nlp 论文 2023 11月18- PEFT（Parameter-Efficient Fine-Tuning）

一.LoRA Low-Rank Adaptation of Large Language Models（在RoBERTa预训练模型上做的,）

1.引入的原因

（1）模型越来越大，其预训练的参数也越来越大，微调的参数越来越多，导致成本费用变得贵，所以我们引入一个微调模型来解决这一个问题。
（2）之前也有人在这一方面展开过研究但是他们有推理延迟（扩展模型深度或减少模型的可用序列长度），这些模型最关键的还是无法匹配基准基线，本文提出的LoRA很好的解决了这一点。（在之前的工作中，引入了适配器模型。适配器模型是一种参数和计算效率更高的方法，用于模型的自适应。它通过在Transformer块中添加适配器层或优化输入层激活的某些形式来实现。然而，适配器模型也存在一些限制，特别是在大规模和延迟敏感的生产场景中。适配器层会引入推理延迟，并且在在线推理设置中可能会导致显著的延迟增加。另一种方法是直接优化提示，但这种方法很难进行优化，并且在可训练参数的数量上表现不稳定）

2.LoRA的构造

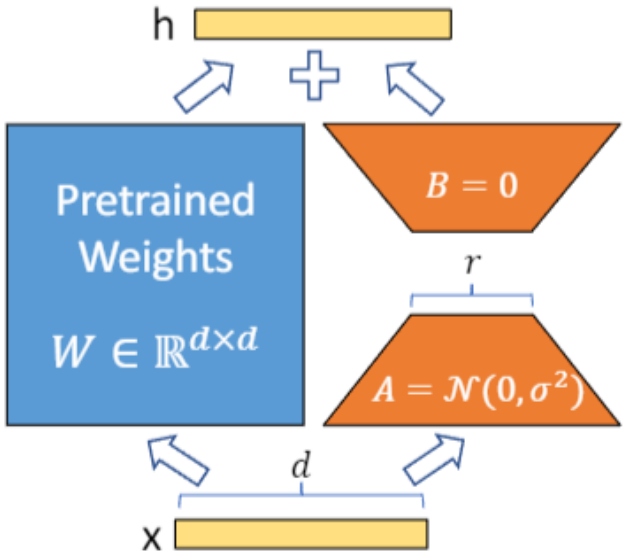


Figure 1: Our reparametrization. We only train A and B.

如图，其是通过把预训练模型的参数给冻结，然后使用几个小LoRA模型来对模型进行微调，小模型矩阵使用的秩r比原始的要小。
其模型运行过程的细节是预训练模型的权重参数维度用全连接层映射到训练层，训练层是在每个transformer块上注入进去的，然后让训练层进行训练，等所有的transformer块都训练好之后再用全连接层进行映射让其维度与预训练模型的权重参数维度一样，最后两者结合起来预测下游任务的输出。（公式是W = W0 + BA W0是预训练模型的权重，B和A都是可以训练的权重，训练过程中A和B发生变化，W0不变）
LoRA 是一种通过对模型权重矩阵进行“剪枝”的方式来控制模型复杂度的方法，它不对模型进行修改。具体来说，LoRA 通过将低秩的权重矩阵分解为更小的子矩阵，从而减少模型的参数数量，同时保持模型的性能。这种方法的好处是可以在不改变模型结构的情况下降低模型的复杂度

