2017; Lin et al., 2020)。我们的方法使用类似的瓶颈结构对权重更新施加低秩约束。关键的功能差异是我们学习的权重可以在推理时与主权重合并，因此不会引入任何延迟，而适配器层则不同（见Section 3）。适配器的一个同期扩展是 compacter (Mahabadi et al., 2021)，本质上是使用具有某些预定义权重共享方案的 Kronecker 积来参数化适配器层。类似地，将 LoRA 与其他基于张量积的方法相结合可能会提高其参数效率，这我们留待未来工作。最近，许多人提出优化输入词嵌入而非微调，类似于提示工程的连续可微广义 (Li & Liang, 2021; Lester et al., 2021; Hambardzumyan et al., 2020; Liu et al., 2021)。我们在实验部分包括与 Li & Liang (2021) 的比较。然而，这类工作只能通过使用更多特殊提示标记来扩展，而这些标记在学习位置嵌入时会占用任务标记的可用序列长度。

深度学习中的低秩结构. 低秩结构在机器学习中非常常见。许多机器学习问题存在某种内在低秩结构 (Li et al., 2016; Cai et al., 2010; Li et al., 2018b; Grasedyck et al., 2013)。此外，众所周知，对于许多深度学习任务，尤其是具有严重过参数化神经网络的任务，训练后学习到的神经网络会呈现出低秩特性 (Oymak et al., 2019)。一些先前的工作甚至在训练原始神经网络时显式施加低秩约束 (Sainath et al., 2013; Povey et al., 2018; Zhang et al., 2014; Jaderberg et al., 2014; Zhao et al., 2016; Khodak et al., 2021; Denil et al., 2014)；然而，据我们所知，这些工作都未考虑对冻结模型进行低秩更新以*适配下游任务*。在理论文献中，已知当底层概念类具有某些低秩结构时，神经网络会优于其他经典学习方法，包括相应的（有限宽度）神经切线核 (Allen-Zhu et al., 2019; Li & Liang, 2018) (Ghorbani et al., 2020; Allen-Zhu & Li, 2019; Allen-Zhu & Li, 2020a)。Allen-Zhu & Li (2020b) 中的另一个理论结果表明低秩适配可能有助于对抗性训练。总的来说，我们相信文献很好地支持了我们提出的低秩适配更新。

不同 r 之间的子空间相似性. 给定使用*相同预训练模型*学习的秩 $r = 8$ 和 64 的适配矩阵 $A_{r=8}$ 和 $A_{r=64}$，我们进行奇异值分解并获得右奇异单位矩阵 $U_{A_{r=8}}$ 和 $U_{A_{r=64}}$。[6] 我们希望回答：$U_{A_{r=8}}$ 中前 i 个奇异向量所跨越的子空间有多少包含在 $U_{A_{r=64}}$ 的前 j 个奇异向量所跨越的子空间中（对于 $1 \le i \le 8$ 和 $1 \le j \le 64$）？我们使用基于 Grassmann 距离的归一化子空间相似性来测量这个量（更正式的讨论请参见Appendix G）

$$\phi(A_{r=8}, A_{r=64}, i, j) = \frac{||U^{i\top}_{A_{r=8}} U^{j}_{A_{r=64}}||_F^2}{\min(i, j)} \in [0, 1] \tag{4}$$

[6]注意，类似的分析可以使用 B 和左奇异单位矩阵进行——我们在实验中坚持使用 A。

其中 $U^i_{A_{r=8}}$ 表示 $U_{A_{r=8}}$ 中对应于前 i 个奇异向量的列。$\phi(\cdot)$ 的范围是 $[0, 1]$，其中 1 表示子空间完全重叠，0 表示完全分离。参见 Figure 3 了解 ϕ 如何随 i 和 j 变化。由于空间限制，我们仅查看第 48 层（共 96 层），但结论对其他层也成立，如Section H.1所示。我们从 Figure 3

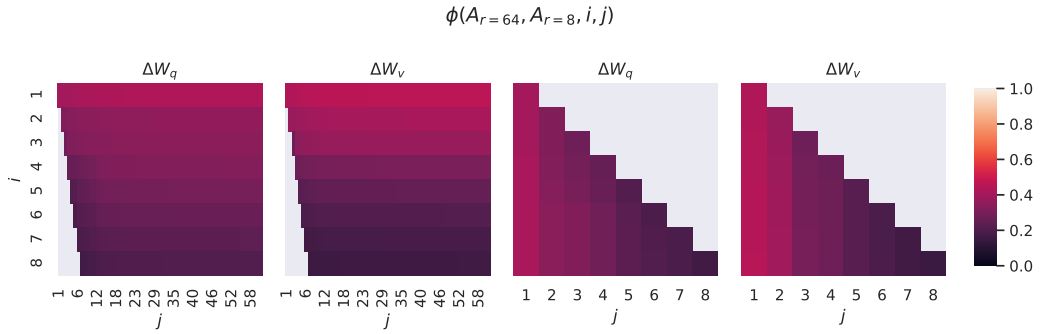

Figure 3: ΔW_q 和 ΔW_v 的 $A_{r=8}$ 和 $A_{r=64}$ 的列向量之间的子空间相似性。第三和第四图放大了前两幅图的左下角三角形。$r = 8$ 中的顶部方向包含在 $r = 64$ 中，反之亦然。

中做出重要观察。

> 对应于顶部奇异向量的方向在 $A_{r=8}$ 和 $A_{r=64}$ 之间显著重叠，而其他方向则不然。具体来说，$A_{r=8}$ 的 ΔW_v（resp. ΔW_q）和 $A_{r=64}$ 的 ΔW_v（resp. ΔW_q）共享维度为 1 的子空间，归一化相似性 > 0.5，这解释了为什么在我们的下游任务中 GPT-3 的 $r = 1$ 表现相当出色。

由于 $A_{r=8}$ 和 $A_{r=64}$ 都是使用相同的预训练模型学习的，Figure 3 表明 $A_{r=8}$ 和 $A_{r=64}$ 的顶部奇异向量方向最有用，而其他方向可能主要包含训练过程中累积的随机噪声。因此，适配矩阵确实可以具有非常低的秩。**不同随机种子间的子空间相似性.** 我们通过绘制两个随

