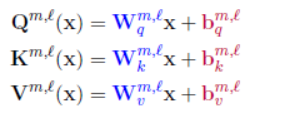
# 微调方法总结

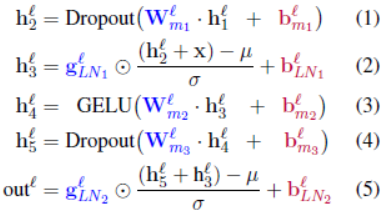
huangkai 2023/6/28

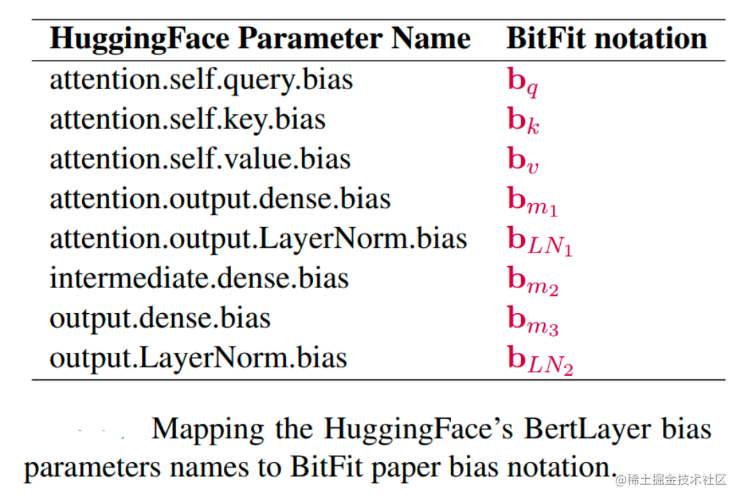
## BitFit

是一种稀疏的微调方法，它训练时只更新bias的参数或者部分bias参数。

Bias参数列表：







## Prefix Tuning

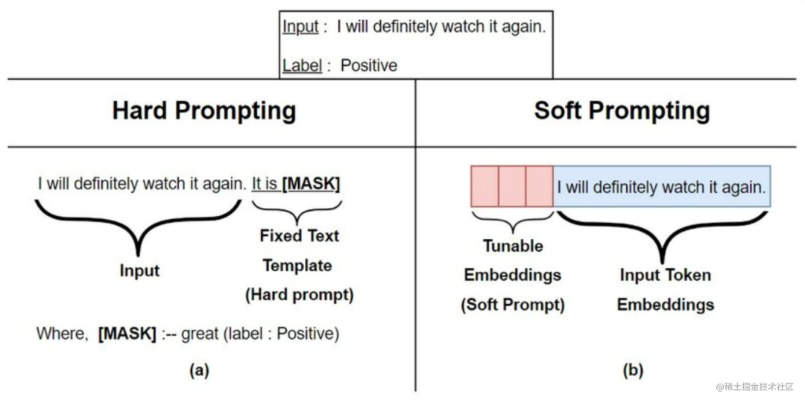
在输入token之前构造一段任务相关的virtual tokens作为Prefix，然后训练的时候只更新Prefix部分的参数，而PLM中的其他部分参数固定。

Prefix构造方法：

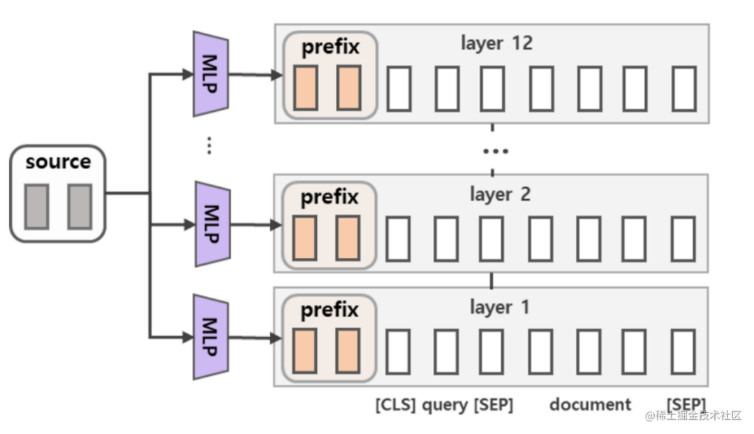
自回归：z = [PREFIX; x; y]，合适的上文能够在固定LM情况下去引导生成下文；

