LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

0.概要

自然语言处理的一个重要范式是基于通用领域数据的大规模预训练，然后适应特定任务或领域。随着预训练的模型变得越来越大，全面的微调（重新训练所有模型参数）变得越来越不可行。以GPT-3 175B为例,部署每个有1750亿参数的独立微调模型代价非常高昂。

我们提出了低秩适配（Low-Rank Adaptation，简称LoRA），该方法冻结预训练模型的权重，并向Transformer架构的每一层注入可训练的低秩分解矩阵，从而大大减少了下游任务中的可训练参数数量。与使用Adam微调的GPT-3 175B相比，LoRA可以减少1万倍的可训练参数，并将GPU内存需求减少3倍。LoRA在RoBERTa、DeBERTa、GPT-2和GPT-3上的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，并且与适配器不同，它不会增加推理延迟。我们还提供了对语言模型适配中秩不足现象的实证研究，揭示了LoRA的有效性。

1.介绍

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多个下游应用。这样的适应通常通过微调实现，即更新预训练模型的所有参数。微调的主要缺点在于，新模型的参数数量与原始模型相同。随着越来越大的模型每隔几个月就被训练出来，这种情况从GPT-2的"轻微不便"演变为GPT-3（2020）中具有1750亿可训练参数的重大部署挑战。

许多人试图通过仅适应部分参数或为新任务学习外部模块来缓解这一问题。这样，我们只需存储和加载少量特定任务的参数，以及加载预训练模型，从而大大提高了部署时的操作效率。然而，现有的技术通常会通过增加模型深度或减少模型的可用序列长度，引入推理延迟。更重要的是，这些方法通常无法与微调基线相匹配，在效率与模型质量之间存在权衡。

我们发现学习到的过参数化模型实际上处于低固有维度上。我们假设模型适应过程中权重的变化也具有低"固有秩"，由此提出了我们的低秩适配（LoRA）方法。LoRA使我们能够通过优化适应过程中稠密层变化的秩分解矩阵来间接训练神经网络中的一些稠密层，同时保持预训练权重冻结。以GPT-3 175B为例，我们展示了即使当全秩（即d）高达12,288时，非常低的秩（即图1中的r可以是1或2）也足够，这使得LoRA在存储和计算上都非常高效。

LoRA的关键优势：

一个预训练模型可以共享并用于构建许多不同任务的小型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，从而显著减少存储需求和任务切换的开销。

LoRA使训练更高效，并通过使用自适应优化器将硬件门槛降低多达3倍

我们不需要计算大部分参数的梯度或维护优化器状态。相反，我们只优化注入的、较小的低秩矩阵。

不引入任何推理延迟

我们简单的线性设计允许在部署时将可训练矩阵与冻结权重合并

LoRA与许多现有方法正交

可以与它们结合，例如前缀调优。

术语与惯例

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dmodel。我们用Wq、Wk、Wv和Wo表示自注意力模块中的查询/键/值/输出投影矩阵。W或W0指的是预训练权重矩阵，ΔW表示适应过程中累积的梯度更新。我们用r表示LoRA模块的秩。我们遵循惯例，并使用Adam进行模型优化，Transformer的MLP前馈维度$d_{ffn} = 4 × d_{model}$。

过参数化

模型中使用的参数比训练数据集可以有效支持的更多，有可能捕捉到噪声并导致过拟合

内在维度

特征中存在的有效信息的维度，可能远小于特征的实际维度。内在维度指的是我们解决这个问题实际上需要的参数空间的维度，我们对模型的微调通常调整的也是这些低秩的内在维度。

2 问题陈述

虽然我们的提议对训练目标是不可知的，但我们专注于语言建模作为我们的激励用例。以下是对语言建模问题的简要描述，特别是给定特定任务提示时条件概率最大化的问题。

假设我们有一个预训练的自回归语言模型 $P_{\Phi}(y|x)$，其参数为 Φ。例如，$P_{\Phi}(y|x)$ 可以是一个基于Transformer架构的通用多任务学习器，如 GPT。考虑将该预训练模型适配到下游的条件文本生成任务，如摘要生成、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： $\mathcal{Z} = {(x_i, y_i)}_{i=1,\dots,N}$，其中 x_i 和 y_i 是一组 token 序列。例如，在 NL2SQL 中，x_i 是自然语言查询，而 y_i 是对应的 SQL 命令；对于摘要生成，x_i 是一篇文章的内容，y_i 是其摘要。

在完全微调过程中，模型被初始化为预训练权重 Φ_0，然后更新为 $\Phi_0 + \Delta \Phi$，通过反复遵循梯度来最大化条件语言建模目标：

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

完全微调的主要缺点在于，对于每个下游任务，我们学习一组不同的参数 $\Delta \Phi$，其维度大小等于 $|\Phi_0|$。因此，如果预训练模型很大（如 GPT-3，$|\Phi_0| \approx 175$ 亿），存储和部署许多独立的微调模型可能会很困难，甚至无法实现。