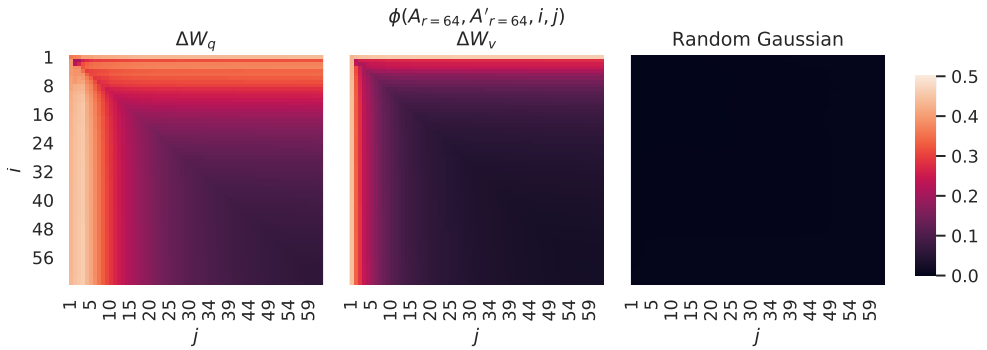

Figure 4: ** 左图和中图：** 两个随机种子下 ΔW_q 和 ΔW_v（在第 48 层）的 $A_{r=64}$ 列向量之间的归一化子空间相似性。** 右图：** 两个随机高斯矩阵的列向量之间的相同热图。其他层请参见Section H.1。

机种子运行（$r = 64$）之间的归一化子空间相似性来进一步确认这一点，如 Figure 4 所示。ΔW_q 似乎具有比 ΔW_v 更高的" 内在秩"，因为两次运行对 ΔW_q 学习了更多公共奇异值方向，这与我们在 **??** 中的经验观察一致。作为对比，我们还绘制了两个随机高斯矩阵，它们彼此之间不共享任何公共奇异值方向。

6.1 适配矩阵 ΔW 如何与 W 比较？

我们进一步研究 ΔW 和 W 之间的关系。特别是，ΔW 是否与 W 高度相关？（或从数学角度说，ΔW 是否主要包含在 W 的顶部奇异方向中？）另外，ΔW 相比其在 W 中对应的方向有多" 大"？这可以揭示适配预训练语言模型的潜在机制。为回答这些问题，我们通过计算 $U^\top W V^\top$ 将 W 投影到 ΔW 的 r 维子空间，其中 U/V 是 ΔW 的左/右奇异向量矩阵。然后，我们比较 $\|U^\top W V^\top\|_F$ 和 $\|W\|_F$ 的 Frobenius 范数。作为对比，我们还通过将 U, V 替换为 W 或随机矩阵的前 r 个奇异向量来计算 $\|U^\top W V^\top\|_F$。我们从 Table 5 得出几个结

	$r=4$			$r=64$		
	ΔW_q	W_q	Random	ΔW_q	W_q	Random
$\|\|U^\top W_q V^\top\|\|_F =$	0.32	21.67	0.02	1.90	37.71	0.33
$\|\|W_q\|\|_F = 61.95$	$\|\|\Delta W_q\|\|_F = 6.91$			$\|\|\Delta W_q\|\|_F = 3.57$		

Table 5: $U^\top W_q V^\top$ 的 Frobenius 范数，其中 U 和 V 是 ΔW_q 的前 r 个左/右奇异向量方向，或 (1) ΔW_q，(2) W_q，或 (3) 随机矩阵。权重矩阵取自 GPT-3 的第 48 层。

论。首先，ΔW 与 W 的相关性比随机矩阵更强，表明 ΔW 放大了 W 中已有的一些特征。其次，ΔW 并不重复 W 的顶部奇异方向，而是放大了 W 中未强调的方向。第三，放大因子相当大：对于 $r = 4$，约为 $21.5 \approx 6.91/0.32$。关于 $r = 64$ 为何有较小的放大因子，请参见Section H.4。我们还在Section H.3 中提供了一个可视化，展示随着包含更多来自 W_q 的顶部奇异方向，相关性如何变化。这表明低秩适配矩阵可能放大了在通用预训练模型中已学习但未强调的针对特定下游任务的重要特征。

7 结论与未来工作

微调巨大的语言模型在所需硬件和为不同任务托管独立实例的存储/切换成本方面极其昂贵。我们提出 LoRA，一种高效的适配策略，既不引入推理延迟，也不减少输入序列长度，同时保持高模型质量。重要的是，通过共享绝大多数模型参数，它允许在部署为服务时快速切换任务。虽然我们聚焦于 Transformer 语言模型，但所提出的原则通常适用于具有密集层的任何神经网络。未来工作有许多方向。1) LoRA 可以与其他高效适配方法相结合，可能提供正交改进。2) 微调或 LoRA 背后的机制尚不清晰——预训练期间学习的特征如何转化以在下游任务中表现良好？我们相信 LoRA 比全面微调更有助于回答这个问题。3) 我们主要依赖启发式方法选择应用 LoRA 的权重矩阵。有没有更有原则的方法？4) 最后，ΔW 的秩亏性表明 W 也可能是秩亏的，这也可以成为未来工作的灵感源泉。

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. *arXiv:2012.13255 [cs]*, December 2020. URL `http://arxiv.org/abs/2012.13255`.

Zeyuan Allen-Zhu and Yuanzhi Li. What Can ResNet Learn Efficiently, Going Beyond Kernels? In *NeurIPS*, 2019. Full version available at `http://arxiv.org/abs/1905.10337`.

Zeyuan Allen-Zhu and Yuanzhi Li. Backward feature correction: How deep learning performs deep learning. *arXiv preprint arXiv:2001.04413*, 2020a.

Zeyuan Allen-Zhu and Yuanzhi Li. Feature purification: How adversarial training performs robust deep learning. *arXiv preprint arXiv:2005.10190*, 2020b.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019. Full version available at `http://arxiv.org/abs/1811.03962`.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

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. *arXiv:2005.14165 [cs]*, July 2020. URL `http://arxiv.org/abs/2005.14165`.

Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017. doi: 10.18653/v1/s17-2001. URL `http://dx.doi.org/10.18653/v1/S17-2001`.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pp. 160–167, New York, NY, USA, July 2008. Association for Computing Machinery. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL `https://doi.org/10.1145/1390156.1390177`.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning, 2014.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019b. URL `http://arxiv.org/abs/1810.04805`. arXiv: 1810.04805.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. URL `https://aclanthology.org/I05-5002`.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pp. 124–133, 2017.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *arXiv preprint arXiv:2006.13409*, 2020.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *CoRR*, abs/1911.12237, 2019. URL http://arxiv.org/abs/1911.12237.

Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

Jihun Ham and Daniel D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *ICML*, pp. 376–383, 2008. URL https://doi.org/10.1145/1390156.1390204.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. WARP: Word-level Adversarial ReProgramming. *arXiv:2101.00121 [cs]*, December 2020. URL http://arxiv.org/abs/2101.00121. arXiv: 2101.00121.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-Efficient Transfer Learning for NLP. *arXiv:1902.00751 [cs, stat]*, June 2019. URL http://arxiv.org/abs/1902.00751.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

Mikhail Khodak, Neil Tenenholtz, Lester Mackey, and Nicolò Fusi. Initialization and regularization of factorized neural layers, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020.

Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale for Parameter-Efficient Prompt Tuning. *arXiv:2104.08691 [cs]*, April 2021. URL http://arxiv.org/abs/2104.08691. arXiv: 2104.08691.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the Intrinsic Dimension of Objective Landscapes. *arXiv:1804.08838 [cs, stat]*, April 2018a. URL http://arxiv.org/abs/1804.08838. arXiv: 1804.08838.

Xiang Lisa Li and Percy Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *arXiv:2101.00190 [cs]*, January 2021. URL http://arxiv.org/abs/2101.00190.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, 2018.

Yuanzhi Li, Yingyu Liang, and Andrej Risteski. Recovery guarantee of weighted low-rank approximation via alternating minimization. In *International Conference on Machine Learning*, pp. 2358–2367. PMLR, 2016.

Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference On Learning Theory*, pp. 2–47. PMLR, 2018b.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 441–459, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.41. URL https://aclanthology.org/2020.findings-emnlp.41.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT Understands, Too. *arXiv:2103.10385 [cs]*, March 2021. URL http://arxiv.org/abs/2103.10385. arXiv: 2103.10385.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers, 2021.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, et al. Dart: Open-domain structured data record to text generation. *arXiv preprint arXiv:2007.02871*, 2020.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*, 2017.

Samet Oymak, Zalan Fabian, Mingchen Li, and Mahdi Soltanolkotabi. Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian. *arXiv preprint arXiv:1906.05392*, 2019.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning, 2021.

Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohammadi, and Sanjeev Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Interspeech*, pp. 3743–3747, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. pp. 12, a.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. pp. 24, b.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018. URL http://arxiv.org/abs/1806.03822.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *arXiv:1705.08045 [cs, stat]*, November 2017. URL http://arxiv.org/abs/1705.08045. arXiv: 1705.08045.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers, 2020.

Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6655–6659. IEEE, 2013.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6000–6010, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems, 2020.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://www.aclweb.org/anthology/N18-1101.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Greg Yang and Edward J. Hu. Feature Learning in Infinite-Width Neural Networks. *arXiv:2011.14522 [cond-mat]*, May 2021. URL `http://arxiv.org/abs/2011.14522`. arXiv: 2011.14522.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2021.

Yu Zhang, Ekapol Chuangsuwanich, and James Glass. Extracting deep neural network bottleneck features using low-rank matrix factorization. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 185–189. IEEE, 2014.

Yong Zhao, Jinyu Li, and Yifan Gong. Low-rank plus diagonal adaptation for deep neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5005–5009. IEEE, 2016.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017. URL `http://arxiv.org/abs/1709.00103`.

A 大型语言模型仍需参数更新

少样本学习，或称提示工程，在只有少量训练样本时非常有优势。然而，在实践中，我们通常能够为对性能敏感的应用收集数千个或更多训练示例。如 Table 6所示，与少样本学习相比，微调可以显著提高大小数据集上的模型性能。我们从 GPT-3 论文 (Brown et al., 2020) 中获取了 RTE 上的 GPT-3 少样本结果。对于 MNLI-matched，我们使用每个类别两个演示，总共六个上下文内示例。

方法	MNLI-m（验证准确率/%）	RTE（验证准确率/%）
GPT-3 少样本	40.6	69.0
GPT-3 微调	89.5	85.4

Table 6: 微调在 GPT-3 上显著优于少样本学习 (Brown et al., 2020)。

B 适配器层引入的推理延迟

适配器层是以顺序方式添加到预训练模型的外部模块，而我们的建议 LoRA 可以看作是以并行方式添加的外部模块。因此，适配器层必须在基础模型之外额外计算，不可避免地引入额外延迟。正如 Rücklé et al. (2020) 指出的，当模型批量大小和/或序列长度足以充分利用硬件并行性时，适配器层引入的延迟可以得到缓解。我们通过对 GPT-2 medium 进行类似的延迟研究确认了他们的观察，并指出在某些场景下，尤其是批量大小较小的在线推理场景，增加的延迟可能会很显著。我们通过在 NVIDIA Quadro RTX8000 上对 100 次试验取平均，测量单次前向传播的延迟。我们变化输入批量大小、序列长度和适配器瓶颈维度 r。我们测试两种适配器设计： Houlsby et al. (2019) 的原始设计，我们称之为 $\text{Adapter}^{\text{H}}$，以及 Lin et al. (2020) 最近的更高效变体，我们称之为 $\text{Adapter}^{\text{L}}$。更多关于这些设计的详情请参见 Section 5.1。我们在 Figure 5中绘制了与无适配器基准线（$r = 0$）相比的百分比减速。

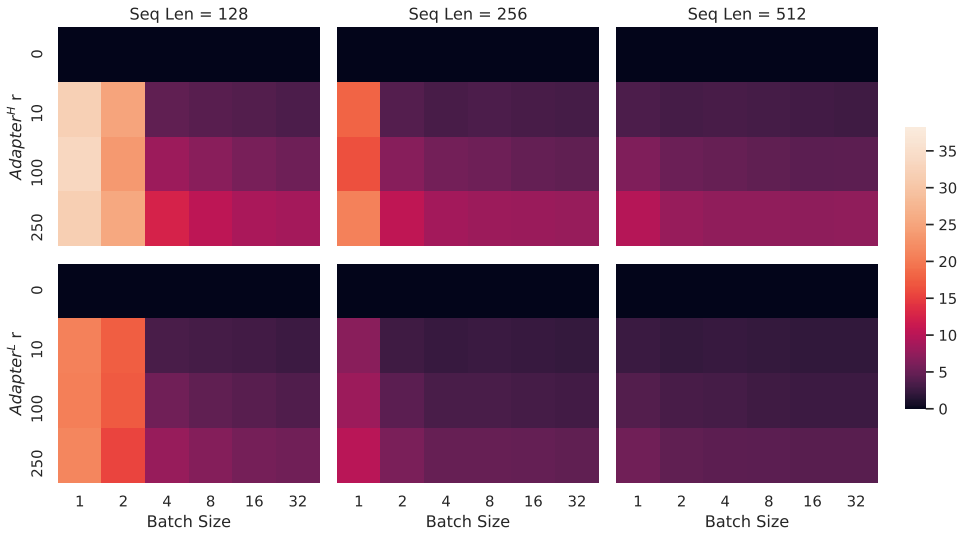

Figure 5: 与无适配器（$r = 0$）基准线相比的推理延迟百分比减速。顶部行显示 Adapter$^{\text{H}}$ 的结果，底部行显示 Adapter$^{\text{L}}$ 的结果。较大的批量大小和序列长度有助于缓解延迟，但在在线、短序列长度场景中，减速可能高达 30% 以上。我们调整了颜色映射以提高可见性。

