

LoRA: 低等级适应的大网络模型

Edward Hu* Yelong Shen* Phillip WallisZeyuan Allen-
ZhuYuanzhi Li Shean Wang Lu Wang Weizhu Chen
微软公司
{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(第2版)

摘要

自然语言处理的一个重要范式是在一般领域数据上进行大规模预训练，然后适应特定任务或领域。随着我们预训练的模型越来越大，重新训练所有模型参数的全面微调就变得不那么可行。以 GPT-3 175B 为例，部署微调模型的独立实例（每个实例有 175B 个参数）的成本过高。我们提出了低秩适应（Low-Rank Adaptation，简称 LoRA）技术，它可以冻结预先训练好的模型权重，并将可训练的秩分解矩阵注入 Transformer 架构的每一层，从而大大减少下游任务的可训练参数数量。与使用 Adam 微调的 GPT-3 175B 相比，LoRA 可将可训练参数的数量减少 10,000 倍，GPU 内存需求减少 3 倍。LoRA 在 RoBERTa、DeBERTa、GPT-2 和 GPT-3 上的模型质量表现与微调相当或更好，尽管可训练参数更少、训练吞吐量更高，而且与适配器不同，没有额外的推理延迟。我们还对语言模型适配中的等级缺陷进行了实证调查，从而揭示了 LoRA 的功效。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> 上提供了我们对 RoBERTa、DeBERTa 和 GPT-2 的实现和模型检查点。

1 引言

自然语言处理中的许多应用都依赖于将一个大规模的预训练语言模型适应多个下游应用。这种适应通常通过微调来完成，微调会更新预训练模型的所有参数。微调的最大缺点是新模型包含的参数与原始模型一样多。由于更大的模型每隔几个月就要

训练一次，这对于 GPT-2（Radford 等人，b）或 RoBERTa large（Liu 等人，2019）来说只是“不便”，而对于拥有 1,750 亿个可训练参数的 GPT-3（Brown 等人

，2020）来说，则是一个关键的部署挑战。¹

许多人试图通过只调整某些参数或针对新任务学习外部模块来缓解这一问题。这样，我们只需在每个任务的预训练模型之外存储和加载少量任务特定参数，从而大大提高了部署时的运行效率。然而，现有技术

*平等贡献。

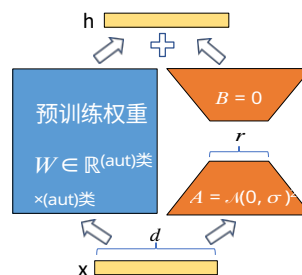


图 1：我们的重构模型。我们只训练 A 和 B 。

⁰与 V1 相比，该草案包括更好的基线、GLUE 实验以及更多关于适配器延迟的内容。

¹虽然 GPT-3 175B 通过几次学习就能达到非同一般的性能，但如附录 A 所示，微调能显著提高其性能。

这些方法往往通过扩展模型深度或减少模型的可用序列长度来引入推理延迟 (Houlsby 等人, 2019 年; Rebuffi 等人, 2017 年) (李和梁, 2021 年; Lester 等人, 2021 年; Hambardzumyan 等人, 2020 年; 刘等人, 2021 年) (第 3 节)。更重要的是, 这些方法往往无法与微调基线相匹配, 这就需要在效率和模型质量之间进行权衡。

我们从 Li 等人 (2018a) 和 Aghajanyan 等人 (2020) 的研究中得到启发, 他们的研究表明, 学习到的过参数化模型实际上驻留在较低的内在维度上。我们假设, 模型适应过程中权重的变化也具有较低的 "内在秩", 从而提出了我们的低秩适应 (Low-Rank Adaptation, LoRA) 方法。如图 1 所示, LoRA 允许我们通过优化密集层在适应过程中的变化的秩分解矩阵来间接训练神经网络中的一些密集层, 同时保持预训练的权重冻结。以 GPT-3 175B 为例, 我们发现即使全秩 (即 d) 高达 12,288 时, 也只需要很低的秩 (即图 1 中的 r , 可以是 1 或 2) 就足够了, 这使得 LoRA 既节省存储空间, 又节省计算时间。

LoRA 具有几个关键优势。

- 预先训练好的模型可以共享, 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, 并通过替换图 1 中的矩阵 A 和 B 来高效地切换任务, 从而大大降低了存储需求和任务切换次数。
- 在使用自适应优化器时, LoRA 使训练更加高效, 并将硬件门槛降低了 3 倍, 因为我们不需要计算梯度或维持大多数参数的优化器状态。相反, 我们只需优化注入的、更小的低秩矩阵。
- 我们的简单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, 与完全微调的模型相比, *不会带来推理延迟*。
- LoRA 与许多先前的方法是正交的, 并可与许多方法 (如前缀调整) 相结合。我们在附录 E 中提供了一个例子。

术语和约定 我们经常提到变换器架构, 并对其维度使用传统术语。我们将 Transformer 层的输入和输出维度大小称为 d_{model} 。我们使用 W_q , W_k , W_v , 和 W_o 来指自我关注模块中的查询/键/值/输出投影矩阵。 W 或 W_0 指预先训练好的权重矩阵, ΔW 指适应过程中累积的梯度更新。我们用 r 表示 LoRA 模块的秩。我们遵循 (Vaswani 等人, 2017 年; Brown 等人, 2020 年) 的惯例, 使用 Adam (Loshchilov & Hutter, 2019 年; Kingma & Ba, 2017 年) 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

虽然我们的建议与训练目标无关，但我们将语言建模作为我们的激励用例。下面简要介绍语言建模问题，特别是在特定任务提示下条件概率的最大化问题。

假设我们得到了一个预先训练好的自回归语言模型 $P_\Phi(y|x)$ ，其参数为 Φ 。例如， $P_\Phi(y|x)$ 可以是一个通用的多任务学习器，如基于 Transformer 架构（Vaswani 等人，2017 年）的 GPT（Radford 等人，2016；Brown 等人，2020 年）。考虑将此预训练模型适应于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)。每个下游任务都由上下文-目标对的训练数据集表示： $\mathcal{D} = (x_i, y_i)_{i=1, \dots, N}$ ，其中 x_i 和 y_i 都是标记序列。例如，在 NL2SQL 中， x_i 是一个自然语言查询， y_i 是其相应的 SQL 命令；对于摘要， x_i 是一篇文章的内容， y_i 是其摘要。

在全面微调过程中，模型初始化为预先训练的权重 Φ_0 ，然后更新为 $\Phi_0 + \Delta\Phi$ 。通过反复跟踪梯度来最大化条件语言建模目标：

$$\underset{\Phi}{\text{最大}} \sum_{(x,y) \in \mathcal{Z}} \sum_{t=1}^{|y|} \log (P_{\Phi}(y_t | x, y)_{<t}) \quad (1)$$

完全微调的主要缺点之一是，对于每个下游任务，我们都要学习一组不同的参数 $\Delta\Phi$ ，其维度 $\Delta\Phi$ 等于 Φ_0 。因此，如果预训练的模型很大（如 GPT-3 的 Φ_0 175 Billion），那么存储和部署许多微调模型的独立实例即使可行，也会很有挑战性。

在本文中，我们采用了一种参数效率更高的方法，即特定于任务的参数增量 $\Delta\Phi = \Delta\Phi(\Theta)$ 由一组更小的参数 Θ 进一步编码，其中 $|\Theta| \ll |\Phi_0|$ 。因此，寻找 $\Delta\Phi$ 的任务就变成了优化 Θ ：

$$\underset{\Theta}{\text{最大}} \sum_{(x,y) \in \mathcal{Z}} \sum_{t=1}^{|y|} \log p_{\Phi_0 + \Delta\Phi(\Theta)}(y_t | x, y)_{<t} \quad (2)$$

在接下来的章节中，我们将提出使用低秩表示法来编码既节省计算又节省内存的 $\Delta\Phi$ 。当预训练模型为 GPT-3 175B 时，训练参数 Θ 的数量可小至 $|\Phi_0|$ 的 0.01%。

3 现有解决方案还不够好吗？

我们要解决的问题并不新鲜。自迁移学习诞生以来，已有数十项研究试图提高模型适应的参数和计算效率。关于其中一些著名的研究，请参见第 6 章。以语言建模为例，在高效适配方面有两种突出的策略：增加适配层（Houlsby 等人，2019 年；Rebuffi 等人，2017 年；Pfeiffer 等人，2021 年；Rućkle 等人，2020 年）或优化输入层激活的某些形式（Li & Liang，2021 年；Lester 等人，2021 年；Hambarzumyan 等人，2020 年；Liu 等人，2021 年）。然而，这两种策略都有其局限性，尤其是在大规模和对延迟敏感的生产场景中。

适配器层引入推理延迟 适配器有很多变体。我们重点关注 Houlsby 等人（2019 年）的原始设计和 Lin 等人（2020 年）的最新设计，前者每个变换器块有两个适配器层，后者每个块只有一个，但有一个额外的层规范（Ba 等人，2016 年）。虽然我们可以通过修剪层或利用多任务设置来减少整体延迟（Rućkle 等人，2020 年；Pfeiffer 等人，2021 年），但没有直接的方法绕过适配器层中的额外计算。这似乎不是一个问题，因为适配器层的设计参数很少（有时小于原始模型的 1%），只有一个小的瓶颈层，这限制了它们可以增加的 FLOP。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。这在批量规模通常只有一个的在线推理设置中造成了差异。在没有模型并行性的通用场景中，例如在单个 GPU 的 GPT-2（Radford 等人，b）介质上运行推理，我们发现使用适配器时，即使瓶颈维度很小，延迟也会明显增加（表 1）。

当我们需要像 Shoeybi 等人（2020 年）和 Lepikhin 等人（2020 年）那样对模型进行分片时，这个问题就会变得更加严重，因为额外的深度需要更多的 GPU 同步操作，如

AllReduce 和 Broadcast，除非我们多次冗余存储适配器参数。

直接优化提示困难重重 以前缀调整 (Li & Liang, 2021 年) 为例，另一个方向面临着不同的挑战。我们观察到，前缀调整很难优化，而且其性能在可训练参数上的变化是非单调的，这证实了原论文中的类似观察。更根本的是，预留部分序列长度用于适应必然会减少可用于处理下游任务的序列长度，我们怀疑这使得调整提示的性能低于其他方法。我们将对任务性能的研究推迟到第 5 节。

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

表 1: GPT-2 介质中单次前向传输的延迟, 以毫秒为单位, 100 次试验的平均值。我们使用的是英伟达 Quadro RTX8000。"Θ" 表示适配器层中可训练参数的数量。适配器^L 和适配器^H 是适配器调整的两种变体, 我们将在第 5.1 节进行介绍。在短序列长度的在线场景中, 适配器层带来的推理延迟可能非常显著。参见附录 B 中的完整研究。

4 我们的方法

我们将介绍 LoRA 的简单设计及其实际优势。这里概述的原则适用于深度学习模型中的任何密集层, 不过我们在实验中只关注 Transformer 语言模型中的某些权重, 并将其作为激励用例。