在本文中，我们采用了一种更具参数效率的方法，其中任务特定的参数增量 $\Delta \Phi = \Delta \Phi(\Theta)$ 进一步由一组较小规模的参数 Θ 编码，且 $|\Theta| \ll |\Phi_0|$。因此，寻找 $\Delta \Phi$ 的任务变成了优化 Θ：

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

我们使用一种低秩表示来编码 $\Delta \Phi$，这在计算和内存方面都是高效的。当预训练模型是 GPT-3 175B 时，可训练的参数数量 $|\Theta|$ 仅为 $|\Phi_0|$ 的 0.01%。

3 现有解决方案的问题

自从迁移学习的概念诞生以来，已经有许多工作致力于使模型的适应更高效。以语言模型为例，目前有两种主要的高效适应策略：添加适配器层，或优化某种形式的输入层激活。然而，这两种策略在大规模且对延迟敏感的生产环境中都有其局限性。

适配器层会引入推理延迟

适配器层有许多变体。Houlsby等（2019）提出的原始设计在每个Transformer块中引入了两个适配器层;Lin等（2020）提出的设计，在每个块中只引入了一个适配器层，但增加了LayerNorm。 虽然可以通过剪枝或利用多任务设置来减少总体延迟，但并没有直接的方法绕过适配器层所增加的计算。由于适配器层设计的初衷是通过小瓶颈维度减少参数（有时不到原模型的1%），因此它们能增加的FLOPs（浮点运算数）非常有限。然而，大型神经网络依赖硬件并行性来保持低延迟，而适配器层必须顺序处理。这在在线推理场景下（通常批量大小只有一个）尤其明显。在没有模型并行的常规场景中，例如在单个GPU上运行GPT-2中型模型，我们发现即使瓶颈维度非常小，使用适配器也会显著增加延迟。
当我们需要share模型时，这个问题变得更加严重，因为额外的深度需要更多同步的GPU操作，如$AllReduce$和$Broadcast$，除非我们多次冗余存储适配器参数。

直接优化提示困难重重

另一种方向是以 前缀调优 为代表的方式，它面临不同的挑战。我们观察到前缀调优难以优化，并且其性能在可训练参数上呈现非单调变化，原始论文中也有类似的观察结果。更根本的是，保留一部分序列长度用于适应必然减少了可用于处理下游任务的序列长度，这可能是提示调优性能不如其他方法的原因。我们将在第5节中进一步研究任务性能。

4 我们的方法 LoRA的简单设计及实用优势

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稠密层，这些层中的权重矩阵通常具有满秩。当适应于特定任务时，Aghajanyan等人表明预训练语言模型具有低的“内在维度”，即使经过随机投影到更小的子空间，它仍然能够高效地学习。

受此启发，我们假设在模型适应的过程中，权重的更新也具有低的“内在秩”。对于一个预训练的权重矩阵 $W_0 \in R^{d \times k}$，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 $B \in R^{d \times r}$，$A \in R^{r \times k}$，并且秩 $r \ll \min(d, k)$。在训练过程中，W_0被冻结，不接收梯度更新，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，且其各自的输出向量逐坐标相加。对于 $h = W_0x$，我们修改后的前向传播公式为：

$h = W_0x + \Delta W x = W_0x + BAx$

图1中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，因此训练开始时 $\Delta W = BA$ 为零。然后我们通过因子 α / r 对 $\Delta W x$ 进行缩放，其中 α 是 r 中的一个常数。当使用Adam优化时，如果我们适当缩放初始化，那么调整 α 大致等同于调整学习率。因此，我们简单地将 α 设置为我们尝试的第一个 r，而无需进一步调整它。这种缩放有助于减少在变化 r 时重新调整超参数的需求。

完全微调的推广

更为一般的微调形式允许训练部分预训练参数。LoRA进一步迈出了一步，不要求在适应过程中累积的梯度更新对权重矩阵具有满秩。这意味着，当LoRA应用于所有权重矩阵并训练所有偏置时，我们可以通过将LoRA的秩 r 设置为预训练权重矩阵的秩，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，LoRA的训练大致收敛于原始模型的训练，而基于适配器的方法收敛于MLP，基于前缀的方法则收敛于无法处理长输入序列的模型。

没有额外的推理延迟

在实际部署中，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。请注意，W_0 和 BA 都位于 $R^{d \times k}$ 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 来恢复 W_0，然后添加不同的 B_0A_0，这是一种快速操作，几乎不会产生内存开销。重要的是，偏置参数的数量相对于权重来说是微不足道的。

