1. LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS（ICLR 2022)

摘要

自然语言处理的一个重要范式包括在通用领域数据上进行大规模预训练,然后适应特定任务或领域。随着我们预训练更大的模型,重新训练所有模型参数的全量微调变得越来越不可行。以GPT-3 175B为例 - 部署每个独立的微调模型实例,每个都有175B参数,成本高昂。

我们提出了低秩适应(Low-Rank Adaptation,LoRA),它冻结预训练模型权重,并在Transformer架构的每一层中注入可训练的秩分解矩阵,大大减少了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可以将可训练参数的数量减少10,000倍,将GPU内存需求减少3倍。

尽管可训练参数更少,训练吞吐量更高,而且与adapters不同,没有额外的推理延迟,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。

我们还对语言模型适应中的秩缺陷进行了实证研究,这为LoRA的有效性提供了洞见。

我们发布了一个软件包,可以方便地将LoRA与PyTorch模型集成,并提供了我们在RoBERTa、DeBERTa和GPT-2上的实现和模型检查点,网址为https://github.com/microsoft/LoRA。

1.1. 引言

许多自然语言处理应用依赖于将一个大规模的预训练语言模型适应到多个下游应用。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。微调的主要缺点是新模型包含与原始模型一样多的参数。随着每隔几个月就训练出更大的模型,这从GPT-2或RoBERTa large的单纯"不便"变成了GPT-3(拥有1750亿可训练参数)的关键部署挑战。

许多人试图通过只适应一些参数或为新任务学习外部模块来缓解这一问题。这样,我们只需要为每个任务存储和加载少量的特定任务参数,除了预训练模型之外,大大提高了部署时的操作效率。

然而,现有技术通常会引入推理延迟(Houlsby等人,2019; Rebuffi等人,2017),方法是扩展模型深度或减少模型的可用序列长度(Li & Liang, 2021; Lester等人, 2021; Hambardzumyan等人, 2020; Liu等人, 2021)(第3节)。更重要的是,这些方法通常无法匹配微调基线,在效率和模型质量之间造成了权衡。

我们从Li等人(2018a)和Aghajanyan等人(2020)的工作中得到启发,他们表明学习到的过度参数化模型实际上位于低内在维度上。我们假设模型适应过程中权重的变化也具有低"内在秩",从而提出了我们的低秩适应(LoRA)方法。

LoRA允许我们通过优化适应过程中密集层变化的秩分解矩阵来间接训练神经网络中的一些密集层,同时保持预训练权重冻结,如图1所示。以GPT-3 175B为例,我们展示了即使完全秩(即d)高达12,288,非常低的秩(即图1中的r可以是1或2)就足够了,使LoRA在存储和计算上都很高效。

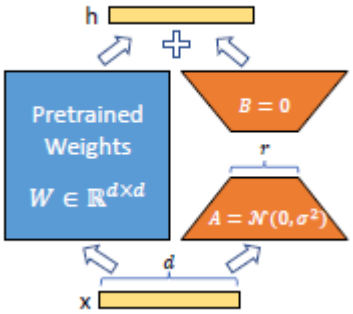


图1:我们的重参数化。我们只训练A和B。

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的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,显著降低存储需求和任务切换开销。
- LoRA使训练更加高效,并降低了硬件门槛,使用自适应优化器时可以降低3倍,因为我们不需要计算大多数参数的梯度或维护优化器状态。相反,我们只优化注入的、更小的低秩矩阵。
- 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,构造上不会引入推理延迟,与完全微调的模型相比。
- LoRA与许多先前的方法正交,可以与它们中的许多方法结合,比如prefix-tuning。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Transformer架构,并使用其维度的常规术语。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指预训练的权重矩阵,ΔW指适应过程中累积的梯度更新。我们使用r来表示LoRA模块的秩。我们遵循Vaswani等人(2017)和Brown等人(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$d_{ffn} = 4 \times d_{model}$。

1.2. 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我们的激励用例。以下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等人,2017)的通用多任务学习器,如GPT(Radford等人,b; Brown等人,2020)。考虑将这个预训练模型适应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:$Z = \{(x_i, y_i)\}_{i=1,..,N}$,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,$y_i$是其对应的SQL命令;对于摘要,$x_i$是文章内容,$y_i$是其摘要。

在全量微调期间,模型初始化为预训练权重Φ_0,并通过反复遵循梯度更新为$\Phi_0 + \Delta\Phi$,以最大化条件语言建模目标:

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

全量微调的主要缺点之一是,对于每个下游任务,我们学习一组不同的参数$\Delta\Phi$,其维度$|\Delta\Phi|$等于$|\Phi_0|$。因此,如果预训练模型很大(例如GPT-3,$|\Phi_0| \approx 175$亿),存储和部署许多独立的微调模型实例可能具有挑战性,如果不是不可行的话。

在本文中,我们采用了一种更加参数高效的方法,其中特定任务的参数增量$\Delta\Phi = \Delta\Phi(\Theta)$进一步由一组维度更小的参数$\Theta$编码,$|\Theta| \ll |\Phi_0|$。因此,找到$\Delta\Phi$的任务变成了优化$\Theta$:

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

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

1.3. 现有解决方案还不够好吗?