ENCODER-DECODER：z = [PREFIX; x; PREFIX0; y]，encoder端前缀引导输入部分编码；decoder端前缀是引导后续token生成。

和prompt构造类似，prompt是认为构造的显示提示，并且无法更新参数；prefix可以学习隐式提示。



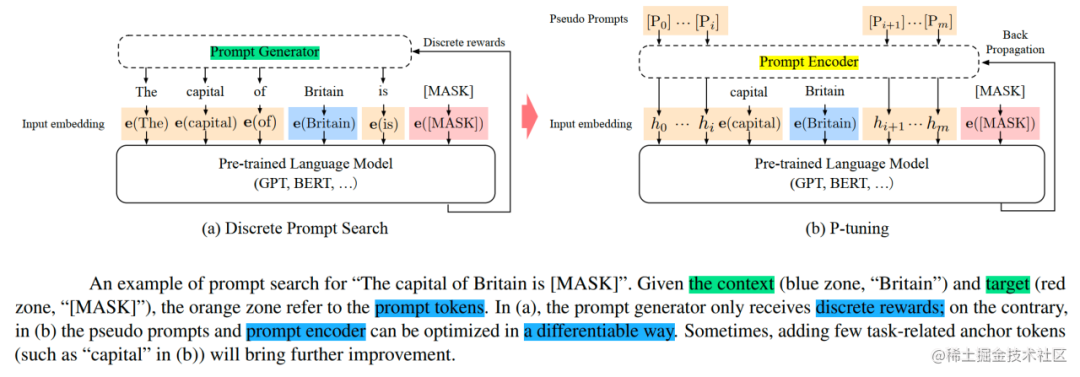
在Prefix层加了MLP结构，然后切分为key和value矩阵，然后和每一层的key-value矩阵拼接。训练完成后，只保留Prefix的参数。



Prefix-tuning的效果优于Infix-tuning，Prefix-tuning形式为 [PREFIX; x; y]，Infix-tuning形式为 [x; INFIX; y]。

## P-Tuning

该方法将Prompt转换为可以学习的Embedding层，并用MLP+LSTM的方式来对Prompt Embedding进行一层处理。



相比Prefix Tuning，P-Tuning加入的可微的virtual token，但仅限于输入层，没有在每一层都加；另外，virtual token的位置也不一定是前缀，插入的位置是可选的。这里的出发点实际是把传统人工设计模版中的真实token替换成可微的virtual token。

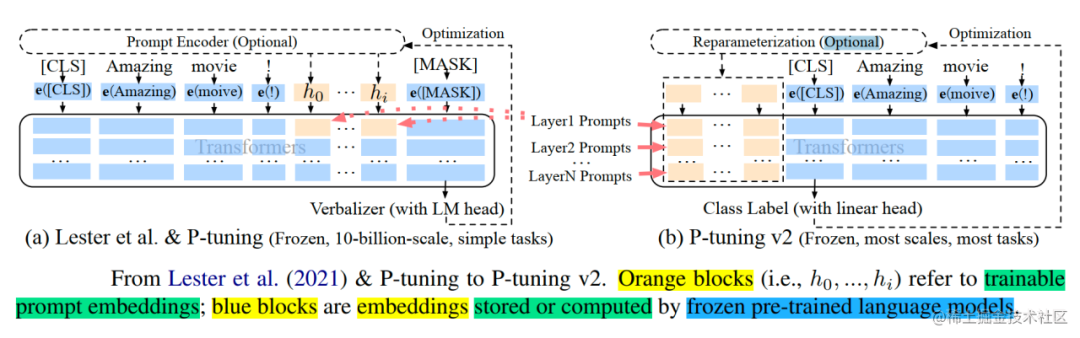
经过预训练的LM的词嵌入已经变得高度离散，如果随机初始化virtual token，容易优化到局部最优值，而这些virtual token理论是应该有相关关联的。因此，作者通过实验发现用一个prompt encoder来编码会收敛更快，效果更好。即用一个LSTM+MLP去编码这些virtual token以后，再输入到模型。