C 数据集详情

GLUE 基准测试是一个覆盖广泛的自然语言理解任务集合。它包括 MNLI（推理，Williams et al. (2018)），SST-2（情感分析， Socher et al. (2013)），MRPC（释义检测， Dolan & Brockett (2005)），CoLA（语言可接受性， Warstadt et al. (2018)），QNLI（推理， Rajpurkar et al. (2018)），QQP[7]（问答），RTE（推理）以及 STS-B（文本相似性， Cer et al. (2017)）。其广泛的覆盖范围使 GLUE 基准测试成为评估 RoBERTa 和 DeBERTa 等 NLU 模型的标准指标。各个数据集以不同的许可证发布。

WikiSQL 在 Zhong et al. (2017) 中引入，包含 $56,355/8,421$ 个训练/验证示例。任务是从自然语言问题和表模式生成 SQL 查询。我们将上下文编码为 $x = \{\text{表模式}, \text{查询}\}$，目标为 $y = \{\text{SQL}\}$。该数据集以 BSD 3-Clause 许可证发布。

SAMSum 在 Gliwa et al. (2019) 中引入，包含 $14,732/819$ 个训练/测试示例。它由两人之间的分阶段聊天对话及语言学家撰写的相应抽象摘要组成。我们将上下文编码为用”\n” 连接的话语，后跟”\n\n”，目标为 $y = \{\text{摘要}\}$。该数据集以非商业许可证：Creative Commons BY-NC-ND 4.0 发布。

E2E 自然语言生成挑战最初在 Novikova et al. (2017) 中引入，作为训练端到端、数据驱动的自然语言生成系统的数据集，通常用于数据到文本的评估。E2E 数据集包含大约 $42,000$ 个训练、$4,600$ 个验证和 $4,600$ 个测试示例，均来自餐厅领域。每个用作输入的源表可以有多个引用。每个样本输入 (x, y) 由一系列插槽-值对组成，以及对应的自然语言引用文本。该数据集以 Creative Commons BY-NC-SA 4.0 发布。

DART 是Nan et al. (2020) 中描述的一个开放域数据到文本数据集。DART 输入被构造为 ENTITY | RELATION | ENTITY 三元组序列。拥有约 $82K$ 个示例，DART 是一个比 E2E 更大、更复杂的数据到文本任务。该数据集以 MIT 许可证发布。

[7] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

WebNLG 是另一个常用于数据到文本评估的数据集 (Gardent et al., 2017)。总共有约 $22K$ 个示例，WebNLG 包含 14 个不同的类别，其中 9 个在训练期间可见。由于 14 个总类别中有 5 个在训练期间未见，但在测试集中有表示，因此评估通常按" 可见" 类别（S)、" 不可见" 类别（U）和" 全部"（A）划分。每个输入示例由 SUBJECT | PROPERTY | OBJECT 三元组序列表示。该数据集以 Creative Commons BY-NC-SA 4.0 发布。

D 实验中使用的超参数

D.1 RoBERTa

我们使用 AdamW 和线性学习率衰减调度进行训练。我们对 LoRA 进行学习率、训练轮数和批量大小的网格搜索。遵循 Liu et al. (2019)，在适配 MRPC、RTE 和 STS-B 时，我们将 LoRA 模块初始化为最佳 MNLI 检查点；预训练模型对所有任务保持冻结。我们报告 5 个随机种子的中位数；每次运行的结果取最佳轮次。为了与 Houlsby et al. (2019) 和 Pfeiffer et al. (2021) 中的设置进行公平比较，我们将模型序列长度限制为 128，并对所有任务使用固定批量大小。重要的是，我们在适配 MRPC、RTE 和 STS-B 时从预训练的 RoBERTa 大模型开始，而不是已经适配到 MNLI 的模型。使用这种受限设置的运行将标记为 †。请参见 **??**中我们运行时使用的超参数。

D.2 DeBERTa

我们再次使用 AdamW 和线性学习率衰减调度进行训练。遵循 He et al. (2021)，我们调整学习率、丢弃概率、预热步数和批量大小。我们使用与 (He et al., 2021) 相同的模型序列长度，以保持我们的比较公平。遵循 He et al. (2021)，在适配 MRPC、RTE 和 STS-B 时，我们将 LoRA 模块初始化为最佳 MNLI 检查点，而不是常规初始化；预训练模型对所有任务保持冻结。我们报告 5 个随机种子的中位数；每次运行的结果取最佳轮次。请参见 Table 7中我们运行时使用的超参数。

方法	数据集	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
	优化器	AdamW							
	预热比率	0.1							
	学习率调度	线性							
DeBERTa XXL LoRA	批量大小	8	8	32	4	6	8	4	4
	训练轮数	5	16	30	10	8	11	11	10
	学习率	1E-04	6E-05	2E-04	1E-04	1E-04	1E-04	2E-04	2E-04
	权重衰减	0	0.01	0.01	0	0.01	0.01	0.01	0.1
	CLS 丢弃	0.15	0	0	0.1	0.1	0.2	0.2	0.2
	LoRA 配置	$r_q = r_v = 8$							
	LoRA α	8							
	最大序列长度	256	128	128	64	512	320	320	128

Table 7: DeBERTa XXL 在 GLUE 基准测试中包含的任务的超参数。

D.3 GPT-2

我们使用 AdamW (Loshchilov & Hutter, 2017) 对所有 GPT-2 模型进行训练，采用线性学习率调度，训练 5 个轮次。我们使用 Li & Liang (2021) 中描述的批量大小、学习率和波束搜索波束大小。相应地，我们也为 LoRA 调整了上述超参数。我们报告 3 个随机种子的平均