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全秩。在适应特定任务时, Aghajanyan 等人 (2020 年) 的研究表明, 预训练语言模型的 "本征维度" 较低, 即使随机投影到较小的子空间, 仍能高效学习。受此启发, 我们假设权重更新在适应过程中也具有较低的本征维度。对于一个预先训练好的权重矩阵 $W_0 \in \mathbb{R}^{d \times k}$, 我们用一个低秩去组成 $W_0 + \Delta W = W_0 + BA$ 来限制它的更新, 其中 $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, 秩为 $r = \min(d, k)$ 。在训练过程中, W_0 被冻结, 不会接受梯度更新, 而 A 和 B 包含可训练参数。请注意, W_0 和 $\Delta W = BA$ 都与相同的输入相乘, 它们各自的输出向量按坐标相加。对于 $h = W_0 x$, 我们修改后的前向传递结果为

$$h = W_0 x + \Delta W x = W_0 x + BAx \quad (3)$$

图 1 展示了我们的重新参数化过程。我们对 A 使用随机高斯初始化, 对 B 使用零初始化, 因此在训练开始时 $\Delta W = BA$ 为零。然后, 我们通过 α 对 $\Delta W x$ 进行缩放, 其中 α 是 r 中的常数。在使用亚当进行优化时, 如果我们对初始化进行适当缩放, 调整 α 与调整学习率大致相同。因此, 我们只需将 α 设为我们尝试的第一个 r , 而无需调整。当我们改变 r 时, 这种缩放有助于减少重新调整超参数的需要 (Yang & Hu, 2021 年)。

全面微调的一般化。微调的一种更普遍的形式是对预训练参数的子集进行训练。LoRA 更进一步, 在适应过程中不要求权重矩阵的累积梯度更新具有全等级。这意味着在对所有权

重矩阵应用 LoRA 并训练所有偏置时²时，通过将 LoRA 的秩 r 设置为预先训练的权重矩阵的秩，我们就能大致恢复完全微调的表达能力。换句话说，随着可训练参数数量的增加³时，训练 LoRA 大致收敛于训练原始模型，而基于适配器的方法收敛于 MLP，基于前缀的方法收敛于无法接受长输入序列的模型。

无额外推理延迟。在生产中部署时，我们可以显式计算并存储 $W = W_0 + BA$ ，然后像往常一样执行推理。请注意， W_0 和 BA 都在 $\mathbb{R}^{d \times k}$ 中。当我们需要切换到另一个下游任务时，可以通过减去 BA 并添加不同的 BA'' 来恢复 W_0 ，这是一个内存开销很小的快速操作。重要的是，这

²与权重相比，这些参数的数量可以忽略不计。

³在适应艰巨任务时，这是不可避免的。

这保证了我们在推理过程中不会引入比构建微调模型更多的延迟。

4.2 将 LoRA 应用于变压器

原则上，我们可以将 LoRA 应用于神经网络中的任意权重矩阵子集，以减少可训练参数的数量。在 Transformer 架构中，自注意模块中有四个权重矩阵 (W_q, W, W_{kv}, W_o)，MLP 模块中有两个。我们将 W_q (或 W_k, W_v) 视为维度为 $d_{model} \times d_{model}$ 的单一矩阵，尽管输出维度通常被切成注意力头。出于简单性和参数效率的考虑，我们的研究**仅限于调整下游任务的注意力权重**，并冻结 MLP 模块（使其不在下游任务中接受训练）。我们将在第 7.1 节中进一步研究在转换器中调整不同类型注意力权重矩阵的效果。关于调整 MLP 层、LayerNorm 层和偏置的实证研究，我们将留待今后的工作中进行。

Practical Benefits and Limitations. The most significant benefit comes from the reduction in memory and storage usage. For a large Transformer trained with Adam, we reduce that VRAM usage by up to $2/3$ if $r \ll d_{model}$ as we do not need to store the optimizer states for the frozen parameters. On GPT-3 175B, we reduce the VRAM consumption during training from 1.2TB to 350GB. With $r = 4$ and only the query and value projection matrices being adapted, the checkpoint size is reduced by roughly $10,000\times$ (from 350GB to 35MB)⁴. This allows us to train with significantly fewer GPUs and avoid I/O bottlenecks. Another benefit is that we can switch between tasks while deployed at a much lower cost by only swapping the LoRA weights as opposed to all the parameters. This allows for the creation of many customized models that can be swapped in and out on the fly on machines that store the pre-trained weights in VRAM. We also observe a 25% speedup during training on GPT-3 175B compared to full fine-tuning⁵ as we do not need to calculate the gradient for the vast majority of the parameters.

LoRA 也有其局限性。例如，如果选择将 A 和 B 吸收到 W 中以消除额外的推理延迟，那么在一次前向传递中批量输入具有不同 A 和 B 的不同任务就不是一件简单的事。不过，在延迟并不重要的情况下，也可以不合并权重，动态选择 LoRA 模块用于批量样本。

5 经验性实验

我们评估了 LoRA 在 RoBERTa (Liu 等人, 2019 年)、De-BERTa (He 等人, 2021 年) 和 GPT-2 (Radford 等人, b) 上的下游任务性能，然后将其扩展到 GPT-3 175B (Brown 等人, 2020 年)。我们的实验涵盖了从自然语言理解 (NLU) 到生成 (NLG) 的各种任务。具体来说，我们在 GLUE (Wang 等人, 2019 年) 基准上对 RoBERTa 和 DeBERTa 进行了评估。我们按照 Li & Liang (2021) 在 GPT-2 上的设置进行了直接比较，并在 GPT-3 上添加了 WikiSQL (Zhong 等人, 2017) (NL 到 SQL 查询) 和 SAMSum (Gliwa 等人, 2019) (对话摘要) 进行大规模实验。有关我们使用的数据集的更多详情，请参阅附录 C。我们使用英伟达™ (NVIDIA®) Tesla V100 进行所有实验。

5.1 基线

为了与其他基线进行广泛比较，我们复制了先前工作中使用的设置，并尽可能重复使用其报告的数字。不过，这意味着某些基线可能只出现在某些实验中。

微调 (FT) 是一种常见的适应方法。在微调过程中，模型被初始化为预先训练好的权重和偏置，所有模型参数都会进行梯度更新。我们在 GPT-2 的前期工作 (Li & Liang, 2021 年) 中报告了一个这样的基线，它只对最后两层进行适应性调整 (FT^{Top^2})。

⁴在部署过程中，我们仍然需要 350GB 的模型；不过，存储 100 个适配模型只需要 $350\text{GB} + 35\text{MB} * 100 \approx 354\text{GB}$ ，而不是 $100 * 350\text{GB} \approx 35\text{TB}$ 。

⁵对于 GPT-3 175B，完全微调的训练吞吐量为每 V100 GPU 32.5 tokens/s；在模型并行化权重碎片数量相同的情况下，LoRA 的吞吐量为每 V100 GPU 43.1 tokens/s。

模式与方法	# 可培训 参数	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B 平均 值	
RoBbase (FT)*	125.0M	87.6	94.8	90.2	63.6	92.8	91.9	78.7	91.2	86.4
RoBbase (BitFit)*	0.1M	84.7	93.7	92.7	62.0	91.8	84.0	81.5	90.8	85.2
RoBbase (Adpt ^D)*	0.3M	87.1±0	94.2±1	88.5±1.1	60.8±4	93.1±1	90.2±0	71.5±2.7	89.7±3	84.4
RoBbase (Adpt ^D)*	0.9M	87.3±1	94.7±3	88.4±1	62.6±9	93.0±2	90.6±0	75.9±2.2	90.3±1	85.4
RoBbase (LoRA)	0.3M	87.5±3	95.1±2	89.7±7	63.4±1.2	93.3±3	90.8±1	86.6±7	91.5±2	87.2
RoBlarge (F ¹)*	355.0M	90.2	96.4	90.9	68.0	94.7	92.2	86.6	92.4	88.9
RoBlarge (LoRA)	0.8M	90.6±2	96.2±5	90.9±1.2	68.2±1.9	94.9±3	91.6±1	87.4±2.5	92.6±2	89.0
RoBlarge (Adpt ^P)†	3.0M	90.2±3	96.1±3	90.2±7	68.3±1.0	94.8±2	91.9±1	83.8±2.9	92.1±7	88.4
RoBlarge (Adpt ^P)†	0.8M	90.5±3	96.6±2	89.7±1.2	67.8±2.5	94.8±3	91.7±2	80.1±2.9	91.9±4	87.9
RoBlarge (Adpt ^H)†	6.0M	89.9±5	96.2±3	88.7±2.9	66.5±4.4	94.7±2	92.1±1	83.4±1.1	91.0±1.7	87.8
RoBlarge (Adpt ^H)†	0.8M	90.3±3	96.3±5	87.7±1.7	66.3±2.0	94.7±2	91.5±1	72.9±2.9	91.5±5	86.4
RoBlarge (LoRA)†	0.8M	90.6±2	96.2±5	90.2±1.0	68.2±1.9	94.8±3	91.6±2	85.2±1.1	92.3±5	88.6
DeBXXL (F ¹)*	1500.0M	91.8	97.2	92.0	72.0	96.0	92.7	93.9	92.9	91.1
DeBXXL (LoRA)	4.7M	91.9±2	96.9±2	92.6±6	72.4±1.1	96.0±1	92.9±1	94.9±4	93.0±2	91.3

表 2：在 GLUE 基准上采用不同适配方法的 RoBERTa_{base}、RoBERTa_{large} 和 DeBERTa_{XXL}。我们报告了 MNLI 的总体（匹配和不匹配）准确率、CoLA 的 Matthew 相关性、STS-B 的 Pearson 相关性以及其他任务的准确率。所有指标都是越高越好。* 表示运行配置与 Houlsby 等人（2019）类似，以便进行公平比较。

纯偏差或 BitFit 是一种基线，即我们只训练偏差向量，而冻结其他一切。与此同时，BitFit 也对这一基线进行了研究（Zaken 等人，2021 年）。

前缀嵌入调整（PreEmbed） 会在输入词块中插入特殊词块。这些特殊字符具有可训练的词嵌入，通常不在模型的词汇表中。将这些标记放在哪里会对性能产生影响。我们将重点放在 "前缀" 和 "后缀" 上，前者是在提示符前添加此类标记，后者是在提示符后添加此类标记。我们用 l_p （respect. l_i ）表示前缀（respect. infix）标记的数量。可训练参数的数量为 $|\theta| = d_{model} \times (l_p + l_i)$ 。

前缀层调整（PreLayer） 是前缀嵌入调整的扩展。我们不只是学习某些特殊字符的词嵌入（或等同于嵌入层之后的激活），而是学习每个变换层之后的激活。前几层计算出的激活值会被可训练的激活值取代。因此，可训练参数的数量为 $|\theta| = L \times d_{model} \times (l_p + l_i)$ ，其中 L 是变换器层数。

Houlsby 等人（2019）提出的**适配器调整**在自我注意模块（和 MLP 模块）与后续残差连接之间插入适配器层。在适配器层中有两个完全连接的层，中间有一个非线性偏置。我们称这种原始设计为**适配器^H**。最近，Lin 等人（2020 年）提出了一种更有效的设计，即适配器层仅应用于 MLP 模块之后和 LayerNorm 之后。我们称之为**适配器^L**。这与 Pfeiffer 等人

(2021 年) 提出的另一种设计非常相似, 我们称之为 **Adapter^P**。我们还包括另一种称为 AdapterDrop 的基线 (Rücklé et al., 2020), 它放弃了一些适配器层以提高效率 (**Adapter^D**)。我们尽可能引用先前作品中的数字, 以最大限度地增加与之比较的基线数量; 它们位于第一列中带有星号 (*) 的行中。

在所有情况下, 我们有 $|\theta| = \hat{L}_{Adpt} \times (2 \times d_{model} \times r + r + d_{model}) + 2 \times \hat{L}_{LN} \times d_{model}$, 其中 \hat{L}_{Adpt}