3.LoRA优势

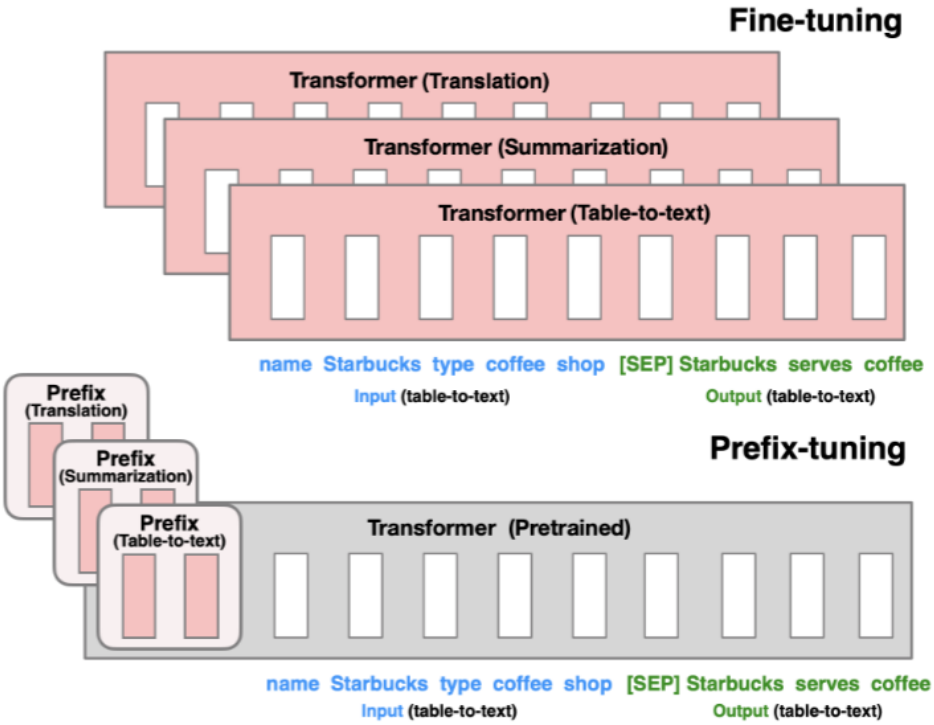
（1）预训练的模型可以共享，并用于为不同的任务构建许多小型LoRA模块
（2）当使用自适应优化器时，LoRA使训练更有效，并将硬件进入障碍降低3倍，因为不需要计算梯度或维护大多数参数的优化器状态。相反，只优化注入的小得多的低秩矩阵
（3）简单的线性设计允许在部署时将可训练矩阵与冻结权重合并，与完全微调的模型相比，通过构造，不会引入推理延迟

4.实验部分

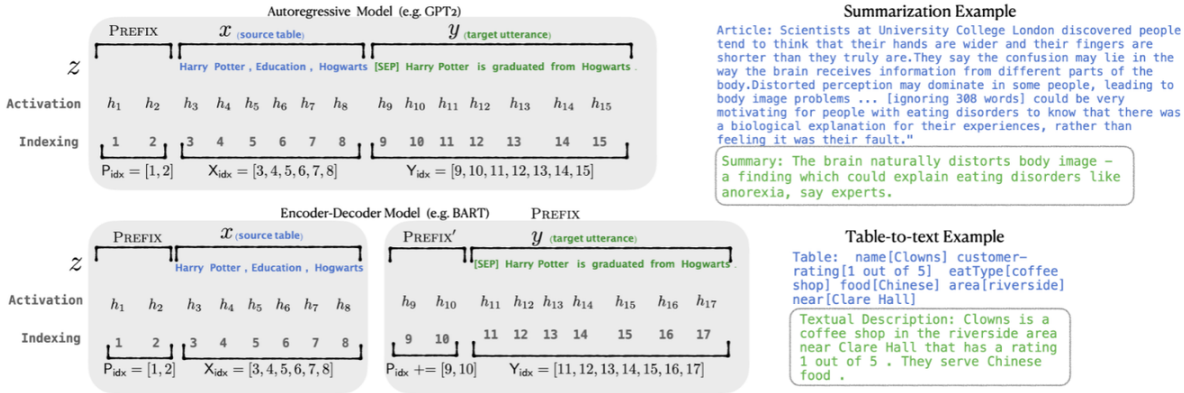
（1）模型讨论了选择transformer里面的哪些权重矩阵进行修改效果会比较好。（文章主要关注了自注意模块中的权重矩阵）
（2）讨论了选了低秩更新权重矩阵中秩的选择。

二.Prefix-Tuning Optimizing Continuous Prompts for Generation（针对NLG，在bert gpt2上做的）

1.模型的组成



上图红色的是全微调，下面是本文提出的prefix前缀



上图是gpt2，bert加入prefix的图

（1）预训练语言模型（Pretrained Language Model）：Prefix-Tuning使用一个大规模的预训练语言模型作为基础模型，例如GPT-2或BART。这些预训练语言模型已经在大规模通用语料上进行了训练，具有强大的语言理解和生成能力.

（2）Prefix向量（Prefix Vector）：Prefix-Tuning引入了一个可训练的连续向量，称为Prefix。这个Prefix向量会被添加到任务输入的开头，作为一个任务特定的前缀。Prefix向量的维度通常比较小，例如几十维或几百维.