我们着手解决的问题绝不是新问题。自迁移学习诞生以来,已有数十项工作试图使模型适应更加参数高效和计算高效。参见第6节对一些知名工作的概述。以语言建模为例,在进行高效适应时有两种突出的策略:添加适配器层(Houlsby等人,2019; Rebuffi等人,2017; Pfeiffer等人,2021; Rückle等人,2020)或优化某种形式的输入层激活(Li & Liang,2021; Lester等人,2021; Hambardzumyan等人,2020; Liu等人,2021)。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 适配器有许多变体。我们关注Houlsby等人(2019)提出的原始设计,即每个Transformer块有两个适配器层,以及Lin等人(2020)最近提出的一种变体,每个块只有一个适配器层,但有一个额外的LayerNorm(Ba等人,2016)。虽然可以通过剪枝层或利用多任务设置来降低总体延迟(Rückle等人,2020; Pfeiffer等人,2021),但没有直接的方法可以绕过适配器层中的额外计算。这看起来不是问题,因为适配器层被设计成只有很少的参数(有时<1%的原始模型),通过使用小的瓶颈维度来限制它们可以添加的FLOPs。然而,大型神经网络依赖于硬件并行性来保持延迟较低,而适配器层必须按顺序处理。这在在线推理设置中会产生差异,在这种情况下,批处理大小通常小到只有1。在不使用模型并行的通用场景中,例如在单个GPU上运行GPT-2(Radford等人,b)中型模型的推理,我们看到使用适配器时延迟明显增加,即使瓶颈维度非常小(表1)。

当我们需要像Shoeybi等人(2020); Lepikhin等人(2020)那样对模型进行分片时,这个问题会变得更糟,因为额外的深度需要更多的同步GPU操作,如AllReduce和Broadcast,除非我们多次冗余存储适配器参数。

直接优化提示是困难的 另一个方向,以prefix tuning(Li & Liang,2021)为代表,面临着不同的挑战。我们观察到prefix tuning很难优化,其性能随可训练参数的变化呈非单调性,这证实了原论文中的类似观察。更根本地说,为适应保留部分序列长度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比其他方法表现更差。我们将任务性能的研究推迟到第5节。

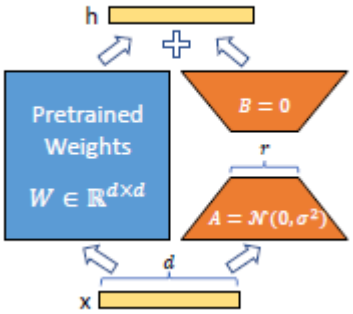
1.4. 我们的方法

我们描述LoRA的简单设计及其实际好处。这里概述的原则适用于深度学习模型中的任何密集层,尽管在我们的实验中,我们仅关注Transformer语言模型中的某些权重,作为激励用例。

1.4.1. 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。在适应特定任务时,Aghajanyan等人(2020)表明,预训练语言模型具有较低的"内在维度",即使在随机投影到更小的子空间后仍能有效学习。受此启发,我们假设适应过程中权重的更新也具有较低的"内在秩"。对于预训练权重矩阵$W_0 \in \mathbb{R}^{d\times k}$,我们通过将其更新表示为低秩分解来约束它:$W_0 + \Delta W = W_0 + BA$,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$\Delta W = BA$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$h = W_0x$,我们修改后的前向传播得到:

$$h = (W_0 + \Delta W)xW_0x + \Delta Wx = W_0x + BAx$$



我们在图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,因此在训练开始时$\Delta W = BA$为零。然后我们将$\Delta W x$缩放$\frac{\alpha}{r}$,其中α是r的常数。当使用Adam优化时,调整α大致相当于调整学习率(如果我们适当地缩放初始化)。因此,我们简单地将α设置为我们尝试的第一个r,并不调整它。这种缩放有助于减少在改变r时重新调整超参数的需要(Yang & Hu, 2021)。

> 训练时使用的公式是 $W_0 x + \frac{\alpha}{r} BAx$。在不同的任务中r的选择可能有所不同，但是作者实验中测试发现即使很小的r也有不错的效果，而对于GPT2，最优的在4到16之间，建议从小开始，之后逐渐增加
>
> 缩放因子$\frac{\alpha}{r}$帮助控制LoRA更新的幅度，α设置为与初始的r一样即可，简化调优
>
> 在训练开始时$\Delta W = BA$为零，这确保了在训练开始时,$\Delta W = BA = 0$,即不改变原始预训练模型的行为。
>
> 在给定需要微调的总参数情况下，平均微调q,k,v,o四个矩阵或者q和v两个矩阵的效果都可以，但是把全部参数放在一个矩阵普遍的效果都不好

> 使用LoRa，我们在训练时，我们原本需要训练的参数量为$d \times d$，现在就变为了$d \times r + r \times d$，如果r远小于d,则大大减少了显存消耗。如果d = 1024，r = 8：
>
> - 原始参数数：1024 × 1024 = 1,048,576
> - LoRA参数数：(1024 × 8) + (8 × 1024) = 16,384
>
> 需要训练的参数数量的显著减少，约为原始数量的1.56%。
>
> 我们自己的实验经验一般可以选择r为4,8,16,32，而α则选择r的2倍

> "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。在适应特定任务时,Aghajanyan等人(2020)表明,预训练语言模型具有较低的"内在维度",即使在随机投影到更小的子空间后仍能有效学习。"——Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning ACL2021。内在维度解释语言模型微调的有效性，可以详见：内在维度解释了语言模型微调的有效性.md
>
> 本文的贡献如下:
>
> - 我们从经验上表明,常见NLP任务在预训练表示的背景下具有比完整参数化低几个数量级的内在维度。
> - 我们提出了内在维度的新解释,即下游微调任务在预训练模型框架内的最小描述长度。在这种解释下,我们从经验上表明预训练过程隐式地优化了NLP任务的平均描述长度,而无需直接访问这些任务。
> - 我们测量了大量最近开发的预训练方法的内在维度。我们发现存在一个幸运的趋势,即更大的模型往往具有更小的内在维度。

- 最后,我们表明基于压缩的泛化界可以应用于我们的内在维度框架,为大型预训练模型提供独立于预训练模型参数数量的泛化界。

> 内在维度(Intrinsic Dimension)是指解决一个特定优化问题（在这里是指微调一个NLP任务）所需的最小参数数量。
>
> 什么是内在维度?
>
> 想象一下有一个复杂的立体拼图，代表一个特定的NLP任务。现在我们有两个人来解决这个拼图:
>
> 1. 小明（代表小模型）:
>   - 小明有100块积木，每块都是不同的形状。
>   - 要完成拼图，小明需要使用90块积木。
>   - 在这个例子中，小明的"内在维度"是90。
> 2. 大华（代表大模型）:
>   - 大华有1000块积木，形状更加丰富多样。
>   - 因为大华的积木更加多样化，他只需要使用30块就能完成同样的拼图。
>   - 大华的"内在维度"是30。
>
> 那如何理解大的模型往往具有更小的内在维度?
>
> 大华（大模型）就像那个更聪明的学生。他可能有更多的背景知识（更多参数），但正是这些丰富的知识使他能够更快、更有效地学习新事物（完成新任务），只需要调整很少的概念（参数）。