是适配器层的数量, L_{LN} 是可训练的 LayerNorms 的数量 (例如, 在适配器中)。

LoRA 在现有权重矩阵的基础上增加了可训练的秩分解矩阵对。如第 4.2 节所述, 为简单起见, 我们在大多数实验中只将 LoRA 应用于 W_q 和 W_v 。可训练参数的数量由等级 r 和原始权重的形状决定:

$|\theta| = 2 \times \hat{L}_{LoRA} \times d_{model} \times r$, 其中 \hat{L}_{LoRA} 是我们应用 LoRA 的权重矩阵数。

模式与方法	# 可培训参数	E2E 无法律约束力文书挑战赛				
		BLEU	NIST	金属	红-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M (适配器 ^L)*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M (适配器 ^L)*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M (适配器) ^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 米 (英尺 ^{Top2}) *	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (前置层) *	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	70.4\pm.1	8.85\pm.02	46.8\pm.2	71.8\pm.1	2.53\pm.02
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (适配器) ^L	0.88M	69.1 \pm .1	8.68 \pm .03	46.3 \pm .0	71.4 \pm .2	2.49\pm.0
GPT-2 L (适配器) ^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 (前置层) *	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	70.4\pm.1	8.89\pm.02	46.8\pm.2	72.0\pm.2	2.47 \pm .02

表 3：在 E2E NLG 挑战赛中采用不同适应方法的 GPT-2 中型 (M) 和大型 (L)。对于所有指标，越高越好。在可训练参数相当或更少的情况下，LoRA 优于几种基线方法。我们进行的实验显示了置信区间。* 表示先前工作中公布的数字。

5.2 RoBERTa 基地/大型

RoBERTa (Liu 等人, 2019) 优化了最初在 BERT (Devlin 等人, 2019a) 中提出的预训练配方，并在不引入更多可训练参数的情况下提高了后者的任务性能。虽然 RoBERTa 近年来在 GLUE 基准 (Wang 等人, 2019 年) 等 NLP 排行榜上被更大的模型超越，但就其规模而言，它仍然是一个在从业者中具有竞争力和受欢迎的预训练模型。我们从 HuggingFace Transformers 库 (Wolf 等人, 2020 年) 中提取了预训练的 RoBERTa base (1.25 亿) 和 RoBERTa large (3.55 亿)，并评估了不同高效适应方法在 GLUE 基准任务上的性能。我们还根据 Houlsby 等人 (2019 年) 和 Pfeiffer 等人 (2021 年) 的设置进行了复制。为了确保比较的公平性，我们在与适配器进行比较时，对 LoRA 的评估方式做了两处重要改动。首先，我们对所有任务使用相同的批量大小，并使用 128 的序列长度以匹配适配器基线。其次，我们将模型初始化为 MRPC、RTE 和 STS-B 的预训练模型，而不是像微调基线那样使用已适应 MNLI 的模型。按照 Houlsby 等人 (2019 年) 的这一更受限制的设置进行的运行。结果见表 2 (前三节)。所用超参数详见 D.1 节。

5.3 DeBERTa XXL

DeBERTa (He 等人, 2021 年) 是 BERT 的最新变体，在更大规模上进行了训练，在 GLUE (Wang 等人, 2019 年) 和 SuperGLUE (Wang 等人, 2020 年) 等基准上的表现极具竞争力。我们评估了 LoRA 在 GLUE 上的性能是否仍能与经过全面微调的 DeBERTa XXL (1.5B) 相媲美。结果见表 2 (底部部分)。所用超参数详见 D.2 节。

5.4 GPT-2 中号/大号

在证明了 LoRA 可以在无损检测单元上替代完全微调之后，我们希望回答 LoRA 在无损检测组模型上是否仍然有效，比如 GPT-2 中型和大型模型（Radford 等人，b）。为了进行直接比较，我们的设置尽可能接近 Li & Liang (2021)。由于篇幅有限，我们在本节中只介绍了 E2E NLG 挑战赛的结果（表 3）。有关 WebNLG（Gardent 等人，2017 年）和 DART（Nan 等人，2020 年）的结果，请参见 F.1 节。我们在 D.3 节中列出了所使用的超参数。

模式和方法	# 可培训参数	维基数据库	MNLI-m	SAMSum
		Acc. (%)	Acc. (%)	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 (预埋)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (前置层)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 (适配器) ^H	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 (适配器) ^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

表 4: 不同适应方法在 GPT-3 175B 上的性能。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 的验证准确率。LoRA 的表现优于之前的方法，包括完全微调。WikiSQL 的结果波动在 $\pm 0.5\%$ 左右，MNLI-m 的结果波动在 $\pm 0.1\%$ 左右，SAMSum 的结果波动在 $\pm 0.5\%$ 左右。三项指标的 $\pm 0.2/\pm 0.2/\pm 0.1$ 。

5.5 扩大到 GPT-3 175B

作为对 LoRA 的最终压力测试，我们将其扩展到具有 1 750 亿个参数的 GPT-3。由于训练成本较高，我们只报告了随机种子中给定任务的典型标准偏差，而不是为每个条目提供一个标准偏差。有关所用超参数的详细信息，请参见 D.4 节。

如表 4 所示，LoRA 在所有三个数据集上都达到或超过了微调基线。请注意，如图 2 所示，并非所有方法都能从更多可训练参数中单调获益。当我们使用超过 256 个特殊标记进行前缀嵌入调整或使用超过 32 个特殊标记进行前缀层调整时，我们观察到性能明显下降。这与 Li & Liang (2021 年) 的类似观察结果相吻合。虽然对这一现象的深入研究超出了本文的研究范围，但我们怀疑，使用更多特殊字符会导致输入分布进一步偏离训练前的数据分布。另外，我们将在 F.3 节研究不同适应方法在低数据机制下的性能。

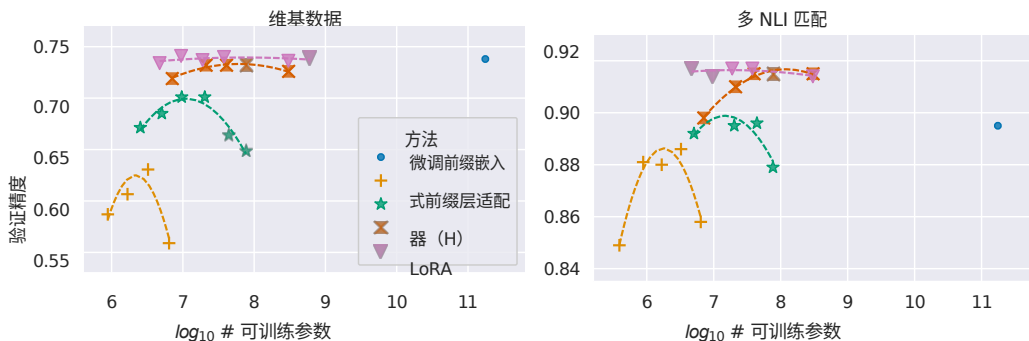


图 2: 几种适应方法在 WikiSQL 和 MNLI 匹配上的 GPT-3 175B 验证准确率与可训练参数数量的关系。LoRA 显示出更好的可扩展性和任务性能。有关绘制数据点的更多详情，请参阅 F.2 节。

6 相关作品

Transformer 语言模型。Transformer (Vaswani 等人, 2017 年) 是一种序列到序列架构, 大量使用了自注意。Radford 等人 (a) 通过使用一叠 Transformer 解码器将其应用于自回归语言建模。从那时起, 基于 Transformer 的语言模型在 NLP 领域占据了主导地位, 在许多任务中都达到了最先进的水平。随着 BERT (Devlin 等人, 2019b) 和 GPT-2 (Radford 等人, b) 的出现, 一个新的范式出现了--这两个模型都是大型 Transformer 语言模型。

与直接在特定任务数据上进行训练相比，在一般领域数据上进行预训练后，在特定任务数据上进行微调可显著提高性能。训练更大的 Transformers 通常会带来更好的性能，这仍然是一个活跃的研究方向。GPT-3 (Brown 等人, 2020 年) 是迄今为止训练过的最大的单个 Transformer 语言模型，拥有 175B 个参数。

提示工程和微调。虽然 GPT-3 175B 只需几个额外的训练示例就能调整其行为，但其结果在很大程度上取决于输入提示 (Brown 等人, 2020 年)。这就需要一种经验艺术，即对提示进行组合和格式化，以最大限度地提高模型在所需任务中的表现，这就是所谓的提示工程或提示黑客 (prompt engineering or prompt hacking)。微调将在一般领域预先训练好的模型重新训练到特定任务 Devlin 等人 (2019b)；Radford 等人 (a)。其变体包括只学习参数的一个子集 Devlin 等人 (2019b)；Collobert & Weston (2008)，但实践者通常会重新训练所有参数，以最大限度地提高下游性能。然而，GPT-3 175B 的庞大规模使得按照常规方法进行微调具有挑战性，因为它会产生大量检查点，而且硬件门槛很高，因为它与预训练具有相同的内存占用。

参数高效自适应。许多人建议在神经网络现有层之间插入 *适配层* (Houlsby 等人, 2019; Rebuffi 等人, 2017; Lin 等人, 2020)。我们的方法使用了类似的瓶颈结构，对权重更新施加低秩约束。功能上的主要区别在于，我们学习到的权重可以在推理过程中与主权重合并，因此不会带来任何延迟，而适配器层则不会出现这种情况 (第 3 节)。适配器的一个当代扩展是 COMPACTER (Mahabadi 等人, 2021 年)，其本质是使用 Kronecker 乘积和某种预先确定的权重共享方案对适配器层进行参数化。同样，将 LoRA 与其他基于张量乘积的方法相结合，也有可能提高其参数效率，但这有待于今后的工作。最近，许多人提议优化输入词嵌入，以代替微调，这类似于提示工程的连续可微分广义 (Li & Liang, 2021; Lester et al.) 我们在实验部分将与 Li & Liang (2021) 进行比较。然而，只有在提示语中使用更多的特殊标记，才能扩大这一研究方向的规模，因为在学习位置嵌入时，这些特殊标记会占用任务标记的可用序列长度。

深度学习中的低阶结构。低秩结构在机器学习中非常常见。很多机器学习问题都有一定的内在低阶结构 (Li 等人, 2016; Cai 等人, 2010; Li 等人, 2018b; Grasedyck 等人, 2013)。此外，众所周知，对于许多深度学习任务，尤其是那些具有严重过参数化神经网络的任务，学习到的神经网络在训练后将享有低秩属性 (Oymak 等人, 2019 年)。一些先前的研究甚至在训练原始神经网络时明确施加了低阶约束 (Sainath 等人, 2013 年; Povey 等人, 2018 年; Zhang 等人, 2014 年; Jaderberg 等人, 2014 年; Zhao 等人, 2016 年; Khodak 等人, 2021 年; Denil 等人, 2014 年)；然而，据我们所知，这些研究都没有考虑对冻结模型进行低阶更新，以 *适应下游任务*。在理论文献中，众所周知，当底层概念类具有一定的低阶结构时，神经网络的表现优于其他经典学习方法，包括相应的 (有限宽度) 神经正切核 (Allen-Zhu 等人, 2019 年; 李和梁, 2018 年) (Ghorbani 等人, 2020 年; Allen-Zhu 和李, 2019 年; Allen-Zhu 和李, 2020a)。Allen-Zhu & Li (2020b) 的另一个理论