值；每次运行的结果取最佳轮次。GPT-2 中 LoRA 使用的超参数列在 Table 8中。关于其他基准的超参数，请参见 Li & Liang (2021)。

数据集	E2E	WebNLG	DART
	训练		
优化器	AdamW		
权重衰减	0.01	0.01	0.0
丢弃概率	0.1	0.1	0.0
批量大小	8		
训练轮数	5		
预热步数	500		
学习率调度	线性		
标签平滑	0.1	0.1	0.0
学习率	0.0002		
适配	$r_q = r_v = 4$		
LoRA α	32		
	推理		
波束大小	10		
长度惩罚	0.9	0.8	0.8
重复 n-gram 大小限制	4		

Table 8: GPT-2 LoRA 在 E2E、WebNLG 和 DART 上的超参数。

D.4 GPT-3

对于所有 GPT-3 实验，我们使用 AdamW (Loshchilov & Hutter, 2017) 训练 2 个轮次，批量大小为 128 个样本，权重衰减因子为 0.1。我们使用 384 的序列长度进行 WikiSQL (Zhong et al., 2017)，768 用于 MNLI (Williams et al., 2018)，以及 2048 用于 SAMSum (Gliwa et al., 2019)。我们为所有方法-数据集组合调整学习率。有关使用的超参数的更多详情，请参见 Section D.4。对于前缀嵌入调优，我们发现最优的 l_p 和 l_i 分别为 256 和 8，总共 $3.2M$ 可训练参数。我们使用 $l_p = 8$ 和 $l_i = 8$ 进行前缀层调优，具有 $20.2M$ 可训练参数，以获得整体最佳性能。我们为 LoRA 提供两个参数预算：4.7M（$r_q = r_v = 1$ 或 $r_v = 2$）和 37.7M（$r_q = r_v = 8$ 或 $r_q = r_k = r_v = r_o = 2$）。我们报告每次运行的最佳验证性能。GPT-3 实验中使用的训练超参数列在 Table 9中。

超参数	微调	预嵌入	预层	BitFit	$\text{Adapter}^{\text{H}}$	LoRA
优化器	AdamW					
批量大小	128					
训练轮数	2					
预热令牌数	250,000					
学习率调度	线性					
学习率	5.00E-06	5.00E-04	1.00E-04	1.6E-03	1.00E-04	2.00E-04

Table 9: 不同 GPT-3 适配方法使用的训练超参数。在调整学习率后，我们对所有数据集使用相同的超参数。

E 结合 LoRA 和前缀调优

LoRA 可以自然地与现有的基于前缀的方法相结合。在本节中，我们在 WikiSQL 和 MNLI 上评估 LoRA 和前缀调优变体的两种组合。**LoRA+ 前缀嵌入（LoRA+PE）**将 LoRA 与前缀嵌入调优相结合，我们插入 $l_p + l_i$ 个特殊标记，其嵌入被视为可训练参数。关于前缀嵌入调优的更多信息，请参见 Section 5.1。**LoRA+ 前缀层（LoRA+PL）**将 LoRA 与前缀层调优相结合。我们同样插入 $l_p + l_i$ 个特殊标记；但是，我们不让这些标记的隐藏表示自然演化，而是在每个 Transformer 块后用与输入无关的向量替换它们。因此，这些标记的嵌入和后续 Transformer 块的激活都被视为可训练参数。关于前缀层调优的更多信息，请参见 Section 5.1。在 Table 12 中，我们展示了 LoRA+PE 和 LoRA+PL 在 WikiSQL 和 MultiNLI 上的评估结果。首先，LoRA+PE 在 WikiSQL 上显著优于 LoRA 和前缀嵌入调优，这表明 LoRA 在某种程度上与前缀嵌入调优是正交的。在 MultiNLI 上，LoRA+PE 的组合并没有比 LoRA 表现得更好，可能是因为 LoRA 本身已经达到了接近人类基准线的性能。其次，我们注意到 LoRA+PL 即使有更多的可训练参数，也比 LoRA 表现稍差。我们将此归因于前缀层调优对学习率的选择非常敏感，因此使得 LoRA+PL 中 LoRA 权重的优化变得更加困难。

F 额外的实证实验

F.1 GPT-2 的额外实验

我们还按照 Li & Liang (2021) 的设置，在 DART (Nan et al., 2020) 和 WebNLG (Gardent et al., 2017) 上重复了我们的实验。结果如 Table 10所示。与我们在 E2E 自然语言生成挑战中报告的结果类似（在 Section 5中），在相同数量的可训练参数下，LoRA 的性能优于或至少与基于前缀的方法相当。

方法	可训练参数数量	DART		
		BLEU↑	MET↑	TER↓
GPT-2 中等				
微调	354M	46.2	**0.39**	**0.46**
$\text{Adapter}^{\text{L}}$	0.37M	42.4	0.36	0.48
$\text{Adapter}^{\text{L}}$	11M	45.2	0.38	**0.46**
FT^{Top2}	24M	41.0	0.34	0.56
前缀层	0.35M	46.4	0.38	**0.46**
LoRA	0.35M	$\mathbf{47.1}_{\pm.2}$	**0.39**	**0.46**
GPT-2 大型				
微调	774M	47.0	**0.39**	0.46
$\text{Adapter}^{\text{L}}$	0.88M	$45.7_{\pm.1}$	0.38	0.46
$\text{Adapter}^{\text{L}}$	23M	$47.1_{\pm.1}$	**0.39**	**0.45**
前缀层	0.77M	46.7	0.38	**0.45**
LoRA	0.77M	$\mathbf{47.5}_{\pm.1}$	**0.39**	**0.45**