（3）Transformer模型：Prefix-Tuning使用Transformer模型作为生成模型。Transformer模型由多个Transformer层组成，每个层包含自注意力机制和前馈神经网络。在Prefix-Tuning中，Transformer模型的参数被冻结，只有Prefix向量是可训练的。

通过将Prefix向量添加到任务输入的开头，Prefix-Tuning使得模型可以根据任务的要求生成相关内容。模型可以通过自注意力机制来关注Prefix向量，从而影响生成结果。Prefix-Tuning通

过优化Prefix向量，使得模型能够在保持预训练语言模型的基础上，针对特定任务进行优化和个性化。

本文用了bert gpt2为基础模型 对Table-to-text 以及Summarization这两个任务做为比较对象。

2.优势

（1）Prefix-Tuning在低数据和外推设置中（训练过程中未见过的示例）都优于微调.
（2）prefix是针对自然语言生成任务。

3.内部比较

（1）Prefix Length：开始增加前缀的长度性能会上涨，到一个阈值就会保持稳定。
（2）Full vs Embedding-only：prefix-tuning除了优化了嵌入层的连续向量之外，还优化了一个小的连续任务特定向量，也称为prefix。这个prefix是一个可训练的向量序列，它与输入序列的每个token相关，并且可以被Transformer模型参考。通过优化prefix，prefix-tuning使得模型可以更好地生成与特定任务相关的文本。相比之下，embedding-only方法仅优化嵌入层的连续向量，没有额外的任务特定向量进行优化
（3）Prefixing vs Infixing：prefix-tuning略优于infix-tuning。这是因为prefix-tuning可以影响到输入序列x和y的激活函数，而infix-tuning只能影响到y的激活函数。因此，prefix-tuning在性能上稍微优于infix-tuning。

三.P-Tuning: GPT Understands, Too（针对NLU，在GPT上做的）

1.模型的组成

1. 预训练语言模型：P-tuning模型以预训练语言模型为基础，可以使用任何类型的预训练语言模型，包括BERT、GPT、RoBERTa等。
2. 输入嵌入层：P-tuning模型在输入端对预训练语言模型的输入嵌入层进行修改，将其替换为针对特定任务的输出嵌入层。（它通过非侵入式地修改输入序列的嵌入表示来提高模型的性能。具体来说，P-tuning将输入序列中的每个词的嵌入表示替换为其对应的输出嵌入表示。这种方法可以在不改变模型参数的情况下，通过对输入序列的微调，提高模型的性能）
3. 任务输出嵌入层：针对不同的任务，P-tuning模型需要构建不同的任务输出嵌入层。任务输出嵌入层将任务特定的提示符转换为语言模型的输入，并生成任务的输出。
4. 调优层：P-tuning模型还包括一个调优层，用于对模型的输出进行微调。这可以包括使用监督学习对模型的输出进行训练，或者使用强化学习对模型的策略进行优化。
5. 预测层：最后，P-tuning模型还包括一个预测层，用于根据任务的输出生成最终的预测结果。

2.实现的原理

P-tuning和prefix都是对输入序列进行调整的方法，但它们进行调整的层次不同。P-tuning是在嵌入层进行调整，它通过替换预训练语言模型的输入嵌入，将连续的prompt嵌入到预训练语言模型中。而prefix是在输入层进行调整，它允许在输入序列的开头添加prompt令牌。

3.比较的对象

文中使用了Knowledge Probing和SuperGLUE的数据集来对模型进行性能评估，该微调模型可以使得GPT在NLU的性能跟bert相当。
其中P@1是Precision@1的缩写，MP指的是手动提示（Manual Prompt），文中把P-tuning跟手动添加提示做了对比，**自动预设提示**，不需要人为的调整。P-tuning方法是一种自动搜索连续空间中的提示的方法，以弥合GPT和NLU应用程序之间的差距。它通过利用少量的连续自由参数作为输入，优化连续提示，从而为GPT带来了实质性的改进。（利用少量的连续自由参数作为输入是为了让其适应下游任务）