结果表明，低等级适应对于对抗训练是有用的。总之，我们认为，我们提出的低等级适应性更新在文献中得到了很好的启发。

7 了解低排名更新

鉴于 LoRA 的经验优势，我们希望进一步解释从下游任务中学习到的低阶适应的特性。请注意，低阶结构不仅降低了硬件门槛，使我们可以并行运行多个实验，还能更好地解释更新权重与预训练权重之间的关联。我们的研究重点是 GPT-3 175B，在不影响任务性能的情况下，我们实现了可训练参数的最大缩减（高达 10,000 倍）。

我们进行了一系列实证研究，以回答以下问题：1) 在有参数预算限制的情况下，我们应该调整预训练变换器中的 *哪个权重矩阵子集*？

以最大限度地提高下游性能？2) "最优"适应矩阵 ΔW 是否真的有等级缺陷？如果是，在实践中使用什么秩比较好？3) ΔW 和 W 之间有什么联系？ ΔW 与 W 高度相关吗？与 W 相比， ΔW 有多大？

我们相信，我们对问题 (2) 和 (3) 的回答揭示了在下游任务中使用预训练语言模型的基本原理，而这正是 NLP 领域的一个重要课题。

7.1 我们应该对变压器中的哪些权重矩阵应用 LoRA？

在参数预算有限的情况下，我们应该利用 LoRA 调整哪类权重，才能在下游任务中获得最佳性能？如第 4.2 节所述，我们只考虑自我关注模块中的权重矩阵。我们在 GPT-3 175B 上设置了 18M 的参数预算（如果存储在 FP16 中，则约为 35MB），如果我们调整一种类型的注意力权重，则对应于 $r = 8$ ；如果我们调整两种类型的权重，则对应于 $r = 4$ ，适用于所有 96 层。结果见表 5。

	# 可训练参数数量 = 1 800 万						
重量类型 W_q, W_v, W_o	W_q	W_v	W_o	W_q, W_v	W_q, W_o	W_v, W_o	W_q, W_v, W_o
等级 r	8	8	8	4	4	2	2
维基数据库 (± 0.5)	70.4	70.0	73.0	73.2	71.4	73.7	73.7
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	91.7

表 5：在可训练参数数量相同的情况下，将 LoRA 应用于 GPT-3 中不同类型的注意力权重后，WikiSQL 和 MultiNLI 的验证准确率。同时调整 W_q 和 W_v 的整体性能最佳。我们发现，对于给定的数据集，不同随机种子的标准偏差是一致的，我们在第一列中报告了这一情况。

需要注意的是，将所有参数都放在 ΔW_q 或 ΔW_k 中会导致性能明显降低，而同时调整 W_q 和 W_v 则会产生最佳结果。这表明，即使是四级权重也能捕捉到 ΔW 中的足够信息，因此调整更多的权重矩阵比调整单一类型的权重级数更大。

7.2 LoRA 的最佳等级 r 是多少？

我们将注意力转向等级 r 对模型性能的影响。我们调整了 $\{W_q, W_v\}$ 、 $\{W_q, W, W_{kv}, W_c\}$ ，仅 W_q 进行比较。

	重量类型	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
维基数据库 (± 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, W_{kv}, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	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, W_{kv}, W_o	91.2	91.7	91.7	91.5	91.4

表 6: 不同等级 r 在 WikiSQL 和 MultiNLI 上的验证准确率。令我们惊讶的是, 在这些数据集上, 小到 1 的等级就足以适应 W_q 和 W_v , 而单独训练 W_q 则需要更大的等级。我们将在 H.2 节的 GPT-2 上进行类似的实验。

表 6 显示, 令人惊讶的是, LoRA 在 r 非常小的情况下 (对于 W_q, W_v 而不仅仅是 W_q 而言, r 越小, 性能越好) 已经具有竞争力。这表明更新矩阵 ΔW 的 "内在秩" 可能非常小。⁶ 为了进一步支持这一发现, 我们检查了不同 r 选择和不同随机种子学习到的子空间的重叠情况。我们认为, 增加 r 并不能覆盖更有意义的子空间, 这表明低秩适应矩阵就足够了。

⁶不过, 我们并不指望小 r 能适用于所有任务或数据集。请看下面的思想实验: 如果下游任务使用的语言与预训练时使用的语言不同, 那么重新训练整个模型 (类似于使用 $r = d$ 的 LoRA_{model}) 的效果肯定会优于使用小 r 的 LoRA。

不同 r 之间的子空间相似性。 给定 $A_{r=8}$ 和 $A_{r=64}$ ，它们是使用相同的预训练模型学习到的秩为 $r=8$ 和 64 的适应矩阵。

分解，并得到右单弦单元矩阵 $U_{A_{r=8}}$ 和 $U_{A_{r=64}}$ 。我们希望通过这两个矩阵进行问题： $U_{A_{r=8}}$ 中前 i 个奇异向量所跨子空间有多少包含在 $U_{A_{r=64}}$ 中前 j 个奇异向量所跨子空间中？我们用基于格拉斯曼距离的归一化子空间相似度来确定这一数量（更正式的讨论见附录 G）。

$$\phi(A_{r=8}, A_{r=64}, i, j) = \frac{\|U_{A_{r=8}}^T U_{A_{r=64}}\|_F^2}{\min(i, j)} \in [0, 1] \quad (4)$$

其中 $U_{A_{r=8}}^i$ 代表 $U_{A_{r=8}}$ 与前 i 个奇异向量相对应。

$\phi(\cdot)$ 的范围为 $[0, 1]$ ，其中 1 代表子空间完全重叠，0 代表完全分离。由于篇幅有限，我们只研究了第 48 层（共 96 层），但正如第 H.1 节所示，结论同样适用于其他层。

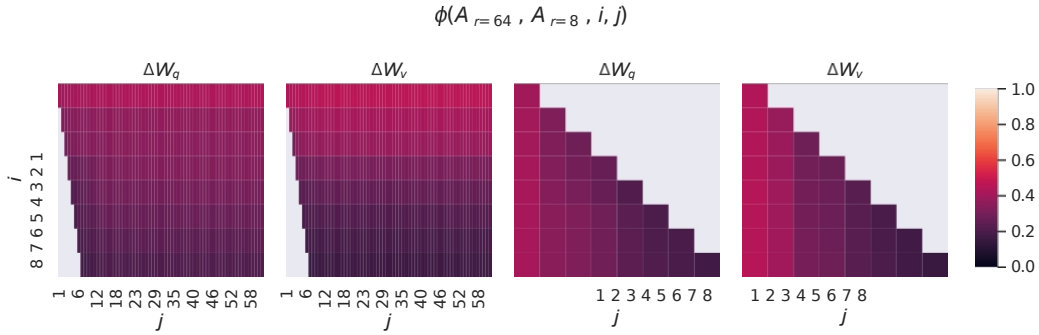


图 3: ΔW_q 和 ΔW_v 的 $A_{r=8}$ 和 $A_{r=64}$ 的列向量之间的子空间相似性。第三和第四张图放大了前两张图中左下角的三角形。 $r=8$ 中的顶部方向包含在 $r=64$ 中，反之亦然。

从图 3 中我们可以看出一个重要的现象。

在 $A_{r=8}$ 和 $A_{r=64}$ 之间，顶部奇异向量对应的方向有明显重叠，而其他方向则没有。具体来说， $A_{r=8}$ 的 ΔW_v （resp. ΔW_q ）和 $A_{r=64}$ 的 ΔW_v （resp. ΔW_q ）共享一个维度为 1 的子空间，归一化相似度大于 0.5，这就解释了为什么 $r=1$ 在 GPT-3 的下游任务中表现相当出色。

由于 $A_{r=8}$ 和 $A_{r=64}$ 都是使用相同的预训练模型学习的，图 3 显示 $A_{r=8}$ 和 $A_{r=64}$ 的顶部奇异向量方向是最有用的，而其他方向可能主要包含训练过程中积累的随机噪音。因此，适应矩阵的秩确实可以很低。

不同随机种子之间的子空间相似性。 我们通过绘制 $r=64$ 的两个随机种子运行之间的归一化子空间相似性，进一步证实了这一点，如图 4 所示。

ΔW_q 似乎比 ΔW_v 具有更高的“内在秩”，因为两次运行都学习到了 ΔW_q 的更多共同奇异值方向，这与我们在表 6 中的经验观察结果一致。作为对比，我们还绘制了两个随机高斯矩阵，它们之间没有任何共同的奇异值方向。

7.3 适应矩阵 ΔW 与 W 相比如何?

我们进一步研究了 ΔW 和 W 之间的关系。特别是, ΔW 是否与 W 高度相关? (或者从数学角度看, ΔW 是否主要包含在 W 的顶部奇异方向中?) 还有、

⁷请注意, 类似的分析也可以用 B 和左单值单元矩阵来进行--我们坚持使用 A 用于我们的实验。

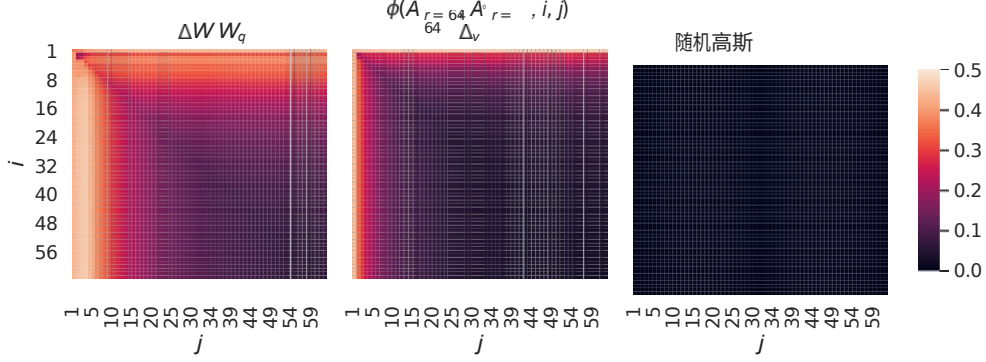


图 4: **左图和中国图**: 第 48 层中 ΔW_q 和 ΔW_v 两个随机种子的 $A_{r=64}$ 列向量之间的归一化子空间相似度。**右图**: 两个随机高斯矩阵列向量之间的热图。其他层请参见第 H.1 节。

与 W 中的相应方向相比, ΔW 有多 "大"? 这可以揭示适应预训练语言模型的内在机制。

为了回答这些问题, 我们通过计算 $U W V^T$ 将 W 投影到 ΔW 的 r 维子空间上, 其中 U/V 是 ΔW 的左/右奇异向量矩阵。然后, 我们比较 $\|U W V^T\|_F$ 和 $\|W\|_F$ 之间的 Frobenius 准则。作为比较, 我们还可以计算 $\|U W V\|_F$, 方法是用 W 的前 r 个奇异向量或随机矩阵代替 U, V 。

	$r = 4$			$r = 64$		
	ΔW_q	W_q	随机	ΔW_q	W_q	随机
$\ U W_q V\ _F =$	0.32	21.67	0.02	1.90	37.71	0.33
$\ W_q\ _F = 61.95$	$\ \Delta W\ _F = 6.91$			$\ \Delta W\ _F = 3.57$		

表 7: $U W V^T$ 的弗罗贝尼斯规范, 其中 U 和 V 是 (1) ΔW_q , (2) W_q , 或 (3) 随机矩阵的左/右顶 r 奇异向量方向。权重矩阵取自 GPT-3 的第 48 层。