Table 10: 不同适配方法在 DART 上的 GPT-2 结果。所有适配方法的 MET 和 TER 方差均小于 0.01。

F.2 GPT-3 的额外实验

我们在 Table 12中展示了在 GPT-3 上使用不同自适应方法的额外运行结果。重点在于识别性能和可训练参数数量之间的权衡。

方法	WebNLG								
	BLEU↑			MET↑			TER↓		
	U	S	A	U	S	A	U	S	A
	GPT-2 中等模型								
微调（354M）	27.7	**64.2**	46.5	.30	**.45**	.38	.76	**.33**	.53
适配器^{L}（0.37M）	45.1	54.5	50.2	.36	.39	.38	.46	.40	.43
适配器^{L}（11M）	**48.3**	60.4	54.9	**.38**	.43	**.41**	**.45**	.35	**.39**
FT^{Top2}（24M）	18.9	53.6	36.0	.23	.38	.31	.99	.49	.72
前缀（0.35M）	45.6	62.9	55.1	**.38**	.44	**.41**	.49	.35	.40
LoRA（0.35M）	$46.7_{\pm.4}$	$62.1_{\pm.2}$	$\mathbf{55.3}_{\pm.2}$	**.38**	.44	**.41**	.46	**.33**	**.39**
	GPT-2 大模型								
微调（774M）	43.1	65.3	55.5	.38	**.46**	.42	.53	.33	.42
适配器^{L}（0.88M）	$\mathbf{49.8}_{\pm.0}$	$61.1_{\pm.0}$	$56.0_{\pm.0}$	.38	.43	.41	**.44**	.35	.39
适配器^{L}（23M）	$49.2_{\pm.1}$	$64.7_{\pm.2}$	$\mathbf{57.7}_{\pm.1}$	**.39**	**.46**	**.43**	.46	.33	.39
前缀（0.77M）	47.7	63.4	56.3	**.39**	.45	.42	.48	.34	.40
LoRA（0.77M）	$48.4_{\pm.3}$	$64.0_{\pm.3}$	$57.0_{\pm.1}$	**.39**	.45	.42	.45	**.32**	**.38**

Table 11: GPT-2 不同自适应方法在 WebNLG 上的表现。我们进行的所有实验中，MET 和 TER 的方差均小于 0.01。”U” 表示未见类别，”S” 表示已见类别，”A” 表示 WebNLG 测试集中的所有类别。

F.3 低数据场景

为了评估不同自适应方法在低数据场景下的性能，我们从 MNLI 的完整训练集中随机抽样 100、1k 和 10k 训练样本，形成低数据 MNLI-n 任务。在Table 13中，我们展示了不同自适应方法在 MNLI-n 上的表现。令人惊讶的是，前缀嵌入和前缀层在 MNLI-100 数据集上表现极差，前缀嵌入的性能仅略高于随机猜测（37.6% vs. 33.3%）。前缀层的性能好于前缀嵌入，但仍然显著低于微调或 LoRA 在 MNLI-100 上的表现。随着训练样本数量增加，基于前缀的方法与 LoRA/微调之间的差距变小，这可能表明前缀方法不适合 GPT-3 中的低数据任务。LoRA 在 MNLI-100 和 MNLI-Full 上均取得了优于微调的性能，在 MNLI-1k 和 MNLI-10K 上考虑随机种子引起的（±0.3）方差，其结果是可比的。

不同自适应方法在 MNLI-n 上的训练超参数在Table 14中报告。我们在 MNLI-100 集上对前缀层使用了较小的学习率，因为较大的学习率下训练损失不会下降。

G 子空间相似性测量

在本文中，我们使用度量 $\phi(A, B, i, j) = \psi(U_A^i, U_B^j) = \frac{\|U_A^{i\top} U_B\|_F^2}{\min\{i,j\}}$ 来测量两个列正交矩阵 $U_A^i \in \mathbb{R}^{d\times i}$ 和 $U_B^j \in \mathbb{R}^{d\times j}$ 之间的子空间相似性，这两个矩阵通过取矩阵 A 和 B 的左奇异矩阵的列得到。我们指出，这种相似性实际上是标准投影度量（测量子空间之间距离）的反向 Ham & Lee (2008)。

具体来说，设 $U_A^{i\top} U_B^j$ 的奇异值为 $\sigma_1, \sigma_2, \cdots, \sigma_p$，其中 $p = \min\{i, j\}$。我们知道投影度量 Ham & Lee (2008) 定义为：

$$d(U_A^i, U_B^j) = \sqrt{p - \sum_{i=1}^{p} \sigma_i^2} \in [0, \sqrt{p}]$$

方法	超参数	可训练参数数量	WikiSQL	MNLI-m
微调	-	175B	73.8	89.5
前缀嵌入	$l_p = 32, l_i = 8$	0.4 M	55.9	84.9
	$l_p = 64, l_i = 8$	0.9 M	58.7	88.1
	$l_p = 128, l_i = 8$	1.7 M	60.6	88.0
	$l_p = 256, l_i = 8$	3.2 M	63.1	88.6
	$l_p = 512, l_i = 8$	6.4 M	55.9	85.8
前缀层	$l_p = 2, l_i = 2$	5.1 M	68.5	89.2
	$l_p = 8, l_i = 0$	10.1 M	69.8	88.2
	$l_p = 8, l_i = 8$	20.2 M	70.1	89.5
	$l_p = 32, l_i = 4$	44.1 M	66.4	89.6
	$l_p = 64, l_i = 0$	76.1 M	64.9	87.9
适配器$^{\text{H}}$	$r = 1$	7.1 M	71.9	89.8
	$r = 4$	21.2 M	73.2	91.0
	$r = 8$	40.1 M	73.2	91.5
	$r = 16$	77.9 M	73.2	91.5
	$r = 64$	304.4 M	72.6	91.5
LoRA	$r_v = 2$	4.7 M	73.4	**91.7**
	$r_q = r_v = 1$	4.7 M	73.4	91.3
	$r_q = r_v = 2$	9.4 M	73.3	91.4
	$r_q = r_k = r_v = r_o = 1$	9.4 M	74.1	91.2
	$r_q = r_v = 4$	18.8 M	73.7	91.3
	$r_q = r_k = r_v = r_o = 2$	18.8 M	73.7	**91.7**
	$r_q = r_v = 8$	37.7 M	73.8	**91.6**
	$r_q = r_k = r_v = r_o = 4$	37.7 M	74.0	**91.7**
	$r_q = r_v = 64$	301.9 M	73.6	91.4
	$r_q = r_k = r_v = r_o = 64$	603.8 M	73.9	91.4
LoRA+PE	$r_q = r_v = 8, l_p = 8, l_i = 4$	37.8 M	75.0	91.4
	$r_q = r_v = 32, l_p = 8, l_i = 4$	151.1 M	**75.9**	91.1
	$r_q = r_v = 64, l_p = 8, l_i = 4$	302.1 M	**76.2**	91.3
LoRA+PL	$r_q = r_v = 8, l_p = 8, l_i = 4$	52.8 M	72.9	90.2

Table 12: 不同自适应方法在 WikiSQL 和 MNLI 上的超参数分析。随着可训练参数数量的增加，前缀嵌入（PrefixEmbed）和前缀层（PrefixLayer）的性能下降，而 LoRA 的性能保持稳定。性能以验证准确率衡量。

方法	MNLI(m)-100	MNLI(m)-1k	MNLI(m)-10k	MNLI(m)-392K
GPT-3（微调）	60.2	**85.8**	88.9	89.5
GPT-3（前缀嵌入）	37.6	75.2	79.5	88.6
GPT-3（前缀层）	48.3	82.5	85.9	89.6
GPT-3（LoRA）	**63.8**	85.6	**89.2**	**91.7**