## P-Tuning v2

该方法在每一层都加入了Prompts tokens作为输入，而不是仅仅加在输入层，这带来两个方面的好处：

1更多可学习的参数（从P-tuning和Prompt Tuning的0.01%增加到0.1%-3%），同时也足够参数高效。

2加入到更深层结构中的Prompt能给模型预测带来更直接的影响。

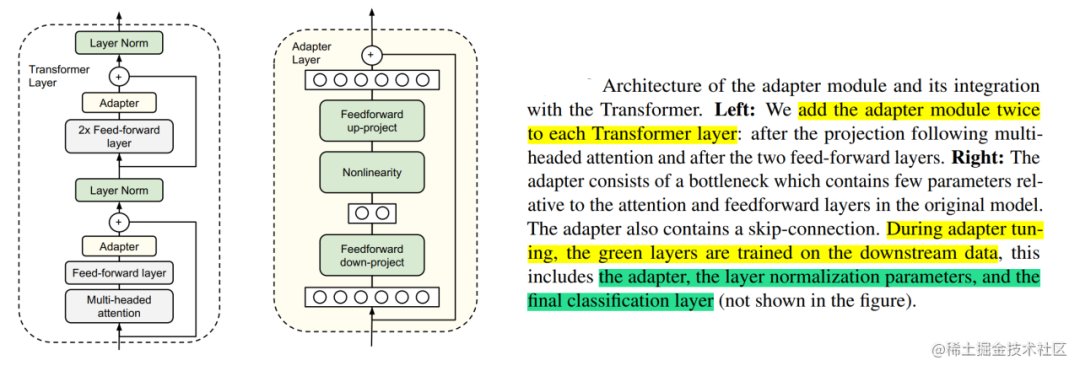


改进点：

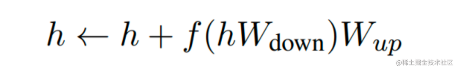
1. 移除重参数化的编码器，如Prefix Tuning中的MLP和P-Tuning中的LSTM；
2. 不同任务采用不同的提示长度；
3. 引入多任务学习，尤其是在序列标注任务中，多任务学习是有益的；
4. 回归传统的分类标签范式，而不是映射器

## Adapter Tuning

该方法设计了Adapter结构，并将其嵌入Transformer的结构里面，针对每一个Transformer层，增加了两个Adapter结构(分别是多头注意力的投影之后和第二个feed-forward层之后)，在训练时，固定住原来预训练模型的参数不变，只对新增的 Adapter 结构和 Layer Norm 层进行微调，从而保证了训练的高效性。