全量微调的推广。 一种更一般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应过程中具有满秩。这意味着,当将LoRA应用于所有权重矩阵并训练所有偏置时,通过将LoRA秩r设置为预训练权重矩阵的秩,我们大致恢复了全量微调的表达能力。换句话说,随着我们增加可训练参数的数量,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不能接受长输入序列的模型。

没有额外的推理延迟。 在生产环境中部署时,我们可以显式计算并存储$W = W_0 + BA$,并像往常一样进行推理。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,我们可以通过减去BA然后添加不同的$B'A'$来恢复W_0,这是一个内存开销很小的快速操作。关键是,这保证了我们在推理过程中不会引入任何额外的延迟,与微调模型相比。

1.4.2. 将LORA应用于TRANSFORMER

原则上,我们可以将LoRA应用于神经网络中任何子集的权重矩阵,以减少可训练参数的数量。在Transformer架构中,自注意力模块中有四个权重矩阵(W_q, W_k, W_v, W_o),MLP模块中有两个。我们将W_q(或W_k, W_v)视为单个维度为$d_{model} \times d_{model}$的矩阵,尽管输出维度通常被切分成注意力头。为了简单和参数效率,我们将研究限制在只适应下游任务的注意力权重,并冻结MLP模块(因此它们在下游任务中不被训练)。我们在7.1节中进一步研究适应Transformer中不同类型注意力权重矩阵的效果。我们将适应MLP层、LayerNorm层和偏置的实证研究留给未来的工作。

实际好处和局限性。 最显著的好处来自内存和存储使用的减少。对于使用Adam训练的大型Transformer,如果$r \ll d_{model}$,我们将VRAM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VRAM消耗从1.2TB减少到350GB。使用$r = 4$并且只适应查询和值投影矩阵,检查点大小减少了大约10,000倍(从350GB减少到35MB)。这允许我们使用明显更少的GPU进行训练,并避免I/O瓶颈。另一个好处是,我们可以以更低的成本在部署时切换任务,只需交换LoRA权重,而不是所有参数。这允许创建许多自定义模型,可以在将预训练权重存储在VRAM中的机器上即时切换。我们还观察到与全量微调相比,GPT-3 175B的训练速度提高了25%,因为我们不需要计算绝大多数参数的梯度。

LoRA也有其局限性。例如,如果选择将A和B吸收到W中以消除额外的推理延迟,在单个前向传递中批处理不同任务的输入(具有不同的A和B)并不简单。尽管在延迟不关键的场景中,可以不合并权重,并动态选择批次中样本使用的LoRA模块。

1.5. 实证实验

我们在RoBERTa(Liu等人,2019)、DeBERTa(He等人,2021)和GPT-2(Radford等人,b)上评估LoRA的下游任务性能,然后扩展到GPT-3 175B(Brown等人,2020)。我们的实验涵盖了广泛的任务,从自然语言理解(NLU)到生成(NLG)。具体来说,我们在GLUE(Wang等人,2019)基准上评估RoBERTa和DeBERTa。我们遵循Li & Liang(2021)在GPT-2上的设置进行直接比较,并添加WikiSQL(Zhong等人,2017)(自然语言到SQL查询)和SAMSum(Gliwa等人,2019)(对话摘要)用于GPT-3的大规模实验。有关我们使用的数据集的更多详细信息,请参见附录C。我们对所有实验使用NVIDIA Tesla V100。

1.5.1. 基线

为了广泛比较其他基线,我们复制了先前工作使用的设置,并在可能的情况下重用他们报告的数字。然而,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,模型初始化为预训练的权重和偏置,所有模型参数都进行梯度更新。一个简单的变体是只更新一些层,同时冻结其他层。我们包括了先前工作(Li & Liang,2021)在GPT-2上报告的一个这样的基线,它只适应最后两层(FT_{Top2})。

仅偏置或BitFit是一个基线,我们只训练偏置向量,同时冻结其他所有内容。同时期,这个基线也被BitFit(Zaken等人,2021)研究。

前缀嵌入调整(PreEmbed)在输入标记中插入特殊标记。这些特殊标记具有可训练的词嵌入,通常不在模型的词汇表中。这些标记的放置位置可能会影响性能。我们专注于"前缀"(将这些标记前置到提示)和"中缀"(将其附加到提示),这两种方法都在Li & Liang(2021)中讨论过。我们使用l_p(分别为l_i)表示前缀(分别为中缀)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。我们不仅学习一些特殊标记的词嵌入(或等效地,嵌入层之后的激活),而是学习每个Transformer层之后的激活。从前一层计算的激活简单地被可训练的激活替换。结果可训练参数的数量为$|\Theta| = L \times d_{model} \times (l_p + l_i)$,其中L是Transformer层的数量。

适配器调整 如Houlsby等人(2019)所提出的,在自注意力模块(和MLP模块)与后续残差连接之间插入适配器层。适配器层中有两个带有偏置的全连接层,中间有一个非线性。我们称这种原始设计为$Adapter_H$。最近,Lin等人(2020)提出了一种更高效的设计,适配器层仅应用于MLP模块之后和LayerNorm之后。我们称之为$Adapter_L$。这与Pfeiffer等人(2021)提出的另一种设计非常相似,我们称之为$Adapter_P$。我们还包括另一个称为AdapterDrop(Rückle等人,2020)的基线,它删除一些适配器层以提高效率($Adapter_D$)。我们尽可能引用先前工作中的数字,以最大化我们比较的基线数量;它们在第一列带有星号(*)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,$\hat{L}_{LN}$是可训练LayerNorms的数量(例如,在$Adapter_L$中)。