Table 13: 使用 GPT-3 175B 在 MNLI 子集上的不同方法验证准确率。MNLI-n 描述了具有 n 个训练样本的子集。我们使用完整的验证集进行评估。LoRA 展示了相比其他方法更有利的样本效率。

而我们的相似性定义为：

$$\phi(A, B, i, j) = \psi(U_A^i, U_B^j) = \frac{\sum_{i=1}^{p} \sigma_i^2}{p} = \frac{1}{p}\left(1 - d(U_A^i, U_B^j)^2\right)$$

超参数	自适应方法	MNLI-100	MNLI-1k	MNLI-10K	MNLI-392K
优化器	-	AdamW			
预热标记	-	250,000			
学习率调度	-	线性			
批量大小	-	20	20	100	128
训练轮数	-	40	40	4	2
学习率	微调	5.00E-6			
	前缀嵌入	2.00E-04	2.00E-04	4.00E-04	5.00E-04
	前缀层	5.00E-05	5.00E-05	5.00E-05	1.00E-04
	LoRA	2.00E-4			
自适应特定	前缀嵌入 l_p	16	32	64	256
	前缀嵌入 l_i	8			
	前缀调优	$l_p = l_i = 8$			
	LoRA	$r_q = r_v = 8$			

Table 14: MNLI(m)-n 上 GPT-3 不同自适应方法使用的超参数。

这种相似性满足：如果 U_A^i 和 U_B^j 共享相同的列空间，则 $\phi(A, B, i, j) = 1$。如果它们完全正交，则 $\phi(A, B, i, j) = 0$。否则，$\phi(A, B, i, j) \in (0, 1)$。

H 低秩矩阵的额外实验

我们呈现了对低秩更新矩阵的进一步研究结果。

H.1 LoRA 模块间的相关性

参见Figure 6和Figure 7，了解Figure 3和Figure 4中的结果如何推广到其他层。

H.2 r 对 GPT-2 的影响

我们在 GPT-2 上重复了关于 r 的影响的实验（参见**??**）。以 E2E NLG 挑战数据集为例，我们报告了在训练 26,000 步后，不同 r 值下的验证损失和测试指标。结果见Table 15。对于 GPT-2 中等模型，最优秩取决于所用指标，在 4 到 16 之间，这与 GPT-3 175B 类似。需要注意的是，模型大小与自适应的最优秩之间的关系仍是一个开放性问题。

H.3 W 与 ΔW 之间的相关性

参见Figure 8，了解不同 r 下 W 和 ΔW 之间的归一化子空间相似性。

再次强调，ΔW 不包含 W 的顶部主奇异方向，因为 ΔW 中前 4 个方向与 W 中前 10% 方向的相似性几乎不超过 0.2。这证明了 ΔW 包含那些在 W 中未被强调的" 任务特定" 方向。

一个有趣的后续问题是：为了模型自适应能够良好工作，我们需要将这些任务特定方向放大到什么程度？

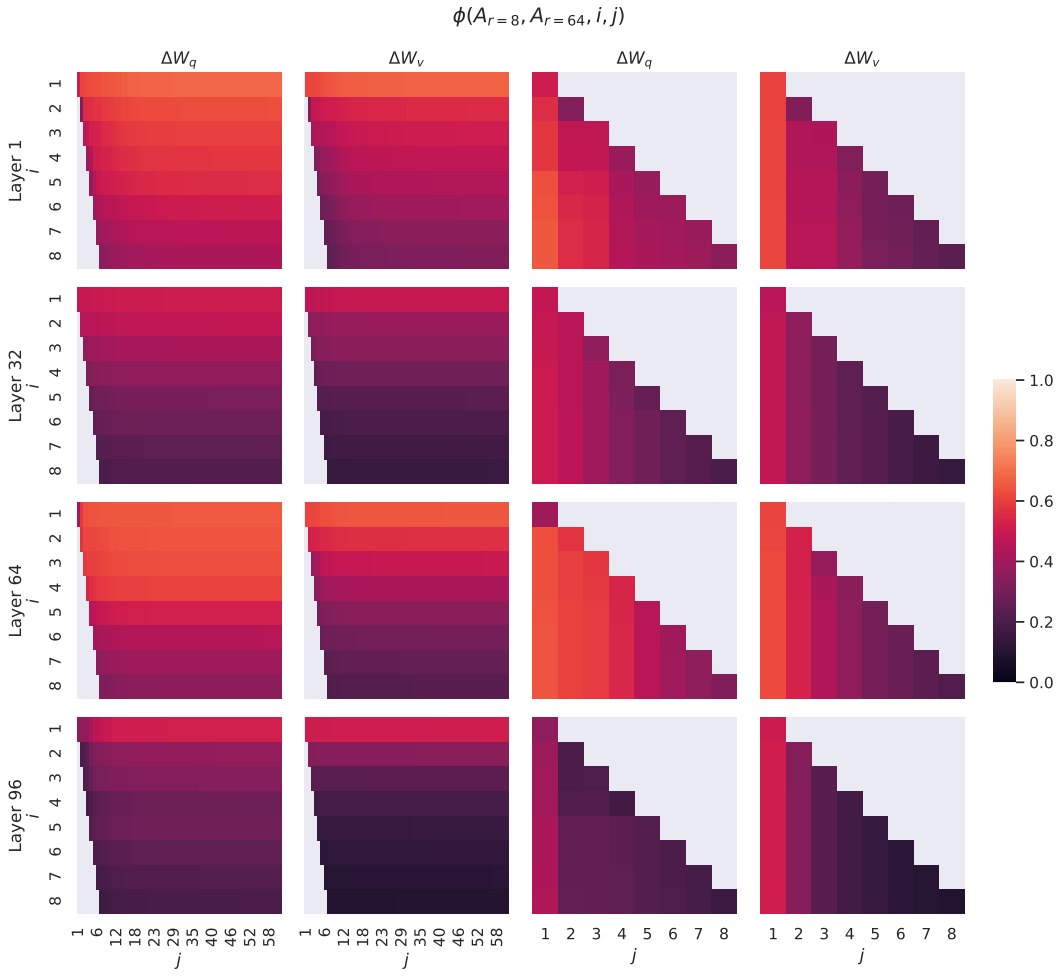

Figure 6: 在 96 层 Transformer 的第 1、32、64 和 96 层中，$A_{r=8}$ 和 $A_{r=64}$ 的列向量之间的归一化子空间相似性，针对 ΔW_q 和 ΔW_v。