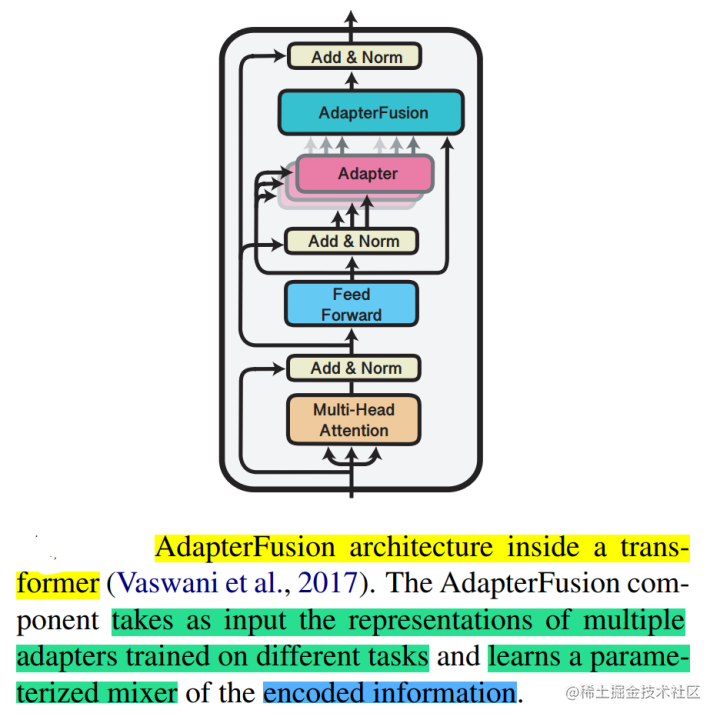
映射公式如下：



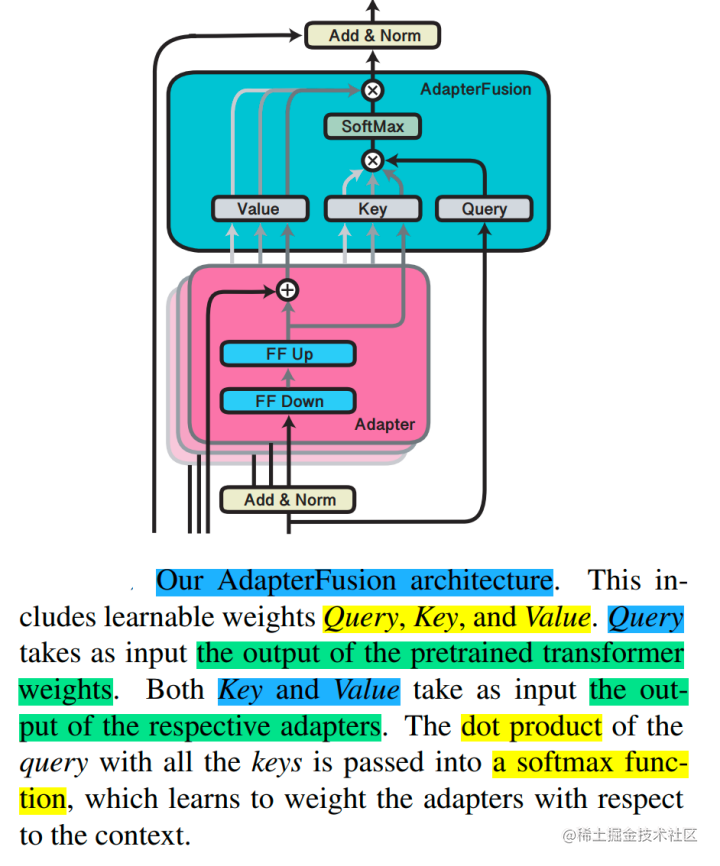
## AdapterFusion

融合多任务信息的Adapter的变体，在Adapter的基础上进行优化，通过将学习过程分为两阶段来提升下游任务表现。

* 知识提取阶段：在不同任务下引入各自的Adapter模块，用于学习特定任务的信息。
* 知识组合阶段：将预训练模型参数与特定于任务的Adapter参数固定，引入新参数（AdapterFusion）来学习组合多个Adapter中的知识，以提高模型在目标任务中的表现。