LoRA在现有权重矩阵的并行中添加可训练的秩分解矩阵对。如4.2节所述,为简单起见,在大多数实验中我们只将LoRA应用于W_q和W_v。可训练参数的数量由秩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的权重矩阵的数量。

1.5.2. ROBERTA BASE/LARGE

RoBERTa(Liu等人,2019)优化了最初在BERT(Devlin等人,2019a)中提出的预训练方法,并在不引入太多可训练参数的情况下提升了后者的任务性能。虽然RoBERTa在近年来已被GLUE基准(Wang等人,2019)等NLP排行榜上的更大模型超越,但对于其规模而言,它仍然是从业者中具有竞争力和流行的预训练模型。我们从HuggingFace Transformers库(Wolf等人,2020)中获取预训练的RoBERTa base(125M)和RoBERTa large(355M),并评估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)和Pfeiffer等人(2021)的设置复制了他们的实验。为确保公平比较,我们对LoRA与适配器的比较做了两个关键改变。首先,我们对所有任务使用相同的批量大小,并使用128的序列长度来匹配适配器基线。其次,在适应MRPC、RTE和STS-B时,我们将模型初始化为预训练模型,而不是像微调基线那样使用已经适应MNLI的模型;所有任务的预训练模型保持冻结。遵循这种来自Houlsby等人(2019)的更受限制设置的运行标有†。结果展示在表2(前三节)中。有关使用的超参数的详细信息,请参见D.1节。

1.5.3. DEBERTA XXL

DeBERTa(He等人,2021)是BERT的一个更新变体,经过大规模训练,在GLUE(Wang等人,2019)和SuperGLUE(Wang等人,2020)等基准测试中表现非常有竞争力。我们评估LoRA是否仍能在GLUE上匹配完全微调的DeBERTa XXL(1.5B)的性能。结果展示在表2(底部)中。有关使用的超参数的详细信息,请参见D.2节。

1.5.4. GPT-2 MEDIUM/LARGE

在展示LoRA可以成为NLU全量微调的有竞争力替代方案后,我们希望回答LoRA在NLG模型(如GPT-2 medium和large(Radford等人,b))上是否仍然占优。为了直接比较,我们尽可能保持与Li & Liang(2021)相同的设置。由于篇幅限制,我们在本节中只展示E2E NLG Challenge的结果(表3)。WebNLG(Gardent等人,2017)和DART(Nan等人,2020)的结果请参见F.1节。我们在D.3节中列出了使用的超参数。

1.5.5. 扩展到GPT-3 175B

作为LoRA的最终压力测试,我们扩展到拥有1750亿参数的GPT-3。由于训练成本高昂,我们只报告给定任务在随机种子上的典型标准偏差,而不是为每个条目提供一个标准偏差。有关使用的超参数的详细信息,请参见D.4节。

如表4所示,LoRA在所有三个数据集上匹配或超过了微调基线。注意,并非所有方法都随可训练参数的增加而单调受益,如图2所示。我们观察到,当我们使用超过256个特殊标记进行前缀嵌入调整或超过32个特殊标记进行前缀层调整时,性能显著下降。这证实了Li & Liang(2021)中的类似观察。虽然对这种现象进行彻底调查超出了本工作的范围,但我们怀疑拥有更多特殊标记会导致输入分布进一步偏离预训练数据分布。另外,我们在F.3节中研究了不同适应方法在低数据体系下的性能。

1.6. 相关工作

Transformer语言模型。Transformer(Vaswani等人,2017)是一种大量使用自注意力的序列到序列架构。Radford等人(a)通过使用Transformer解码器堆栈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领域占据主导地位,在许多任务中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,一种新范式形成 - 两者都是在大量文本上训练的大型Transformer语言模型 - 在通用领域数据上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,提供了显著的性能提升。训练更大的Transformer通常会带来更好的性能,这仍然是一个活跃的研究方向。GPT-3(Brown等人,2020)是迄今为止训练的最大单一Transformer语言模型,拥有1750亿参数。

提示工程和微调。虽然GPT-3 175B只需要几个额外的训练样本就可以适应其行为,但结果在很大程度上取决于输入提示(Brown等人,2020)。这就需要一种经验性的艺术来组合和格式化提示,以最大化模型在所需任务上的性能,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等人,2019b; Radford等人,a)。它的变体包括只学习参数的子集(Devlin等人,2019b; Collobert & Weston,2008),但从业者经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的方式进行微调变得具有挑战性,因为它产生的检查点很大,并且由于内存占用与预训练相同,因此硬件门槛很高。

参数高效适应。许多人提出在神经网络的现有层之间插入适配器层(Houlsby等人,2019; Rebuffi等人,2017; Lin等人,2020)。我们的方法使用类似的瓶颈结构来对权重更新施加低秩约束。关键的功能区别在于,我们学习的权重可以在推理过程中与主要权重合并,因此不会引入任何延迟,这对适配器层来说是不可能的(第3节)。适配器的一个同期扩展是COMPACTER(Mahabadi等人,2021),它本质上使用带有一些预定义权重共享方案的Kronecker积来参数化适配器层。同样,将LoRA与其他基于张量积的方法结合可能会提高其参数效率,我们将此留给未来的工作。最近,许多人提出优化输入词嵌入而不是微调,这类似于提示工程的连续和可微分泛化(Li & Liang,2021; Lester等人,2021; Hambardzumyan等人,2020; Liu等人,2021)。我们在实验部分包括了与Li & Liang(2021)的比较。然而,这一系列工作只能通过在提示中使用更多特殊标记来扩展,当学习位置嵌入时,这些标记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。许多机器学习问题具有某种内在的低秩结构(Li等人,2016; Cai等人,2010; Li等人,2018b; Grasedyck等人,2013)。此外,对于许多深度学习任务,特别是那些具有严重过参数化神经网络的任务,已知训练后的神经网络将具有低秩特性(Oymak等人,2019)。一些先前的工作甚至在训练原始神经网络时显式地施加低秩约束(Sainath等人,2013; Povey等人,2018; Zhang等人,2014; Jaderberg等人,2014; Zhao等人,2016; Khodak等人,2021; Denil等人,2014);然而,据我们所知,这些工作都没有考虑对冻结模型进行低秩更新以适应下游任务。在理论文献中,已知当基础概念类具有某种低秩结构时,神经网络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等人,2019; Li & Liang,2018)(Ghorbani等人,2020; Allen-Zhu & Li,2019; Allen-Zhu & Li,2020a)。Allen-Zhu & Li(2020b)的另一个理论结果表明,低秩适应对对抗训练有用。总之,我们相信我们提出的低秩适应更新得到了文献的充分支持。