从表 7 中我们可以得出几个结论。首先, 与随机矩阵相比, ΔW 与 W 的相关性更强, 这表明 ΔW 放大了 W 中已有的一些特征。其次, ΔW 不会重复 W 的顶部奇异方向, 而只会放大 W 中没有强调的方向。第三, 放大系数相当大: $r = 4$ 时为 $21.5 \approx 6.91/0.32$ 。关于 $r = 64$ 放大系数较小的原因, 请参见第 H.4 节。我们还在第 H.3 节中提供了一个可视化图表, 说明当我们从 W_q 中包含更多的顶奇异方向时, 相关性是如何变化的。这表明低秩适应矩阵可能会放大特定下游任务的重要特征, 而这些特征在一般的预训练模型中已被学习到但未被强调。

8 结论和未来工作

对庞大的语言模型进行微调, 所需的硬件成本和为不同任务托管独立实例的存储/切换成本都非常高昂。我们提出的 LoRA 是一种高效的适应策略, 既不会带来推理延迟, 也不会减少输入序列长度, 同时还能保持较高的模型质量。重要的是, 它通过共享绝大多数模型参

数，允许在作为服务部署时快速切换任务。虽然我们专注于 Transformer 语言模型，但所提出的原则一般适用于任何具有密集层的神经网络。

未来的工作方向有很多。1) LoRA 可以与其他高效的适应方法相结合，从而提供正交改进。2) 微调或 LoRA 背后的机制尚不清楚--在预训练中学习到的特征是如何转变为在下游任务中表现出色的？我们认为 LoRA 比完全微调更容易回答这个问题。

调整。3) 我们主要依靠启发式方法来选择应用 LoRA 的权重矩阵。有没有更有原则的方法？4) 最后, ΔW 的秩缺陷表明 W 也可能是秩缺陷的, 这也是未来工作的灵感来源。

参考资料

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 本征维度解释了语言模型微调的有效性. *arXiv:2012.13255 [cs]*, 2020 年 12 月。URL <http://arxiv.org/abs/2012.13255>.

Zeyuan Allen-Zhu 和 Yuanzhi Li. 超越内核, ResNet 能高效学习什么? 在 *NeurIPS*, 2019. 完整版本见 <http://arxiv.org/abs/1905.10337>.

Zeyuan Allen-Zhu 和 Yuanzhi Li. 后向特征校正: *arXiv preprint arXiv:2001.04413*, 2020a.

Zeyuan Allen-Zhu 和 Yuanzhi Li. 特征净化: 对抗训练如何实现稳健的深度学习. *arXiv preprint arXiv:2005.10190*, 2020b.

Zeyuan Allen-Zhu, Yuanzhi Li 和 Zhao Song. 通过超参数化实现深度学习的收敛理论. In *ICML*, 2019. 完整版本见 <http://arxiv.org/abs/1811.03962>.

Jimmy Lei Ba, Jamie Ryan Kiros 和 Geoffrey E. Hinton. 层规范化, 2016 年。

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, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, 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. *ArXiv:2005.14165 [cs]*, 2020 年 7 月。URL <http://arxiv.org/abs/2005.14165>.

Jian-Feng Cai, Emmanuel J Candes, and Zuowei Shen. 矩阵补全的奇异值阈值算法. *SIAM 优化学报*, 20 (4) : 1956-1982, 2010.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio 和 Lucia Specia. Semeval-2017 任务1: 语义文本相似性多语言和跨语言重点评估. *第 11 届语义评估国际研讨会 (SemEval-2017) 论文集*, 2017 年。doi: 10.18653/v1/s17-2001。URL <http://dx.doi.org/10.18653/v1/S17-2001>.

Ronan Collobert 和 Jason Weston. 自然语言处理的统一架构: 多任务学习的深度神经网络. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pp. 美国计算机协会。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 和 Nando de Freitas. 深度学习中的参数预测, 2014 年。

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 伯特: 用于语言理解的深

度双向变换器的预训练》，2019a.

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova. BERT: 用于语言理解的深度双向变换器预训练。 *arXiv:1810.04805 [cs]* , 2019 年 5 月 b。URL <http://arxiv.org/abs/1810.04805>. arXiv: 1810.04805.

William B. Dolan 和 Chris Brockett. 自动构建语句转述语料库。 *第三届国际转述研讨会 (IWP2005) 论文集*, 2005 年。URL <https://aclanthology.org/I05-5002>.

Claire Gardent、Anastasia Shimorina、Shashi Narayan 和 Laura Perez-Beltrachini. webnlg 挑战: 从 rdf 数据生成文本。 *第 10 届自然语言生成国际会议论文集* , 第 124-133 页, 2017 年。

-
- Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. 神经网络何时优于核方法? *arXiv preprint arXiv:2006.13409*, 2020.
- Bogdan Gliwa、Iwona Mochol、Maciej Biesek 和 Aleksander Wawer。Samsun 语料库：用于抽象概括的人类注释对话数据集。 *CoRR* , abs/1911.12237 , 2019 。 URL <http://arxiv.org/abs/1911.12237>.
- Lars Grasedyck, Daniel Kressner, and Christine Tobler. 低阶张量近似技术文献概览。 *GAMM-Mitteilungen* , 36 (1) : 53-78, 2013.
- Jihun Ham 和 Daniel D. Lee. 格拉斯曼判别分析：基于子空间学习的统一观点。 In *ICML*, pp. URL <https://doi.org/10.1145/1390156.1390204>.
- Karen Hambardzumyan、Hrant Khachatrian 和 Jonathan May。WARP: *ArXiv:2101.00121 [cs]* , 2020 年 12 月。 URL <http://arxiv.org/abs/2101.00121>. arXiv: 2101.00121.
- 何鹏程、刘晓东、高剑锋、陈伟柱。Deberta：具有分散注意力的解码增强贝尔特，2021.
- Neil Houlsby、Andrei Giurgiu、Stanislaw Jastrzebski、Bruna Morrone、Quentin de Laroussilhe、Andrea Gesmundo、Mona Attariyan 和 Sylvain Gelly。用于 NLP 的参数高效迁移学习。 *arXiv:1902.00751 [cs, stat]* , 2019 年 6 月 。 URL <http://arxiv.org/abs/1902.00751>.
- Max Jaderberg、Andrea Vedaldi 和 Andrew Zisserman. 用低级展开加速卷积神经网络》, *arXiv preprint arXiv:1405.3866*, 2014.
- Mikhail Khodak、Neil Tenenholz、Lester Mackey 和 Nicolo` Fusi。因子化神经层的初始化和正则化》, 2021 年。
- Diederik P. Kingma 和 Jimmy Ba. 亚当：随机优化方法》, 2017 年。
- Dmitry Lepikhin、HyukJoong Lee、Yuanzhong Xu、Dehao Chen、Orhan Firat、Yanping Huang、Maxim Krikun、Noam Shazeer 和 Zhifeng Chen。Gshard：利用条件计算和自动分片扩展巨型模型，2020 年。
- Brian Lester、Rami Al-Rfou 和 Noah Constant. 参数高效提示调整的规模力量》。 *arXiv:2104.08691 [cs]* , 2021 年 4 月 。 URL <http://arxiv.org/abs/2104.08691>. arXiv: 2104.08691.
- 李春元、Heerad Farkhoor、Rosanne Liu 和 Jason Yosinski. 测量客观景观的内在密度》, *arXiv:1804.08838 [cs, stat]*, April 2018a. URL <http://arxiv.org/abs/1804.08838>. arXiv: 1804.08838.
- Xiang Lisa Li 和 Percy Liang. 前缀调谐：优化连续提示的生成。 *arXiv:2101.00190 [cs]* , 2021 年 1 月。 URL <http://arxiv.org/abs/2101.00190>.
- 李远志和梁颖宇在结构化数据上通过随机梯度下降学习过参数化神经网络 *神经信息处理系统进展*》, 2018 年。

李远志、梁颖宇、安德烈-里斯特斯基。通过交替最小化保证加权低阶 ap-proximation 的恢复。《国际机器学习大会》，第 2358-2367 页。PMLR, 2016。

李远志、马腾宇、张红阳。过参数化矩阵传感和二次激活神经网络中的算法正则化。In *Conference On Learning Theory*, pp.PMLR, 2018b.

林兆江、Andrea Madotto 和 Pascale Fung.通过参数高效迁移学习探索多功能生成语言模型。In *Findings of the Association for Computational Linguistics: EMNLP 2020*》，第 441-459 页，在线，2020 年 11 月。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]*, 2021 年 3 月。URL <http://arxiv.org/abs/2103.10385>. arXiv: 2103.10385.

刘寅涵、迈勒-奥特、纳曼-戈亚尔、杜京飞、曼达尔-乔希、陈丹琪、奥默-利维、迈克-刘易斯、卢克-泽特勒莫耶和韦塞林-斯托扬诺夫。罗伯塔鲁棒优化的伯特预训练方法，2019 年。

Ilya Loshchilov 和 Frank Hutter. 解耦权重衰减正则化》，*arXiv preprint arXiv:1711.05101*, 2017.

伊利亚-洛希洛夫和弗兰克-胡特解耦权重衰减正则化》，2019 年。

Rabeeh Karimi Mahabadi、James Henderson 和 Sebastian Ruder。Compacter：高效低阶超复杂适配层，2021 年。

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, et al. Dart： *ArXiv preprint arXiv:2007.02871*, 2020.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. e2e 数据集：端到端生成的新挑战。*arXiv preprint arXiv:1706.09254*, 2017.

Samet Oymak, Zalan Fabian, Mingchen Li, and Mahdi Soltanolkotabi. 通过利用 Jacobian 的低秩结构实现神经网络的泛化保证（Generalization Guarantees）。*arXiv 预印本 arXiv:1906.05392*, 2019.

Jonas Pfeiffer、Aishwarya Kamath、Andreas Ruckle、Kyunghyun Cho 和 Iryna Gurevych。适配器-融合：用于迁移学习的非破坏性任务组合，2021 年。

丹尼尔-波维（Daniel Povey）、程高峰、王一鸣、李珂、徐海南、马赫萨-亚莫哈马迪（Mahsa Yarmohammadi）和桑-吉夫-库丹普尔（Sanjeev Khudanpur）。深度神经网络的半正交低阶矩阵因式分解。*Interspeech*, pp.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 通过生成预训练提高语言理解能力》，第 12 页，第 a 部分。

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 语言模型是无监督的多任务学习者》，第 24、b 页。

Pranav Rajpurkar、Robin Jia 和 Percy Liang。知道你不知道的：小队无法回答的问题。*CoRR*, abs/1806.03822, 2018。URL <http://arxiv.org/abs/1806.03822>.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 用残差适配器学习多个视觉域。*arXiv:1705.08045 [cs, stat]*, November 2017. URL <http://arxiv.org/abs/1705.08045>. arXiv: 1705.08045.

Andreas Ruckle、Gregor Geigle、Max Glockner、Tilman Beck、Jonas Pfeiffer、Nils Reimers

和 Iryna Gurevych。Adapterdrop：关于变压器中适配器的效率，2020 年。

Tara N Sainath、Brian Kingsbury、Vikas Sindhwani、Ebru Arisoy 和 Bhuvana Ramabhadran。用于高维输出目标深度神经网络训练的低秩矩阵因式分解。In *2013 IEEE international conference on acoustics, speech and signal processing*, pp.IEEE, 2013.