H.4 放大因子

人们自然可以考虑特征放大因子作为比率 $\frac{\|\Delta W\|_F}{\|U^\top W V^\top\|_F}$，其中 U 和 V 是 ΔW 的奇异值分解中的左、右奇异矩阵。(回顾 $UU^\top WV^\top V$ 给出了 W 在 ΔW 跨度的子空间上的" 投影"。)

直观地说，当 ΔW 主要包含任务特定方向时，这个量度量了 ΔW 对这些方向的放大程度。如Section 6.1所示，对于 $r = 4$，这个放大因子高达 20。换句话说，在预训练模型 W 的整个特征空间中，通常每一层有四个特征方向需要被放大 20 倍，以便为下游特定任务实现我们报告的准确率。并且，对于每个不同的下游任务，人们应该预期会放大一组完全不同的特征方向。

然而，对于 $r = 64$，这个放大因子仅约为 2，这意味着在 $r = 64$ 的 ΔW 中学习到的大多数方向并未被太多放大。这并不令人意外，事实上再次证明了表示" 任务特定方向"（因此用于模型自适应）所需的内在秩是低的。相比之下，ΔW 的秩-4 版本（对应 $r = 4$）中的那些方向被放大了 20 倍。

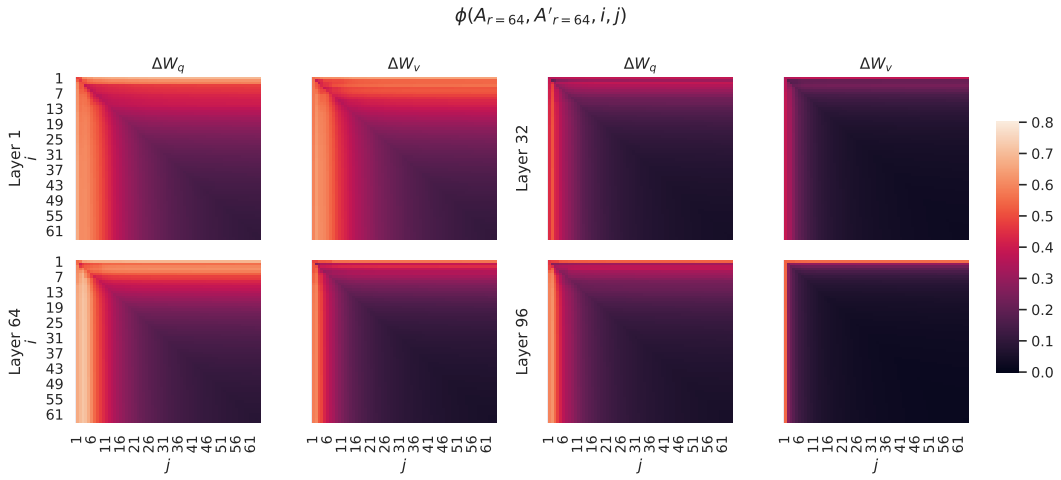

Figure 7: 在 96 层 Transformer 的第 1、32、64 和 96 层中，来自两个随机种子运行的 $A_{r=64}$ 的列向量之间的归一化子空间相似性，针对 ΔW_q 和 ΔW_v。

秩 r	验证损失	BLEU	NIST	METEOR	ROUGE_L	CIDEr
1	1.23	68.72	8.7215	0.4565	0.7052	2.4329
2	1.21	69.17	8.7413	0.4590	0.7052	2.4639
4	1.18	**70.38**	**8.8439**	**0.4689**	0.7186	**2.5349**
8	1.17	69.57	8.7457	0.4636	**0.7196**	2.5196
16	**1.16**	69.61	8.7483	0.4629	0.7177	2.4985
32	**1.16**	69.33	8.7736	0.4642	0.7105	2.5255
64	**1.16**	69.24	8.7174	0.4651	0.7180	2.5070
128	**1.16**	68.73	8.6718	0.4628	0.7127	2.5030
256	**1.16**	68.92	8.6982	0.4629	0.7128	2.5012
512	**1.16**	68.78	8.6857	0.4637	0.7128	2.5025
1024	1.17	69.37	8.7495	0.4659	0.7149	2.5090

Table 15: 使用 GPT-2 中等模型在 E2E NLG 挑战数据集上，不同秩 r 下的验证损失和测试集指标。与 GPT-3 中 $r = 1$ 足以应对许多任务不同，这里验证损失在 $r = 16$ 时达到峰值，BLEU 在 $r = 4$ 时达到峰值，这表明 GPT-2 中等模型在自适应方面具有与 GPT-3 175B 相似的内在秩。需要注意的是，我们的部分超参数是在 $r = 4$ 上调整的，这与另一个基准的参数计数相匹配，因此可能对其他 r 值不是最优的。

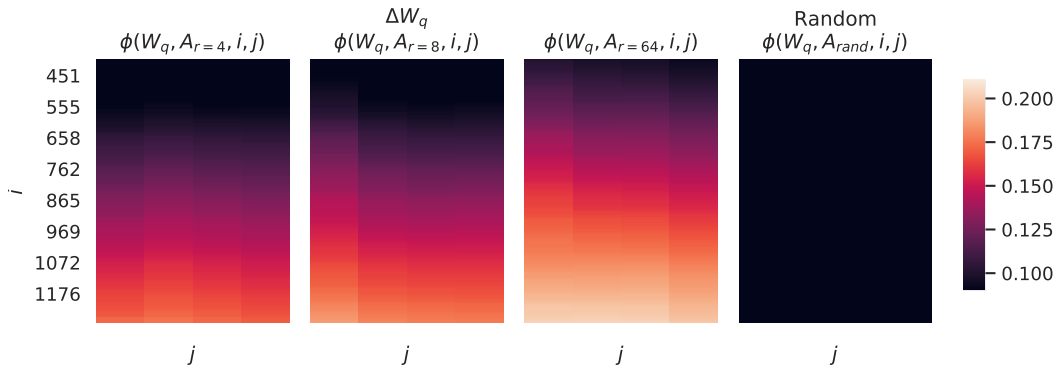

Figure 8: 不同 r 和随机基线下 W_q 的奇异方向与 ΔW_q 的奇异方向之间的归一化子空间相似性。ΔW_q 放大了 W 中重要但未被强调的方向。具有较大 r 的 ΔW 倾向于捕捉 W 中已经强调的更多方向。