Adapter结构：

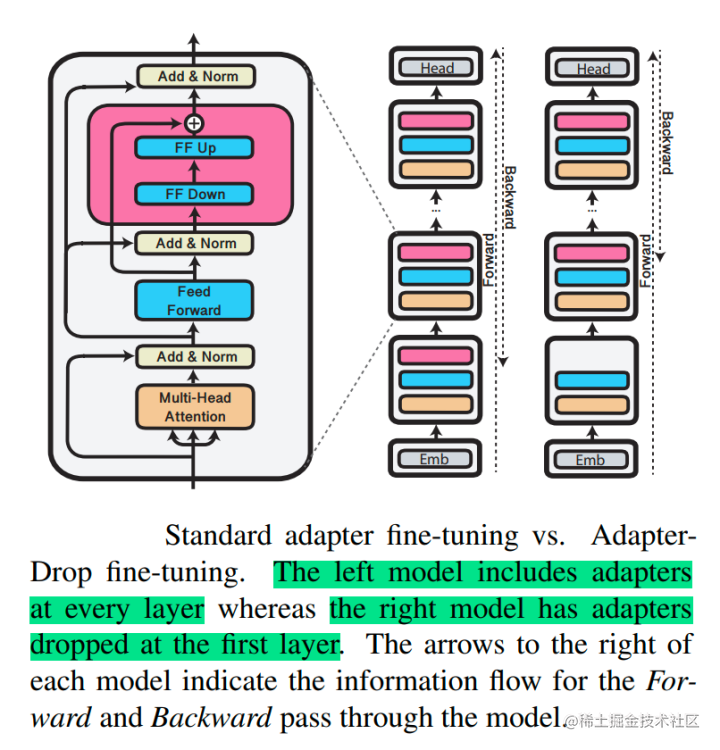


## AdapterDrop

与全量微调相比，Adapter在训练时快60%，但是在推理时慢4%-6%。AdapterDrop方法缓解该问题。

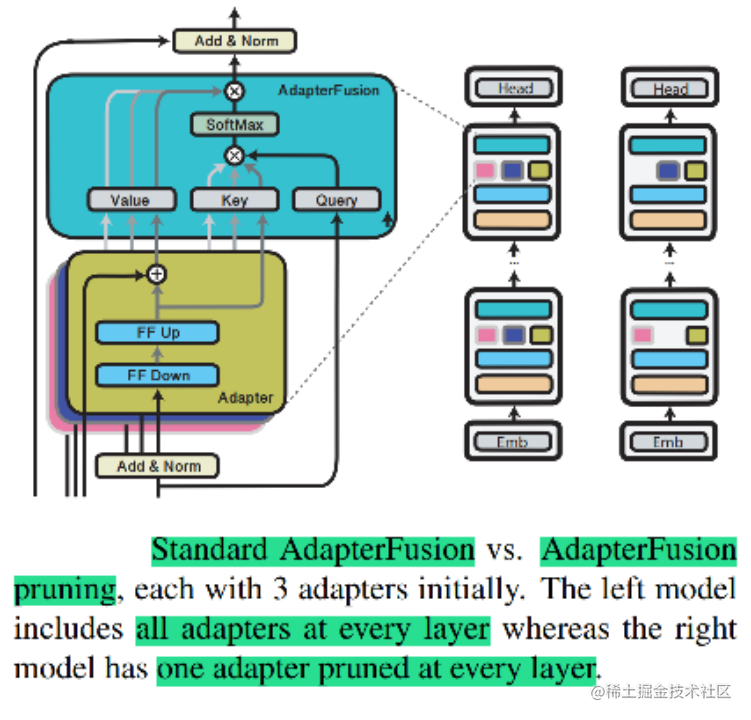
AdapterDrop在不影响任务性能的情况下，对Adapter动态高效的移除，尽可能的减少模型的参数量，提高模型在反向传播（训练）和正向传播（推理）时的效率。

Adapter Tuning中的剪枝图解：



实验表明，从较低的Transformer层中删除Adapter可以显着提高多任务设置中的推理速度。例如，将前五个Transformer层中的Adapter丢弃，在对8个任务进行推理时，速度提高了39%。并且即使有多个丢弃层，AdapterDrop也能保持良好的结果。