四.Prompt Tuning:The Power of Scale for Parameter-Efficient Prompt Tuning（在T5上做的）

1.模型的组成

其与prefix类似，可以说成prefix的简化版
（1）Prompt tuning 是一种学习“软提示（soft prompt）”以将冻结的语言模型应用于特定下游任务的简单而有效的机制。与 GPT-3 中使用的离散文本提示不同，软提示是通过反向传播学习得到的，可以调整以合并来自任何数量的标签或其他形式的信号。在 prompt tuning 中，模型的架构通常包括一个前缀或提示，它被附加到输入序列的开头，以帮助模型更好地理解任务。这个前缀可以是文本，也可以是特定的编码或表示，用于指示模型如何解释输入数据。

（2）Prompt tuning 还包括一个参数化提示嵌入的机制，这些嵌入被添加到模型的嵌入表中。与 GPT-3 中的离散提示不同，这些嵌入可以通过反向传播进行更新，以优化模型在特定任务上的性能。在训练期间，模型会根据输入序列和目标输出序列的相似性来学习这些嵌入。

（3）prompt tuning 还涉及一个调整模型参数的过程，以更好地适应特定的下游任务。这可以通过在训练期间冻结模型的一部分参数，然后只调整剩余的参数来实现。这种方法可以减少需要训练的参数数量，从而提高训练速度和效率。

2.与prefix差别

Prompt Tuning是一种模型调整方法，通过向模型提供额外的信息（即“提示”）来改变模型的行为。这种提示可以是以文本形式呈现的任务描述和/或几个示例。这种方法在GPT-3等大型预训练模型中特别有效，可以有效地调整模型以适应各种不同的下游任务，而无需对模型进行大量的重新训练。Prompt Tuning的优势在于，它允许一个通用的模型服务于多个不同的任务，从而减少了模型管理的复杂性。
而prefix在GPT系列模型中，是一种用于生成特定输出的一种输入形式。比如在GPT-3中，用户输入的文本会被分解成一系列的token，然后这些token会被加入到前文（prefix）中，一起

作为新的输入，以生成新的输出。Prefix在GPT系列模型中的作用是帮助模型理解用户的意图，从而生成符合用户要求的输出。
总的来说，Prompt Tuning和prefix都是为了改善GPT系列模型的表现，但它们的应用场景和目的是不同的。Prompt Tuning主要用于调整模型以适应不同的下游任务，而prefix则主要用于帮助模型理解用户的输入意图。

3.这些PEFT模型的共同点

都是通过对外部参数进行调整，而不是对原始的大型预训练模型的参数进行调整，以达到减少功耗和提高效率的效果。这些方法的主要思想是，通过添加额外的信息或结构（例如提示、前缀或锚定标记），使模型能够更好地理解和适应特定的下游任务。这些额外的信息或结构可以被视为模型的一部分，但它们并不直接改变原始模型的参数。通过只对外部参数进行调整，这些方法可以避免对原始模型进行大量的重新训练，从而减少了训练时间和计算资源的消耗。此外，由于外部参数是针对特定任务的，因此可以根据任务需求进行灵活调整和优化，进一步提高了模型的效率和性能

五.Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning（在 DeBERTaV3-base，dBART-large）

1.与Lora的区别

（1）其发现了lora的缺陷，因为lora对矩阵的r是提前就决定了而没有考虑神经网络每个层各自的重要性，所以adalora引入了动态的调整矩阵的权重，根据矩阵的重要性来给处相应的权重。
（2）其adalora引入了动态的调整矩阵的权重是通过奇异值分解（SVD），然后对奇异值进行剪枝，减去值小的矩阵奇异值。

2.实验部分

跟Full fine-tuning，Bitfit，Adapter tuning ， Lora 的微调方法进行比较(Bitfit的原理是先分配一小部分资源看看效果好不好，如果好的话就多分配一些资源，Adapter tuning是在每个transform模型里面加入两个适配器)

3.结果

（1）那些下游任务是adaptive效果更好。
（2）发现AdaLoRA总是倾向于将更多的预算分配给ffn（前馈神经网络）和顶层。这种行为与我们在图1中给出的经验结论一致，即FFN模型和顶层的权重矩阵对模型性能更重要，也应证了每个权重矩阵的重要性是不一样的，所以说adalora这种方式是更加的有效的。