1.7. 理解低秩更新

鉴于LoRA的经验优势,我们希望进一步解释从下游任务学习的低秩适应的属性。注意,低秩结构不仅降低了硬件门槛,允许我们并行运行多个实验,而且还提供了更好的可解释性,说明更新权重如何与预训练权重相关。我们将研究重点放在GPT-3 175B上,在这里我们实现了可训练参数数量的最大减少(高达10,000倍),而不会对任务性能产生不利影响。

我们进行一系列实证研究来回答以下问题:1)给定参数预算约束,我们应该适应预训练Transformer中的哪些权重矩阵子集以最大化下游性能?2)"最优"适应矩阵ΔW是否真的是秩亏的?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是否与W高度相关?与W相比,ΔW有多大?

我们相信,我们对问题(2)和(3)的回答揭示了使用预训练语言模型进行下游任务的基本原理,这是NLP中的一个关键主题。

1.7.1. 我们应该将LORA应用于TRANSFORMER中的哪些权重矩阵?

给定有限的参数预算,我们应该用LoRA适应哪些类型的权重以获得下游任务的最佳性能?如4.2节所述,我们只考虑自注意力模块中的权重矩阵。我们在GPT-3 175B上设置18M(如果以FP16存储约35MB)的参数预算,这对应于如果我们适应一种类型的注意力权重则r=8,如果我们适应两种类型则r=4,对所有96层。结果在表5中呈现。

注意,将所有参数放在ΔW_q或ΔW_k中会导致显著较低的性能,而同时适应W_q和W_v会产生最佳结果。这表明即使是秩为4的矩阵也能捕获ΔW中足够的信息,以至于适应更多的权重矩阵比用更大的秩适应单一类型的权重更可取。

1.7.2. LORA的最优秩r是多少?

我们将注意力转向秩r对模型性能的影响。我们适应{W_q, W_v}、{W_q, W_k, W_v, W_c}和仅W_q进行比较。

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

Table 6: Validation accuracy on WikiSQL and MultiNLI with different rank r. To our surprise, a rank as small as one suffices for adapting both W_q and W_v on these datasets while training W_q alone needs a larger r. We conduct a similar experiment on GPT-2 in Section H.2.

表6显示,令人惊讶的是,LoRA即使在很小的r下也表现得很有竞争力(对于{W_q, W_v}比仅W_q更明显)。这表明更新矩阵ΔW可能具有非常小的"内在秩"。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的子空间的重叠。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。

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为r=8和r=64的适应矩阵$A_{r=8}$和$A_{r=64}$,我们进行奇异值分解并获得右奇异酉矩阵$U_{A_{r=8}}$和$U_{A_{r=64}}$。我们希望回答:$U_{A_{r=8}}$中前i个奇异向量(对于1 ≤ i ≤ 8)跨越的子空间有多少包含在$U_{A_{r=64}}$中前j个奇异向量(对于1 ≤ j ≤ 64)跨越的子空间中?我们用基于Grassmann距离的归一化子空间相似性来衡量这个量(参见附录G以获得更正式的讨论)

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

其中$U^{i}_{A_{r=8}}$表示$U_{A_{r=8}}$对应于前i个奇异向量的列。

$\phi(\cdot)$的范围为[0, 1],其中1表示子空间完全重叠,0表示完全分离。图3显示了当我们改变i和j时ϕ如何变化。由于篇幅限制,我们只看第48层(共96层),但结论对其他层也成立,如H.1节所示。

我们从图3中得出一个重要观察:

$A_{r=8}$和$A_{r=64}$对应于顶部奇异向量的方向显著重叠,而其他方向则不然。具体来说,$A_{r=8}$的ΔW_v(分别是ΔW_q)和$A_{r=64}$的ΔW_v(分别是ΔW_q)共享一个归一化相似度>0.5的维度为1的子空间,这解释了为什么r=1在我们的GPT-3下游任务中表现得相当好。

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表明$A_{r=8}$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声。因此,适应矩阵确实可以有很低的秩。

不同随机种子之间的子空间相似性。我们通过绘制r=64的两次随机种子运行之间的归一化子空间相似性来进一步确认这一点,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更多共同的奇异值方向,这与我们在表6中的经验观察一致。作为比较,我们还绘制了两个随机高斯矩阵,它们之间不共享任何共同的奇异值方向。

> "SVD将一个矩阵A分解为$U\Sigma V^T$, 其中U和V是酉矩阵, Σ是对角矩阵。右奇异酉矩阵V包含了A^TA特征向量, 这些特征向量按对应的奇异值大小排序, 代表了矩阵A在不同方向上的"重要性"。$U_{A_{r=8}}$的前i个列向量跨越一个i维子空间, $U_{A_{r=64}}$的前j个列向量跨越一个j维子空间, Grassmann距离衡量两个子空间之间的"角度", 范围在$[0,1]$之间, 1表示子空间完全重叠, 0表示完全正交。
>
> 图3左两列(ΔW_q和ΔW_v)展示了$r=8$和$r=64$之间的子空间相似性，右两列是左两列左下角的放大图，颜色越亮表示相似度越高。可以观察到:
> a. 左上角(小i和j)的颜色最亮, 表示顶部奇异向量之间的高度重叠。
> b. 随着i和j增加, 相似度逐渐降低(颜色变暗)。

> c. ΔW_v比ΔW_q呈现出更高的相似度。这说明$r = 8$和$r = 64$的适应矩阵共享最重要的几个方向,而其他方向可能包含更多随机噪声。”

1.7.3. 适应矩阵ΔW与W相比如何?