Mohammad Shoeybi、Mostofa Patwary、Raul Puri、Patrick LeGresley、Jared Casper 和 Bryan Catanzaro。Megatron-lm：2020年，利用模型参数化训练多亿参数语言模型。

Richard Socher、Alex Perelygin、Jean Wu、Jason Chuang、Christopher D. Manning、Andrew Ng 和 Christopher Potts。情感树库语义合成的递归深度模型。《2013 年自然语言处理实证方法会议论文集》，第 1631-1642 页，美国华盛顿州西雅图，2013 年 10 月。计算语言学协会。URL <https://aclanthology.org/D13-1170>.

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。注意力就是你所需要的一切。《第31届国际神经信息处理系统大会论文集》，第6000-6010页，2017年。

Alex Wang、Amanpreet Singh、Julian Michael、Felix Hill、Omer Levy 和 Samuel R. Bowman。Glue：用于自然语言理解的多任务基准和分析平台，2019年。

Alex Wang、Yada Pruksachatkun、Nikita Nangia、Amanpreet Singh、Julian Michael、Felix Hill、Omer Levy 和 Samuel R. Bowman。超级胶水：通用语言理解系统的粘性基准，2020年。

Alex Warstadt、Amanpreet Singh 和 Samuel R Bowman。神经网络可接受性判断。*arXiv preprint arXiv:1805.12471*, 2018.

Adina Williams、Nikita Nangia 和 Samuel Bowman。通过推理进行句子理解的广覆盖挑战语料库。In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: 人类语言技术》第1卷（长篇论文）*，第1112-1122页，路易斯安那州新奥尔良市，2018年6月。计算语言学协会。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、Re'mi 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 和 Alexander M. Rush。变形金刚：最先进的自然语言处理技术。《自然语言处理经验方法2020年会议论文集：系统演示》，第38-45页，在线，2020年10月。计算语言学协会。URL <https://www.aclweb.org/anthology/2020.emnlp-demos.6>。

Greg Yang 和 Edward J. Hu。《无穷宽神经网络中的特征学习》，*arXiv:2011.14522 [cond-mat]*，2021年5月。URL <http://arxiv.org/abs/2011.14522>。arXiv: 2011.14522.

Elad Ben Zaken、Shauli Ravfogel 和 Yoav Goldberg。比特菲特：基于变压器的掩码语言模型的简单参数高效微调，2021年。

Yu Zhang, Ekapol Chuangsuwanich, and James Glass. 使用低秩矩阵因式分解提取深度神经网络瓶颈特征。In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. IEEE, 2014.

赵勇、李金玉、龚一凡。深度神经网络的低秩加对角线自适应。《2016年电气和电子工程师学会声学、语音和信号处理国际会议 (ICASSP)》，pp.5005-5009. IEEE, 2016.

Victor Zhong、Caiming Xiong 和 Richard Socher.Seq2sql: 利用强化学习从自然语言生成结构化查询。 *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

当我们只有少量训练样本时, "快速学习 "或 "快速工程 "就显得非常有利。然而, 在实际应用中, 对于性能敏感的应用, 我们往往需要几千个或更多的训练样本。如表 8 所示, 无论在大数据集还是小数据集上, 微调都能大幅提高模型性能。我们采用了 GPT-3 论文 (Brown 等人, 2020 年) 中关于 RTE 的 GPT-3 few-shot 结果。对于 MNLI-matched, 我们在每个类中使用两个演示, 总共使用六个上下文示例。

方法	MNLI-m (Val. Acc./%)	RTE (Val. Acc./%)
GPT-3 Few-Shot	40.6	69.0
GPT-3 微调	89.5	85.4

表 8: GPT-3 (Brown 等人, 2020 年) 的微调效果明显优于少量学习。

B 适配器层带来的推理延迟

适配器层是以顺序方式添加到预训练模型中的外部模块，而我们提出的 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进行计算，这不可避免地会带来额外的延迟。正如 Rücklé 等人 (2020 年) 所指出的，当模型批量大小和/或序列长度足够大以充分利用硬件并行性时，适配器层带来的延迟可以得到缓解。我们在 GPT-2 介质上进行了类似的延迟研究，证实了他们的观察结果，并指出在某些情况下，特别是批量规模较小的在线推断，增加的延迟可能非常明显。

我们在英伟达 Quadro RTX8000 上通过 100 次试验的平均值来测量单次前向传输的延迟。我们改变了输入批次大小、序列长度和适配器瓶颈维度

r 。我们测试了两种适配器设计：一种是 Houlsby 等人 (2019) 的原始设计，我们称之为适配器^H；另一种是 Lin 等人 (2020) 最新设计的更高效的变体，我们称之为适配器^L。有关设计的更多详情，请参见第 5.1 节。我们在图 5 中绘制了与无适配器基线相比的减速百分比。

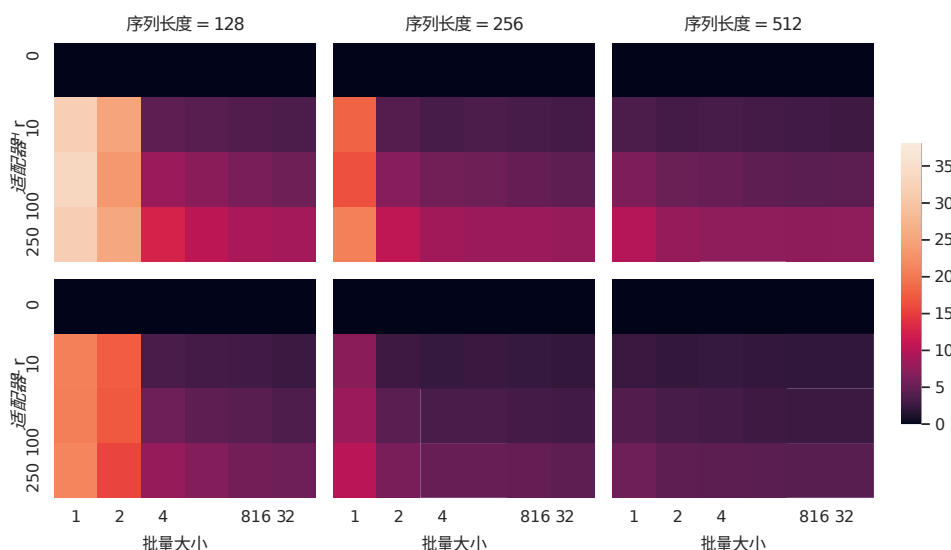


图 5: 与无适配器 ($r = 0$) 基线相比，推理延迟减慢的百分比。上行显示适配器^H的结果，下行显示适配器^L的结果。较大的批量大小和序列长度有助于减轻延迟，但在在线、短序列长度的情况下，延迟可能高达 30% 以上。我们对颜色映射进行了调整，以提高可视性。

C 数据集详情

GLUE Benchmark 是一个范围广泛的自然语言理解任务集合。它包括 MNLI（推理，Williams 等人（2018 年））、SST-2（情感分析，Socher 等人（2013 年））、MRPC（意译检测，Dolan & Brockett（2005 年））、CoLA（语言可接受性，Warstadt 等人（2018 年））、QNLI（推理，Rajpurkar 等人（2018 年））、QQP⁸(问题解答)、RTE（推理）、

⁸<https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>

和 STS-B (文本相似性, Cer 等人 (2017 年))。广泛的覆盖范围使 GLUE 基准成为评估 RoBERTa 和 DeBERTa 等 NLU 模型的标准指标。各个数据集根据不同的许可协议发布。

Zhong 等人 (2017) 介绍了 **WikiSQL**, 其中包含 56、355/8、421 个训练/验证示例。任务是根据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 = 表模式, 将查询和目标编码为 y = SQL。数据集以 BSD 3 条款许可发布。

Gliwa 等人 (2019) 介绍了 **SAMSum**, 其中包含 14 732/819 个训练/测试示例。它由两个人之间的阶段性聊天对话和语言学家撰写的相应抽象摘要组成。我们将上下文编码为 \backslash nl" 串联语句, 后跟一个 "n n", 将目标编码为 y = 摘要。该数据集采用非商业许可发布: Creative Commons BY-NC-ND 4.0。

E2E NLG Challenge 是 Novikova 等人 (2017 年) 首次提出的用于训练端到端、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 数据集来自餐饮领域的大约 42,000 个训练数据、4,600 个验证数据和 4,600 个测试数据组成。作为输入的每个源表可以有多个引用。每个样本输入 (x, y) 由槽值对序列和相应的自然语言参考文本组成。该数据集以知识共享 BY-NC-SA 4.0 协议发布。

DART 是 Nan 等人 (2020 年) 描述的一个开放域数据到文本的数据集。DART 输入结构为 ENTITY - RELATION - ENTITY 三元组序列。与 E2E 相比, DART 是一项规模更大、更复杂的数据到文本任务, 共有 82K 个示例。该数据集采用 MIT 许可发布。

WebNLG 是另一个常用于数据到文本评估的数据集 (Gardent 等人, 2017 年)。WebNLG 共有 22K 个示例, 包括 14 个不同的类别, 其中 9 个在训练过程中出现。由于在总共 14 个类别中, 有 5 个类别在训练过程中没有出现过, 但在测试集中出现过, 因此通常按照 "出现过的" 类别 (S)、"未出现过的" 类别 (U) 和 "全部" 类别 (A) 进行评估。每个输入示例由 SUBJECT - PROPERTY - OBJECT 三元组序列表示。该数据集以知识共享 BY-NC-SA 4.0 协议发布。

D 实验中使用的超参数

D.1 罗伯塔

我们使用 AdamW 和线性学习率衰减计划进行训练。我们对 LoRA 的学习率、训练历元数和批次大小进行了扫描。按照 Liu 等人 (2019) 的做法, 在适应 MRPC、RTE 和 STS-B 时, 我们将 LoRA 模块初始化为最佳 MNLI 检查点, 而不是通常的初始化; 预训练模型在所有任务中都保持冻结。我们报告的是 5 个随机种子的中位数; 每次运行的结果均取自最佳历时。为了与 Houlsby 等人 (2019 年) 和 Pfeiffer 等人 (2021 年) 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的批次大小。重要的是, 在适应 MRPC、RTE 和 STS-B 时, 我们从预先训练的 RoBERTa 大型模型开始, 而不是已经适应

MNLI 的模型。使用这种受限设置的运行以 。超参数见表 9。

D.2 德贝塔

我们再次使用 AdamW 和线性学习率衰减计划进行训练。按照 He 等人（2021 年）的方法，我们调整了学习率、辍学概率、热身步骤和批量大小。为了保持比较的公平性，我们使用了与（He 等，2021 年）相同的模型序列长度。按照 He 等人（2021 年）的做法，在适应 MRPC、RTE 和 STS-B 时，我们将 LoRA 模块初始化为最佳 MNLI 检查点，而不是通常的初始化；预训练模型在所有任务中都保持冻结。我们报告的是 5 个随机种子的中位数；每次运行的结果取自最佳历时。请参见表 10 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							
	热身比例	0.06							
	LR 时间表	线性							
罗伯塔基地 LoRA	批量大小	16	16	16	32	32	16	32	16
	学习率	30	60	30	80	25	25	80	40
	LoRA 配置。	5E-04	5E-04	4E-04	4E-04	4E-04	5E-04	5E-04	4E-04
	LoRA α	$r_q = r_v = 8$							
	最大序列Len.	8							
RoBERTa large LoRA	批量大小	4	4	4	4	4	4	8	8
	学习率	10	10	20	20	10	20	20	30
	LoRA 配置。	3E-04	4E-04	3E-04	2E-04	2E-04	3E-04	4E-04	2E-04
	LoRA α	$r_q = r_v = 8$							
	最大序列Len.	16							
RoBERTa large LoRA+	批量大小	10	10	20	20	10	20	20	10
	学习率	3E-04	4E-04	3E-04	2E-04	2E-04	3E-04	4E-04	2E-04
	LoRA 配置。	$r_q = r_v = 8$							
	LoRA α	16							
	最大序列Len.	128							
RoBERTa large Adpt ^p (3M) \dagger	批量大小	32							
	# 年代	10	20	20	20	10	20	20	20
	学习率	3E-05	3E-05	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04
	瓶颈 r	64							
	最大序列Len.	128							
RoBERTa large Adpt ^p (0.8M) \dagger	批量大小	32							
	# 年代	5	20	20	20	10	20	20	20
	学习率	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04
	瓶颈 r	16							
	最大序列Len.	128							
RoBERTa large Adpt ^H (6M) \dagger	批量大小	32							
	# 时代	10	5	10	10	5	20	20	10
	学习率	3E-05	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04
	瓶颈 r	64							
	最大序列Len.	128							
RoBERTa large Adpt ^H (0.8M) \dagger	批量大小	32							
	# 时代	10	5	10	10	5	20	20	10
	学习率	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04
	瓶颈 r	8							
	最大序列Len.	128							