六.Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning（用

在T0预训练模型上，使用IA3微调）

Few-Shot Parameter-Efficient Fine-Tuning是指的这几篇论文里面的微调方法，少参数的方法，而不是专指这篇代码。

1.基础概念

上下文无关的嵌入模型（ICL）是一种学习方法，旨在通过将输入和目标例子连接起来，以引导模型执行特定任务。这种方法特别适用于少样本学习，因为它只需要少量的带标签数据就可以训练模型。
ICL的不足：

1. 每次模型做出预测时，处理所有的输入-目标对需要付出巨大的计算成本。
2. ICL通常产生的性能相比于 fine-tuning 要差。
3. prompt 的确切格式（包括措辞和例子的顺序）会对模型的性能产生重大且不可预测的影响，这远远超出了 fine-tuning 的运行间变化。
4. 可能存在学习效果不明显的问题：最近的研究表明，即使提供了错误的标签，ICL仍然可以表现良好，这引发了关于ICL是否真正“学习”了标签示例的质疑。

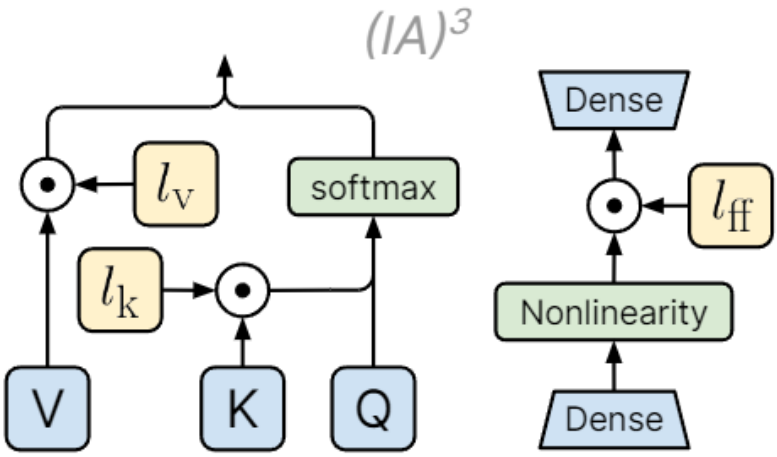
2.T-Few的结构

1. T0模型：T-Few使用T0模型作为基础模型。T0模型是一个具有110亿参数的语言模型，用于提供基本的语言理解和生成能力。
2. (IA)3参数：T-Few引入了(IA)3参数用于下游任务的适应。这些参数是在与T0相同的多任务混合训练中进行预训练的，并在每个独立的下游数据集上进行微调。
3. 损失函数：T-Few使用多个损失函数来训练模型。这些损失函数包括标准的语言建模损失LLM、错误选择的非似然损失LUL和长度归一化损失LLN。这些损失函数的组合构成了T-Few的训练目标。
4. 提示模板：在训练和推断过程中，T-Few使用提示模板将下游数据集转换为文本到文本的格式。这些提示模板用于将每个示例转化为具有指导性的文本形式。

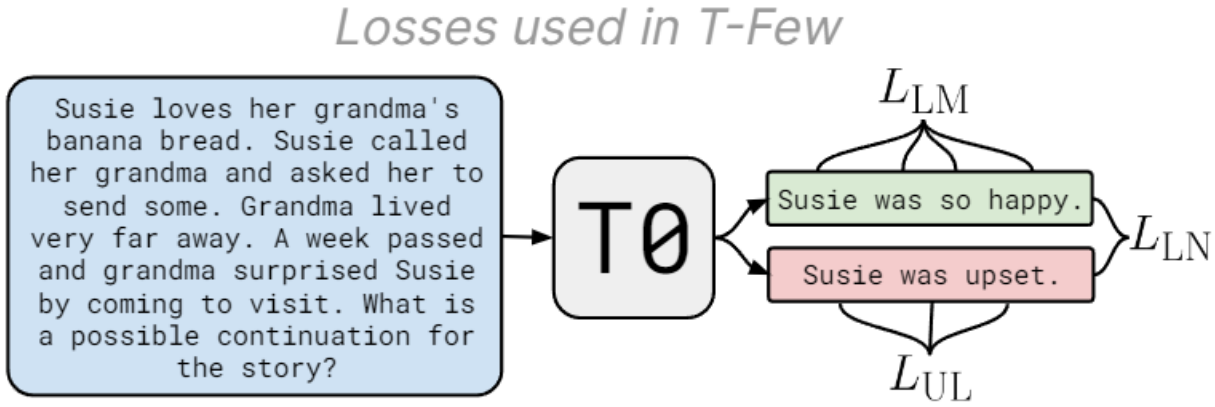
3.比较的逻辑

因为要比较ICL与T-Few的效率，且ICL支持多任务批处理，所以我们选择的微调方法就不能是单任务处理的，所以想出来IA3的方法。
文中用了RAFT数据集作为下游任务来对模型进行评估的。