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?(或者从数学上讲,ΔW是否主要包含在W的顶部奇异方向中?)另外,与W中相应的方向相比,ΔW有多"大"?这可以揭示适应预训练语言模型的潜在机制。

为了回答这些问题,我们通过计算$U^T W V^T$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$\|U^T W V^T\|_F$和$\|W\|_F$之间的Frobenius范数。作为比较,我们还通过用W或随机矩阵的前r个奇异向量替换U,V来计算$\|U^T W V^T\|_F$。

	$r = 4$			$r = 64$		
	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 7: The Frobenius norm of $U^\top W_q V^\top$ where U and V are the left/right top r singular vector directions of either (1) ΔW_q, (2) W_q, or (3) a random matrix. The weight matrices are taken from the 48th layer of GPT-3.

> 第一列 (ΔW_q, r=4): 0.32 这里 U 和 V 来自 ΔW_q 的前4个左/右奇异向量。所以这是 W_q 投影到 ΔW_q 的前4个奇异向量张成的子空间上。
>
> 第二列 (W_q, r=4): 21.67 这里 U 和 V 来自 W_q 自身的前4个左/右奇异向量。这是 W_q 投影到其自身的前4个奇异向量张成的子空间上。
>
> 第三列 (Random, r=4): 0.02 这里 U 和 V 来自一个随机矩阵的前4个左/右奇异向量。这是 W_q 投影到随机矩阵的前4个奇异向量张成的子空间上。

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有更强的相关性,表明ΔW放大了W中已经存在的一些特征(因为 ΔW与W相关性强于随机矩阵： 0.32 > 0.02（r=4时））。其次,ΔW并不是重复W的顶部奇异方向,而是只放大W中未被强调的方向（0.32 << 21.67（r=4时））。第三,放大因子相当大:对于r=4,21.5≈6.91/0.32。参见H.4节了解为什么r=64的放大因子较小。我们还在H.3节中提供了一个可视化,说明当我们包含更多来自W_q的顶部奇异方向时相关性如何变化。

这表明低秩适应矩阵可能放大了特定下游任务的重要特征,这些特征在通用预训练模型中已经学习但未被强调。

> 对于r=4的情况，放大因子计算如下：
> 放大因子 $\approx \frac{\|\Delta W_q\|_F}{\|U^T W_q V^T\|_F} \approx \frac{6.91}{0.32} \approx 21.5$
>
> 解释：
>
> 1. $\|\Delta W_q\|_F = 6.91$：这是LoRA更新矩阵ΔW_q的Frobenius范数，表示LoRA引入的总体变化大小。
> 2. $\|U^T W_q V^T\|_F = 0.32$：这是原始权重矩阵W_q在ΔW_q的前r个奇异向量定义的子空间上的投影的Frobenius范数。它表示W_q在LoRA关注的方向上的分量大小。
> 3. 放大因子的意义：
>   - 它衡量了LoRA引入的变化(ΔW_q)相对于原始权重(W_q)在这些方向上的分量的比例。

> - 高放大因子（如21.5）意味着LoRA在这些特定方向上引入了显著大于原始权重在这些方向上分量的变化。

1.8. 结论和未来工作

微调巨大的语言模型在硬件需求以及存储/切换不同任务独立实例的成本方面是极其昂贵的。我们提出了LoRA,这是一种高效的适应策略,它既不引入推理延迟,也不减少输入序列长度,同时保持高模型质量。重要的是,它允许在部署为服务时快速切换任务,因为绝大多数模型参数是共享的。虽然我们专注于Transformer语言模型,但提出的原则普遍适用于任何包含密集层的神经网络。

未来工作有许多方向。1) LoRA可以与其他高效适应方法结合,可能提供正交的改进。2) 微调或LoRA背后的机制还远未明确 - 预训练期间学习的特征是如何转化为在下游任务上表现良好的?我们相信LoRA比全量微调更容易回答这个问题。3) 我们主要依靠启发式方法来选择应用LoRA的权重矩阵。是否有更有原则的方法来做这个选择?4) 最后,ΔW的秩缺陷表明W本身可能也是秩缺陷的,这也可以成为未来工作的灵感来源。

1.9. A 大型语言模型仍然需要参数更新

少样本学习,或提示工程,在我们只有少量训练样本时非常有优势。然而,在实践中,我们通常可以为性能敏感的应用收集几千或更多的训练示例。如表8所示,与少样本学习相比,微调在大小数据集上都显著提高了模型性能。我们从GPT-3论文(Brown等人,2020)中取了GPT-3在RTE上的少样本结果。对于MNLI-matched,我们对每个类别使用两个演示和总共六个上下文示例。

1.10. 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,而我们的提议LoRA可以被视为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的延迟。如Rückle等人(2020)指出,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,特别是在批量大小小的在线推理中,增加的延迟可能是显著的。

我们通过在100次试验中取平均来测量NVIDIA Quadro RTX8000上单次前向传递的延迟。我们改变输入批量大小、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:Houlsby等人(2019)提出的原始设计,我们称之为$Adapter_H$,以及Lin等人(2020)最近提出的更高效变体,我们称之为$Adapter_L$。有关设计的更多细节,请参见5.1节。我们在图5中绘制了与无适配器基线相比的百分比减速。

1.11. C 数据集详情

GLUE基准是一个广泛的自然语言理解任务集合。它包括MNLI(推理,Williams等人(2018))、SST-2(情感分析,Socher等人(2013))、MRPC(释义检测,Dolan & Brockett(2005))、CoLA(语言可接受性,Warstadt等人(2018))、QNLI(推理,Rajpurkar等人(2018))、QQP(问答)、RTE(推理)和STS-B(文本相似性,Cer等人(2017))。广泛的覆盖使GLUE基准成为评估NLU模型(如RoBERTa和DeBERTa)的标准指标。各个数据集以不同的许可证发布。

WikiSQL由Zhong等人(2017)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格模式生成SQL查询。我们将上下文编码为x = {表格模式,查询},目标编码为y = {SQL}。该数据集以BSD 3-Clause许可证发布。