4.2 将LoRA应用于Transformer

原则上，我们可以将LoRA应用于神经网络中的任何权重矩阵子集，以减少可训练参数的数量。在Transformer架构中，注意力模块中有四个权重矩阵（Wq, Wk, Wv, Wo），MLP模块中有两个权重矩阵。我们将Wq（或Wk, Wv）视为一个维度为 $d_{\text{model}} \times d_{\text{model}}$ 的单一矩阵，尽管输出维度通常被划分到注意力头中。为了简化和提高参数效率，我们仅对下游任务中调整注意力权重，而冻结MLP模块（即在下游任务中不训练它们）。

实际的好处与限制

最显著的好处是内存和存储使用量的减少。对于使用Adam训练的大型Transformer，如果 $r \ll d_{\text{model}}$，我们可以将VRAM使用量减少至三分之二，因为我们不需要为冻结的参数存储优化器状态。在GPT-3 175B模型中，我们将训练时的VRAM消耗从1.2TB减少到350GB。在 $r = 4$ 并且仅调整查询和值投影矩阵的情况下，检查点的大小大约减少了1万倍（从350GB减少到35MB）。这使得我们可以用显著更少的GPU进行训练，并避免I/O瓶颈。

另一个好处是，在实际部署时，我们可以通过仅交换LoRA权重，而不是所有参数，来以更低的成本在任务之间切换。这允许我们在存储了预训练权重的机器上即时切换多个定制模型。此外，我们观察到，在GPT-3 175B上，LoRA相比于完全微调可以加快训练速度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，那么很难在单次前向传播中对不同任务的输入进行批处理。然而，在不要求低延迟的场景中，可以不合并权重，并为批次中的样本动态选择要使用的LoRA模块。

5 实证实验

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机制的序列到序列架构。Radford 等人（a）将其应用于自回归语言建模，使用了一堆Transformer解码器。从那时起，基于Transformer的语言模型在NLP领域占据了主导地位，且在许多任务中实现了最先进的成果。随着BERT（Devlin 等人，2019b）和GPT-2（Radford 等人，b）的出现，一种新的范式逐渐流行——这两者都是在大量文本上训练的大型Transformer语言模型。通过在通用领域数据上预训练后，再在任务特定数据上进行微调，这与直接在任务特定数据上训练相比，带来了显

著的性能提升。训练更大的Transformer模型通常会带来更好的表现，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个Transformer语言模型，拥有1750亿个参数。

Prompt工程与微调

虽然GPT-3 175B可以仅通过几个额外的训练示例来调整其行为，但其结果严重依赖于输入提示。这促使了一种经验性的技巧，即编写和格式化提示，以最大化模型在特定任务上的性能，这被称为prompt工程或prompt调整。

微调是指将预训练于通用领域的模型重新训练以适应特定任务。微调的变体包括仅学习一部分参数，但从业者通常会重新训练所有参数，以最大化下游任务的性能。然而，由于GPT-3 175B模型的庞大性，使得以常规方式进行微调变得具有挑战性，因为它生成的检查点很大，并且由于其与预训练相同的内存占用，导致硬件门槛较高。

参数高效的适配

许多研究提出了在神经网络的现有层之间插入适配层（adapter layers）（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，对权重更新施加了低秩约束。*关键的功能差异在于，我们学习的权重可以在推理过程中与主权重合并，因此不会引入任何延迟，而适配层并不能做到这一点*。适配层的一个当代扩展是COMPACTER，本质上是通过Kronecker积和一些预定的权重共享方案来参数化适配层。类似地，结合LoRA与其他基于张量积的方法可能会进一步提高其参数效率，这点我们留待未来工作进行探讨。更近期的研究提出了优化输入词嵌入来替代微调，类似于一种连续可微的prompt工程泛化（Li & Liang，2021；Lester 等人，2021；Hambardzumyan 等人，2020；Liu 等人，2021）。我们在实验部分包含了与Li & Liang（2021）的比较。然而，这一系列工作只能通过使用更多的特殊标记来扩展，而当位置嵌入是可学习时，这些特殊标记会占用任务标记可用的序列长度。

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。此外，已知对于许多深度学习任务，特别是那些具有高度过参数化的神经网络，训练后学习到的神经网络将表现出低秩特性。一些先前的工作甚至在训练原始神经网络时显式地施加了低秩约束；然而，据我们所知，没有这些工作考虑到对冻结的模型进行低秩更新以适应下游任务。在理论文献中，已知当底层概念类具有某种低秩结构时，神经网络在性能上优于其他传统学习方法，包括相应的（有限宽度的）神经切线核。Ghorbani 等人（2020）；Allen-Zhu & Li（2019）；Allen-Zhu & Li（2020a）指出，低秩结构的理论表明，低秩适应可以对抗对抗性训练产生帮助。总之，我们认为我们提出的低秩适应更新得到了文献的充分支持。