4.IA3，T0结构图



其中LV LK Lff是其激活向量，通过对模型的激活进行元素级乘法（即重新缩放）来实现。具体而言，它引入了一组学习到的任务特定向量，将其应用于自注意力机制、编码器-解码器注意力机制和位置编码网络的中间激活。这些向量通过元素级乘法与激活进行逐元素的乘法操作。



在T0模型中，使用了以下几个损失函数： 1. LLM (Language Modeling Loss)：这是标准的语言建模损失函数，用于训练模型生成下一个词的概率分布。它帮助模型学习语言的概率分布和上下文信息。 2. LUL (Unlikelihood Loss)：这是一种非似然损失函数，用于惩罚模型生成不正确选择的概率。它帮助模型学习避免生成不合理或错误的输出。 3. LLN (Length-Normalized Loss)：这是一种长度归一化损失函数，用于平衡不同长度的序列对模型训练的影响。它帮助模型在生成不同长度的输出时保持一致性。

七.Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning（用的MPT，用在T5上)

1.MPT的特点以及实验流程

MPT的特点是通过使用多个源任务来训练共享矩阵，从而实现参数高效的迁移学习。（MPT中的源任务可以从现有任务中获取，也可以人工生成。这些任务可以是不同的机器学习任务，如文本分类、图像识别等。在获取源任务后，MPT使用这些任务的训练数据来学习一个共享的prompt矩阵，这个矩阵可以用于后续的目标任务适应。因此，MPT可以有效地利用现有任务的知识，以便更快速地适应新任务）
实验流程：1. 预训练模型选择：选择一个适合的预训练模型作为基础模型。常用的预训练模型包括BERT、GPT等。 2. 模型初始化：使用预训练模型的参数初始化MPT模型。这样可以利用预训练模型学到的语言表示来加速模型的收敛和提高性能。 3. 数据准备：准备源任务和目标任务的数据集。源任务是用于训练共享矩阵的任务，而目标任务是要在其上进行迁移学习的任务。 4. 共享矩阵训练：使用源任务的数据集，对共享矩阵进行训练。共享矩阵是用来捕捉跨任务的共同特征和知识的关键组件。 5. 参数更新：在训练过程中，只更新少量的任务特定参数，而共享矩阵的参数是固定的。这样可以提高参数的利用效率。 6. 目标任务适应：在训练完成后，使用目标任务的数据集对模型进行适应。这包括使用共享矩阵和源任务的提示来初始化目标任务的特定向量，并对目标任务进行微调。 7. 实验评估：使用评估指标（如F1和准确率）对模型在目标任务上的性能进行评估。通常会进行多次实验，并报告平均性能和标准差。
总结：MPT中使用目标任务的数据来训练共享矩阵的参数。具体来说，MPT首先使用源任务的数据来训练共享矩阵，然后使用目标任务的数据对共享矩阵进行微调。这样可以通过源任务的知识来初始化共享矩阵，并通过目标任务的数据进行进一步的优化和适应。这种方式可以提高参数的利用效率，并在目标任务上获得更好的性能。