SAMSum由Gliwa等人(2019)引入,包含14,732/819个训练/测试示例。它由两个人之间的分阶段聊天对话和语言学家编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",目标编码为y = {摘要}。该数据集以非商业许可Creative Commons BY-NC-ND 4.0发布。

E2E NLG挑战赛最初由Novikova等人(2017)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,来自餐厅领域。每个用作输入的源表可以有多个参考。每个样本输入(x, y)由一系列槽值对组成,以及相应的自然语言参考文本。该数据集以Creative Commons BY-NC-SA 4.0许可发布。

DART是Nan等人(2020)描述的开放域数据到文本数据集。DART输入结构化为ENTITY — RELATION — ENTITY三元组序列。DART总共有82K个示例,是一个明显更大更复杂的数据到文本任务,相比E2E。该数据集以MIT许可证发布。

WebNLG是另一个常用于数据到文本评估的数据集(Gardent等人,2017)。WebNLG总共有22K个示例,包括14个不同的类别,其中9个在训练期间可见。由于14个类别中的5个在训练期间未见,但在测试集中有代表,因此评估通常分为"已见"类别(S)、"未见"类别(U)和"全部"(A)。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许可发布。

1.12. D 实验中使用的超参数

1.12.1. D.1 ROBERTA

我们使用带线性学习率衰减计划的AdamW进行训练。我们针对LoRA调整学习率、训练轮数和批量大小。遵循Liu等人(2019)的做法,当适应MRPC、RTE和STS-B任务时,我们用最佳的MNLI检查点来初始化LoRA模块,而不是通常的初始化方式;预训练模型在所有任务中都保持冻结。我们报告5个随机种子的中位数;每次运行的结果取自最佳epoch。为了与Houlsby等人(2019)和Pfeiffer等人(2021)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量大小。重要的是,当适应MRPC、RTE和STS-B时,我们从预训练的RoBERTa large模型开始,而不是已经适应MNLI的模型。这种受限设置的运行标记为†。表9中列出了我们运行中使用的超参数。

1.12.2. D.2 DEBERTA

我们再次使用带线性学习率衰减计划的AdamW进行训练。遵循He等人(2021)的做法,我们调整学习率、dropout概率、预热步数和批量大小。我们使用He等人(2021)使用的相同模型序列长度,以保持比较的公平性。遵循He等人(2021)的做法,当适应MRPC、RTE和STS-B任务时,我们用最佳的MNLI检查点来初始化LoRA模块,而不是通常的初始化方式;预训练模型在所有任务中都保持冻结。我们报告5个随机种子的中位数;每次运行的结果取自最佳epoch。表10中列出了我们运行中使用的超参数。

1.12.3. D.3 GPT-2

我们使用AdamW(Loshchilov & Hutter, 2017)对所有GPT-2模型进行训练,使用线性学习率计划训练5个epoch。我们使用Li & Liang(2021)描述的批量大小、学习率和束搜索束大小。相应地,我们也为LoRA调整上述超参数。我们报告3个随机种子的平均值;每次运行的结果取自最佳epoch。表11列出了GPT-2中LoRA使用的超参数。对于其他基线使用的超参数,请参见Li & Liang(2021)。

1.12.4. D.4 GPT-3

对于所有GPT-3实验,我们使用AdamW(Loshchilov & Hutter, 2017)训练2个epoch,批量大小为128个样本,权重衰减因子为0.1。我们对WikiSQL(Zhong et al., 2017)使用384的序列长度,对MNLI(Williams et al., 2018)使用768,对SAMSum(Gliwa et al., 2019)使用2048。我们针对所有方法-数据集组合调整学习率。有关使用的超参数的更多详细信息,请参见D.4节。对于前缀嵌入调优,我们发现最佳的l_p和l_i分别为256和8,总共有3.2M可训练参数。对于前缀层调优,我们使用$l_p = 8$和$l_i = 8$,有20.2M可训练参数,以获得总体最

佳性能。我们展示了LoRA的两种参数预算:4.7M($r_q = r_v = 1$或$r_v = 2$)和37.7M($r_q = r_v = 8$或$r_q = r_k = r_v = r_o = 2$)。我们报告每次运行的最佳验证性能。表12列出了我们GPT-3实验中使用的训练超参数。

1.13. 将LORA与PREFIX TUNING结合

LoRA可以自然地与现有的基于前缀的方法结合。在本节中,我们评估LoRA与前缀调优变体的两种组合在WikiSQL和MNLI上的表现。

LoRA+PrefixEmbed(LoRA+PE)将LoRA与前缀嵌入调优相结合,我们插入$l_p + l_i$个特殊令牌,其嵌入被视为可训练参数。有关前缀嵌入调优的更多信息,请参见5.1节。

LoRA+PrefixLayer(LoRA+PL)将LoRA与前缀层调优相结合。我们也插入$l_p + l_i$个特殊令牌;然而,我们不是让这些令牌的隐藏表示自然演化,而是在每个Transformer块之后用与输入无关的向量替换它们。因此,嵌入和后续Transformer块激活都被视为可训练参数。有关前缀层调优的更多信息,请参见5.1节。

在表15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上的表现显著优于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本身已经达到了与人类基线相当的性能。其次,我们注意到LoRA+PL的表现略差于LoRA,即使有更多的可训练参数。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权重的优化更加困难。

Method	Hyperparameters	# Trainable Parameters	WikiSQL	MNLI-m
Fine-Tune	-	175B	73.8	89.5
PrefixEmbed	$l_p = 32, l_i = 8$	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
PrefixLayer	$l_p = 2, l_i = 2$	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
$\text{Adapter}^{\text{H}}$	$r = 1$	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 15: Hyperparameter analysis of different adaptation approaches on WikiSQL and MNLI. Both prefix-embedding tuning (PrefixEmbed) and prefix-layer tuning (PrefixLayer) perform worse as we increase the number of trainable parameters, while LoRA's performance stabilizes. Performance is measured in validation accuracy.