表 9：我们在 GLUE 基准上使用的 RoBERTa 超参数。

D.3 GPT-2

我们使用 AdamW (Loshchilov & Hutter, 2017 年) 对所有 GPT-2 模型进行训练, 采用线性学习率计划, 持续 5 个历元。我们使用了 Li 和 Liang (2021 年) 中描述的批次大小、学习率和波束搜索波束大小。因此, 我们也为 LoRA 调整了上述超参数。我们报告的是 3 个随机种子的平均值; 每次运行的结果取自最佳历时。表 11 列出了 GPT-2 中用于 LoRA 的超

参数。其他基线使用的超参数见 Li & Liang (2021)。

D.4 GPT-3

在所有 GPT-3 实验中，我们使用 AdamW (Loshchilov 和 Hutter, 2017 年) 进行 2 个历时训练，批量大小为 128 个样本，权重衰减因子为 0.1。我们使用序列长度为 384 的

方法	数据集	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
	优化器 热身比例 LR 时间表	AdamW 0.1 线性							
DeBERTa XXL	批量大小	8	8	32	4	6	8	4	4
		5	16	30	10	8	11	11	10
	LoRA 子句半	1E-04	6E-05	2E-04	1E-04	1E-04	1E-04	2E-04	2E-04
	重量衰减	0	0.01	0.01	0	0.01	0.01	0.01	0.1
	CLS 中途退学	0.15	0	0	0.1	0.1	0.2	0.2	0.2
	LoRA 配置。 LoRA α 最大序列Len.	$r_q = r_v = 8$ 8 256 128 128 64 512 320 320 128							

表 10: DeBERTa XXL 在 GLUE 基准任务中的超参数。

数据集	E2E	WebNLG	DART
	培训		
优化器重量	AdamW		
衰减	0.01	0.01	0.0
辍学概率	0.1	0.1	0.0
批量大小	8		
# Epoch	5		
热身步骤	500		
学习费率表	线性		
光滑标签	0.1	0.1	0.0
学习率适应	0.0002		
LoRA α	$R_q = R_v = 4$ 32		
	推论		
光束尺寸	10		
长度处罚	0.9	0.8	0.8
无重复	4		

表 11: GPT-2 LoRA 在 E2E、WebNLG 和 DART 上的超参数。

WikiSQL (Zhong 等人, 2017 年), MNLI (Williams 等人, 2018 年) 为 768, SAMSum (Gliwa 等人, 2019 年) 为 2048。我们调整了所有方法-数据集组合的学习率。有关所用超参数的更多详情, 请参见 D.4 节。在前缀嵌入调整方面, 我们发现最优的 l_p 和 l_i 分别为 256 和 8, 共计 320 万个可训练参数。我们使用 $l_p = 8$ 和 $l_i = 8$ 进行前缀层调整, 共使用 2020 万个可训练参数, 以获得整体最佳性能。我们为 LoRA 提出了两种参数预算: 4.7M ($r_q = r_v = 1$ 或 $r_v = 2$) 和 37.7M ($r_q = r_v = 8$ 或 $r_q = r = r_{kv} = r_o = 2$)。我们报告了每次运行的最佳验证性能。表 12 列出了 GPT-3 实验中使用的训练超参数。

E 将 LoRA 与前缀调整相结合

LoRA 可以很自然地与现有的基于前缀的方法相结合。在本节中, 我们将在 WikiSQL 和

MNLI 上评估 LoRA 与前缀调整变体的两种组合。

LoRA+PrefixEmbed (LoRA+PE) 将 LoRA 与前缀嵌入调整结合起来，我们在其中插入了 $l_p + l_i$ 特殊标记，其嵌入被视为可训练参数。有关前缀嵌入调整的更多信息，请参见第 5.1 节。

LoRA+PrefixLayer (LoRA+PL) 将 LoRA 与前缀层调整相结合。我们还插入 $l_p + l_i$ ，然而，我们并不是让这些标记的隐藏表示自然演化，而是让这些标记的隐藏表示自然演化。

超参数	微调	预埋	预置层	BitFit	适配器 ^H	LoRA
优化器			AdamW			
批量大小			128			
# Epoch			2			
热身代币 LR 时间表			250,000 线性			
学习率	5.00E-06	5.00E-04	1.00E-04	1.6E-03	1.00E-04	2.00E-04

表 12：不同 GPT-3 适应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。

因此，我们在每个变换器区块之后都用一个与输入无关的向量来替换它们。因此，嵌入和随后的变换器块激活都被视为可训练参数。有关前缀层调整的更多信息，请参见第 5.1 节。

表 15 显示了 LoRA+PE 和 LoRA+PL 在 WikiSQL 和 MultiNLI 上的评估结果。首先，在 WikiSQL 上，LoRA+PE 明显优于 LoRA 和前缀嵌入调优，这表明 LoRA 与前缀嵌入调优具有一定的正交性。在 MultiNLI 上，LoRA+PE 组合的性能并不优于 LoRA，这可能是因为 LoRA 本身已经达到了与人类基线相当的性能。其次，我们注意到 LoRA+PL 的表现略逊于 LoRA，即使使用了更多的可训练参数。我们认为这是由于前缀层调整对学习率的选择非常敏感，因此在 LoRA+PL 中，LoRA 权重的优化变得更加困难。

F 其他经验性实验

F.1 关于 GPT-2 的其他实验

我们还按照 Li & Liang (2021) 的设置在 DART (Nan 等人, 2020 年) 和 WebNLG (Gardent 等人, 2017 年) 上重复了实验。结果如表 13 所示。与我们在第 5 节中报告的 E2E NLG 挑战赛的结果类似，在可训练参数数量相同的情况下，LoRA 的表现优于或至少与基于前缀的方法相当。

方法	# 可培训	DART		
		<i>bleu</i> ↑	<i>met</i> ↑	<i>ter</i> ↓
GPT-2 介质				
微调	354M	46.2	0.39	0.46
适配器 ^L	0.37M	42.4	0.36	0.48
适配器 ^L	11M	45.2	0.38	0.46
FT ^{返回顶部2}	24M	41.0	0.34	0.56
前置层	0.35M	46.4	0.38	0.46
LoRA	0.35M	47.1 _{±.2}	0.39	0.46
GPT-2 大型				

微调	774M	47.0	0.39	0.46
适配器 ^L	0.88M	45.7 \pm .1	0.38	0.46
适配器 ^L	23M	47.1 \pm .1	0.39	0.45
前置层	0.77M	46.7	0.38	0.45
LoRA	0.77M	47.5\pm.1	0.39	0.45

表 13：在 DART 上采用不同适应方法的 GPT-2。所有适配方法的 MET 和 TER 方差均小于 0.01。

方法	网络中立小组								
	BLEU \uparrow			MET \uparrow			TER \downarrow		
	U	S	A	U	S	A	U	S	A
GPT-2中									
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	55.3\pm.2	.38	.44	.41	.46	.33	.39
GPT-2 大型									
微调 (774M)	43.1	65.3	55.5	.38	.46	.42	.53	.33	.42
适配器 ^L (0.88M)	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	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

表 14: 在 WebNLG 上采用不同适应方法的 GPT-2。在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.2 关于 GPT-3 的其他实验

表 15 列出了采用不同适应方法对 GPT-3 进行的其他运行情况。重点是确定性能与可训练参数数量之间的权衡。

F.3 低数据模式

为了评估不同适应方法在低数据条件下的性能, 我们从 MNLI 的完整训练集中随机抽取 100、1k 和 10k 个训练示例, 形成低数据的 *MNLI-n* 任务。表 16 显示了不同适配方法在 MNLI-n 任务中的性能。

n. 出乎我们意料的是, PrefixEmbed 和 PrefixLayer 在 MNLI-100 数据集上的表现非常糟糕, PrefixEmbed 的表现仅略高于随机概率 (37.6% 对 33.3%)。PrefixLayer 的表现好于 PrefixEmbed, 但在 MNLI-100 数据集上的表现仍明显差于 Fine-Tune 或 LoRA。

100. 随着训练示例数量的增加, 基于前缀的方法与 LoRA/微调之间的差距越来越小, 这可能表明基于前缀的方法不适合 GPT-3 中的低数据任务。在 MNLI-100 和 MNLI-Full 上, LoRA 的性能都优于微调, 而在 MNLI-1k 和 MNLI-10K 上, 考虑到随机种子造成的 (± 0.3) 方差, LoRA 的结果也不相上下。

第 17 页报告了不同适应方法在 MNLI-n 上的训练超参数。在 MNLI-100 集上, 我们对 PrefixLayer 使用了较小的学习率, 因为较大的学习率并不会减少训练损失。

G 测量子空间之间的相似性

在本文中，我们使用度量 $\varphi(A, B, i, j) = \psi(U^i_A, U^j_B) = \frac{\|U^{iT}_A U^j_B\|_F}{\min_{\{i,j\}} \|U^i_A\|_F \|U^j_B\|_F}$ 来度量量子空间两列正交矩阵 $U^i_A \in \mathbb{R}^{d \times i}$ 和 $U^j_B \in \mathbb{R}^{d \times j}$ ，由以下公式得到

我们指出，这种相似性只是衡量量子空间间距离的标准投影度量（Projection Metric）的一种反向测量。

方法	超参数	# 可训练参数	维基数据库	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 ^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
	$R_q = R_v = 64, L_p = 8, L_i = 4$			
LoRA+PL	$R_q = R_v = 8, L_p = 8, L_i = 4$	52.8 M	72.9	90.2

表 15：不同适配方法在 WikiSQL 和 MNLI 上的超参数分析。随着可训练参数数量的增加，前缀嵌入调整（PrefixEmbed）和前缀层调整（PrefixLayer）的性能都有所下降，而 LoRA 的性能趋于稳定。性能以验证准确率来衡量。

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

表 16：不同方法在使用 GPT-3 175B 的 MNLI 子集上的验证精度。MNLI- n 表示包含 n 个训练实例的子集。我们使用完整的验证集进行评估。与其他方法（包括微调）相比，LoRA 的样本效率更高。

具体来说, 让 $U U^T$ 的奇异值为 $\sigma_1, \sigma_2, \dots, \sigma_p$, 其中 $p = \min\{i, j\}$ 。我们
 我们知道 Ham 和 Lee (2008 年) 对 "投影度量" 的定义是