MPT针对的是多任务预训练提高模型的泛化能力，而Prefix-Tuning主要针对特定任务进行优化。
MPT里面还涉及到迁移学习以及蒸馏学习。

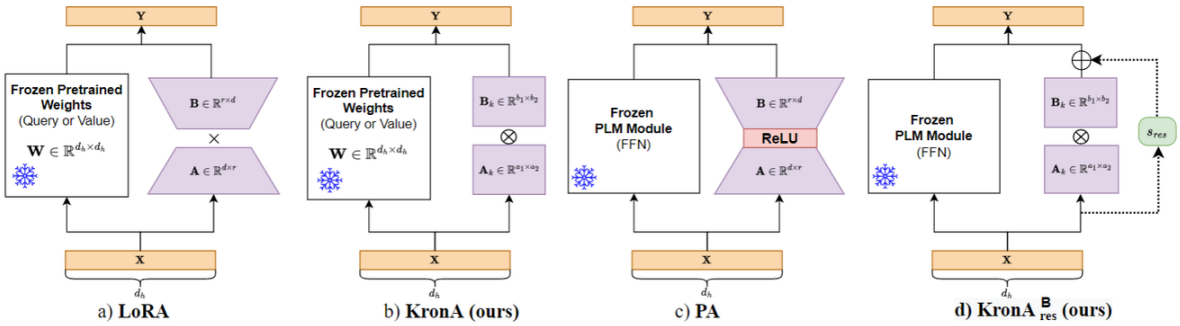
八.FedPara Low-Rank Hadamard Product for Communication-Efficient Federated Learning

文章中提到了FedPara，这是一种低秩Hadamard积参数化方法，用于提高通信效率的联邦学习。联邦学习是一种机器学习技术，可以在分布式数据集上进行训练，同时保护用户隐私。在联邦学习中，多个参与者共享模型参数，而不是原始数据。FedPara通过减少通信成本和优化模型参数，提高了联邦学习的效率。
此外，文章还提到了Jacobian Correction，这是一种正则化方法，通过诱导X1、X2、Y1、Y2的雅可比矩阵与构造矩阵W的雅可比矩阵相符。这种正则化方法在自然语言处理中可以用于优化模型参数，提高模型的泛化能力。
这个方法的主要特点是通过将模型分为全局层和本地层来解决个性化问题。客户端只传输全局层（顶层），而将本地层（底层）保留在每个设备上。全局层共同学习提取通用特征，而本地层则偏向于每个用户。使用Hadamard乘积作为全局内部权重W1和本地内部权重W2之间的桥梁。个性化模型的每一层由W=W1⊙(W2+1)构建，其中W1在训练期间传输到服务器，而W2在本地设备上保留。这使得W1能够隐式地学习全局共享知识，并充当术语(W2+1)的开关。相比于传统的联邦学习方法，这种方法在相同的秩条件下提高了通信效率

九.KronA Parameter Efficient Tuning with Kronecker Adapter

1.KronA简介

KronA是一种基于Kronecker乘积的适配器模块，用于对预训练语言模型进行微调。Kronecker乘积是一种矩阵运算，它将两个输入矩阵的每个元素相乘，并生成一个块矩阵作为输出。KronA模块通过将低秩投影替换为Kronecker乘积来改进现有的适配器方法，如LoRA、PA和Adapter。 KronA模块的结构类似于LoRA模块，但是使用Kronecker乘积替代了普通的矩阵乘法。KronA模块包含两个Kronecker因子Ak和Bk，它们分别替代了LoRA模块中的下投影和上投影。KronA模块还具有一个固定的缩放因子s，用于调整输出的幅度。 在微调阶段，KronA模块并行应用于预训练语言模型的权重矩阵。一旦微调完成，Kronecker因子会被相乘，然后通过缩放和合并操作与原始的预训练语言模型权重矩阵相结合。因此，与LoRA类似，KronA不会增加推理时间。 KronA模块的初始化可以选择将其中一个Kronecker因子初始化为零，这种初始化方式相对于使用正态分布初始化两个因子可以显著提高模型的性能。 总之，KronA模块通过使用Kronecker乘积替代低秩投影，提供了一种改进的适配器方法，可以在微调预训练语言模型时提高性能。



PA代表的是一种名为PA（Parameter Adaptation）的方法。PA是一种参数适应方法，用于对Transformer-based PLMs进行微调。它通过在原始模型中插入可训练的参数适应器来实现。这些适应器可以根据特定任务的需求来调整模型的参数，从而提高性能。PA方法在GLUE基准测试中取得了不错的结果，并且相对于其他一些参数高效调整（PET）方法具有较低的计算成本。

KronAB模型是由KronA模块和FFN（前馈网络）模块并行组合而成的。其中，KronA模块利用Kronecker product decomposition替代了LoRA中的低秩分解，以提高模型的准确性，而不增加推理延迟。KronAB模块的设计目的是为了在保持较高准确性的同时，允许增加一定的推理时间和计算量。在训练完成后，KronAB模块可以作为预训练模型的一部分进行合并（KronAB模型是受到了PA方法的影响，让其与FFN并行，且加入了res残差连接）

2.结果

比之前的peft的结果都好。