1.14. F 额外的实证实验

1.14.1. F.1 GPT-2上的额外实验

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。结果如表13所示。与我们在第5节中报告的E2E NLG Challenge的结果类似,给定相同数量的可训练参数,LoRA的表现优于或至少与基于前缀的方法相当。

1.14.2. F.2 GPT-3上的额外实验

我们在表15中展示了GPT-3上不同适应方法的额外运行结果。重点是识别性能和可训练参数数量之间的权衡。

1.14.3. F.3 低数据量情况

为了评估不同适应方法在低数据量情况下的性能,我们从MNLI的完整训练集中随机抽样100、1k和10k个训练样本,形成低数据量MNLI-n任务。在表16中,我们展示了不同适应方法在MNLI-n上的性能。令人惊讶的是,PrefixEmbed和PrefixLayer在MNLI-100数据集上表现非常差,PrefixEmbed的表现仅略优于随机猜测(37.6%对33.3%)。PrefixLayer的表现优于PrefixEmbed,但在MNLI-100上仍然显著差于Fine-Tune或

LoRA。随着我们增加训练样本数,基于前缀的方法与LoRA/Fine-tuning之间的差距变小,这可能表明基于前缀的方法不适合GPT-3中的低数据量任务。在MNLI-100和MNLI-Full上,LoRA的性能优于微调,在MNLI-1k和MNLI-10K上,考虑到由于随机种子导致的(±0.3)方差,LoRA达到了可比的结果。

表17报告了不同适应方法在MNLI-n上的训练超参数。我们在MNLI-100集上为PrefixLayer使用较小的学习率,因为使用较大的学习率时训练损失不会下降。

1.15. G 测量子空间之间的相似度

在本文中,我们使用度量$\phi(A, B, i, j) = \psi(U_A^i, U_B^j) = \frac{\|U_A^{i\top} U_B\|_F^2}{\min\{i,j\}}$来衡量两个列正交矩阵$U_A^i \in \mathbb{R}^{d\times i}$和$U_B^j \in \mathbb{R}^{d\times j}$之间的子空间相似度,这两个矩阵是通过取A和B的左奇异矩阵的列得到的。我们指出,这种相似度只是衡量子空间之间距离的标准投影度量的反向(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:

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

而我们的相似度定义为:

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

这个相似度满足: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,$\phi(A, B, i, j) \in (0, 1)$。

1.16. H 低秩矩阵的额外实验

我们展示了对低秩更新矩阵调查的额外结果。

1.16.1. H.1 LORA模块之间的相关性

见图6和图7,展示了图3和图4中呈现的结果如何推广到其他层。

1.16.2. 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(7.2节)。以E2E NLG Challenge数据集为例,我们报告了训练26,000步后不同r选择达到的验证损失和测试指标。我们的结果在表18中呈现。GPT-2 Medium的最佳秩在4到16之间,取决于使用的指标,这与GPT-3 175B的情况类似。注意,模型大小与适应的最佳秩之间的关系仍然是一个开放的问题。

1.16.3. H.3 W和ΔW之间的相关性

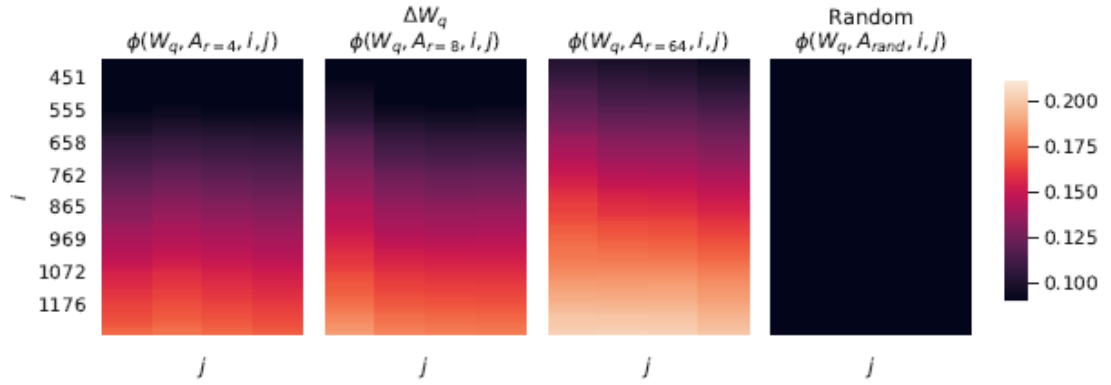


Figure 8: Normalized subspace similarity between the singular directions of W_q and those of ΔW_q with varying r and a random baseline. ΔW_q amplifies directions that are important but not emphasized in W. ΔW with a larger r tends to pick up more directions that are already emphasized in W.

见图8,展示了W和ΔW之间随r变化的归一化子空间相似度。

再次注意,ΔW不包含W的顶部奇异方向,因为ΔW中前4个方向与W中前10%方向的相似度几乎不超过0.2。这为ΔW包含那些在W中未被强调的"特定任务"方向提供了证据。

接下来要回答的一个有趣问题是,我们需要将这些特定任务的方向放大多少,才能使模型适应良好工作?

1.16.4. H.4 放大因子

人们自然可以考虑一个特征放大因子,即比率$\frac{\|\Delta W\|_F}{\|U^\top W V^\top\|_F}$,其中U和V是ΔW的SVD分解的左右奇异矩阵。(回想一下,UU^TWV^TV给出了W在ΔW所张成的子空间上的"投影"。)

直观地说,当ΔW主要包含特定任务的方向时,这个数量衡量了它们被ΔW放大了多少。如7.3节所示,对于r=4,这个放大因子高达20。换句话说,在每一层中(从预训练模型W的整个特征空间中)通常有四个特征方向需要被放大很大的因子20,以达到我们报告的特定下游任务的准确率。而且,对于每个不同的下游任务,应该期望一组非常不同的特征方向被放大。

然而,人们可能注意到,对于r=64,这个放大因子只有2左右,这意味着在r=64时ΔW中学到的大多数方向并没有被放大太多。这不应该令人惊讶,事实上再次证明了表示"特定任务方向"(因此用于模型适应)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更大的因子20。