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

超参数	改编	MNLI-100	MNLI-1k	MNLI-10K	MNLI-392K
优化器	-			AdamW	
热身代币	-			250,000	
LR 时间表	-			线性	
批量大小	-	20	20	100	128
# Epoch	-	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
适应-- 具体内容	LoRA			2.00E-4	
	前缀嵌入 l_p	16	32	64	256
	前缀嵌入 l_i			8	
	前缀调谐			$L_p = L_i = 8$	
	LoRA			$R_q = R_v = 8$	

表 17: MNLI(m)-n 上不同 GPT-3 适应方法使用的超参数。

其中, 我们的相似性定义为

$$\varphi(A, B, i, j) = \psi(U_A^i, U_B^j) = \frac{\sum_p \sigma^2}{p} \frac{1}{p} \frac{1}{1 - d(U_A^i, U_B^j)^2}$$

如果 U^i 和 U^j 的列跨度相同, 则 $\varphi(A, B, i, j) = 1$ 。如果
否则, $\varphi(A, B, i, j) \in (0, 1)$ 。

H 低库矩阵的其他实验

我们还介绍了对低阶更新矩阵的研究结果。

H.1 LoRA 模块之间的相关性

关于图 3 和图 4 所示结果如何推广到其他层, 请参见图 6 和图 7。

H.2 r 对 GPT-2 的影响

我们在 GPT-2 中重复了关于 r 影响的实验 (第 7.2 节)。以 E2E NLG Challenge 数据集为例, 我们报告了在训练 26000 步后, 不同 r 选择所带来的验证损失和测试指标。结果见表 18。根据所使用的指标, GPT-2 Medium 的最佳排名在 4 到 16 之间, 这与 GPT-3 175B 的排名相似。需要注意的是, 模型大小与适应性最佳等级之间的关系仍是一个未决问题。

H.3 W 和 ΔW 之间的相关性

W 和 ΔW 之间的归一化子空间相似性见图 8, r 值不断变化。

请再次注意, ΔW 并不包含 W 中最重要的奇异方向, 因为 ΔW 中最重要的 4 个方向与 W 中最重要的 10% 方向之间的相似度几乎不超过 0.2。这证明 ΔW 包含了 W 中没有强调的 "特定任务" 方向。

接下来要回答的一个有趣问题是, 我们需要多 "强" 地放大这些特定任务的方向, 才能使模型适应性良好地发挥作用?

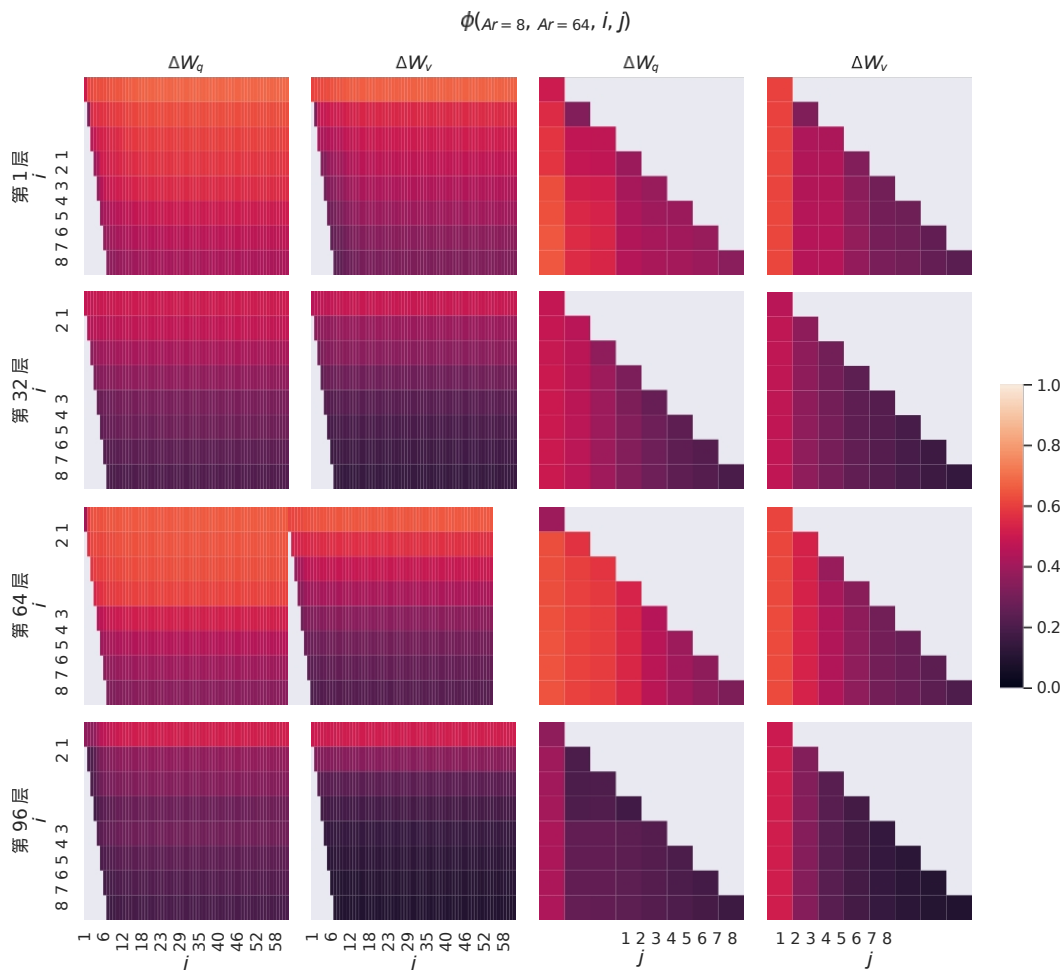


图 6: $A_{r=8}$ 和 $A_{r=64}$ 的列向量之间的归一化子空间相似性。 ΔW_q 和 ΔW_v 来自 96 层变压器中的第 1 层、第 32 层、第 64 层和第 96 层。

H.4 放大系数

我们可以很自然地将特征放大系数视为比率 $\frac{\|\Delta W^F\|}{\|U^T W V\|_F}$ 其中 U 和 V 是 ΔW 的 SVD 分解的左旋和右旋矩阵。（回顾 $U U^T W V V^T$ 给出了 W 在 ΔW 所跨子空间上的“投影”）。

直观地说，当 ΔW 主要包含特定任务方向时，这个量可以衡量 ΔW 放大了多少特定任务方向。如第 7.3 节所示，当 $r = 4$ 时，这一放大系数高达 20。换句话说，（一般来说）每层有四个特征方向（来自预训练模型 W 的整个特征空间）需要放大 20 倍，才能达到我们报告的下游特定任务的准确率。而且，我们应该想到，对于每个不同的下游任务，需要放大的特征方向集也会大相径庭。

然而，我们可以注意到，当 $r = 64$ 时，放大系数仅为 2 左右，这意味着在 $r = 64$ 的 ΔW 中学习到的大多数方向并没有被放大很多。这并不令人吃惊，事实上，这也再次证明了代表“特定任务方向”所需的内在等级（即模型适应所需的等级）很低。相比之下，秩 4 版本的 ΔW （对应 $r=4$ ）中的那些方向被放大了 20 倍之多。

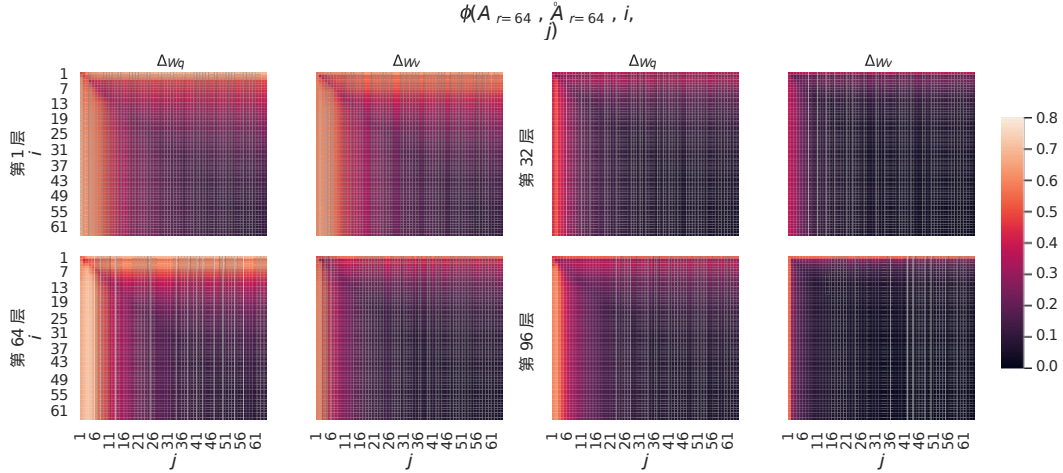


图 7: 96 层 Trans- former 中第 1 层、第 32 层、第 64 层和第 96 层的 ΔW_q 和 ΔW_v 两次随机播种运行的 $A_{r=64}$ 的列向量之间的归一化子空间相似度。

等级 r	val_loss	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

表 18: 使用 GPT-2 Medium 的 LoRA 在不同等级 r 的 E2E NLG 挑战赛中取得的验证损失和测试集指标。与 GPT-3 不同的是, 在 GPT-3 中, $r = 1$ 对许多任务来说都足够了, 而在这里, 验证损失的峰值是 $r = 16$, BLEU 的峰值是 $r = 4$, 这表明与 GPT-3 175B 相比, GPT-2 Medium 具有相似的内在适应等级。请注意, 我们的一些超参数是在 $r = 4$ 的基础上调整的, 这与另一个基线的参数数相匹配, 因此对于其他 r 的选择可能不是最佳的。

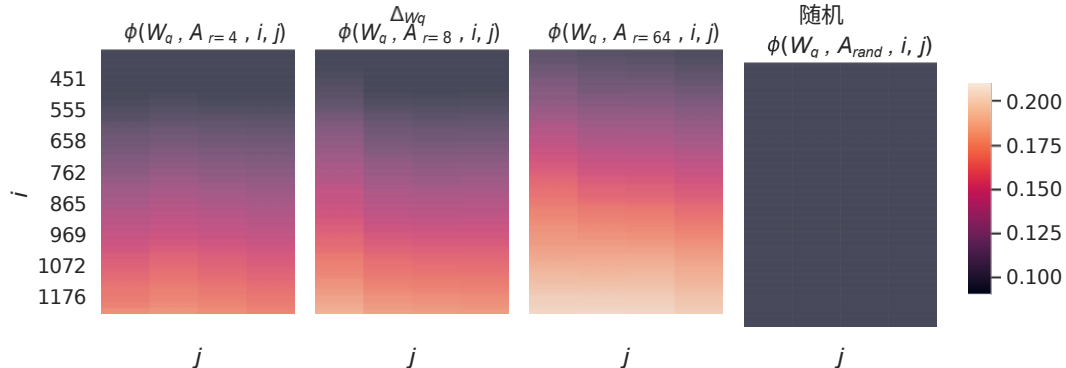


图 8: W_q 的奇异方向与 ΔW_q 的奇异方向之间的归一化子空间相似度, r 和随机基线各不相同。

同。 ΔW_q 放大了 W 中重要但未被强调的方向。 r 越大的 ΔW 越倾向于选取 W 中已被强调的方向。

$$\phi(A_{r=64}, \hat{A}_{r=64}, i, j)$$

ΔW_q
 ΔW_v
 ΔW_q
 ΔW_v