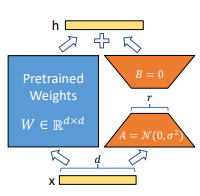
Adapter Fusion的剪枝图解：

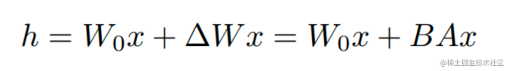


通过实验表明可以移除AdapterFusion中的大多数Adapter而不影响任务性能。使用剩余的两个Adapter，实现了与具有八个Adapter的完整AdapterFusion模型相当的结果，并将推理速度提高了68%。

## LoRA

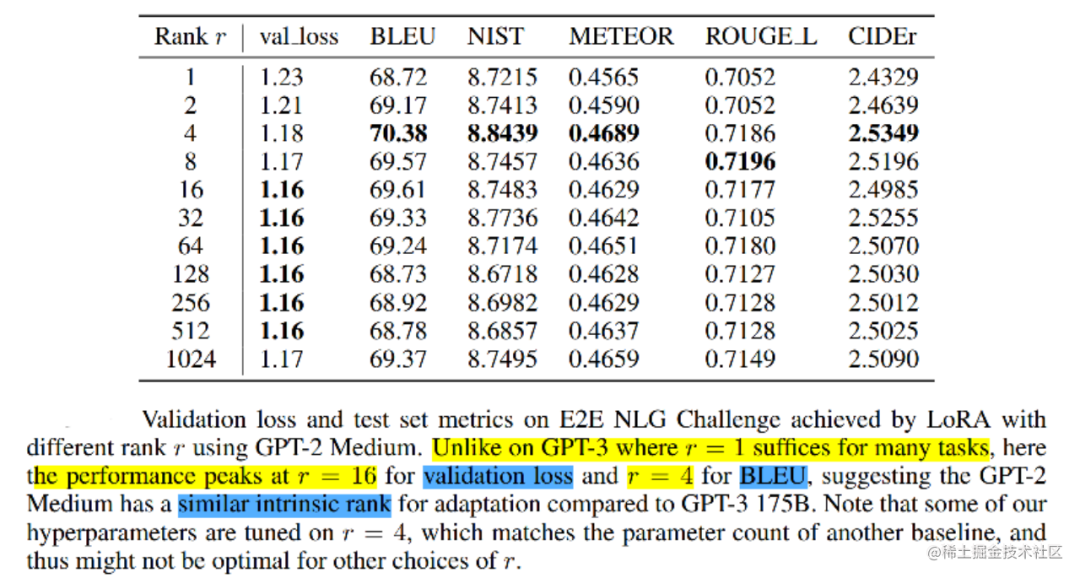
原理：冻结预训练好的模型权重参数，在冻结原模型参数的情况下，通过往模型中加入额外的网络层，并只训练这些新增的网络层参数。由于这些新增参数数量较少，这样不仅 finetune 的成本显著下降，还能获得和全模型微调类似的效果。





在推理时，将左右两部分的结果加到一起即可，h=Wx+BAx=(W+BA)x，所以只要将训练完成的矩阵乘积BA跟原本的权重矩阵W加到一起作为新权重参数替换原本PLM的W即可，对于推理来说，不会增加额外的计算资源。

此外，Transformer的权重矩阵包括Attention模块里用于计算query,key,value的Wq，Wk，Wv以及多头attention的Wo,以及MLP层的权重矩阵，LoRA只应用于Attention模块中的4种权重矩阵，而且通过消融实验发现同时调整Wq和Wv会产生最佳结果。



秩rank的选择取4，8，16最佳

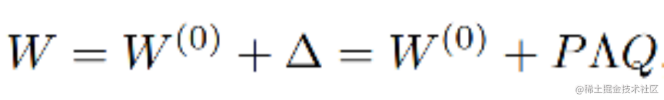
## AdaLoRA

LoRA的问题：LoRA则需要预先指定每个增量矩阵的本征秩 r 相同，忽略了在微调预训练模型时，权重矩阵的重要性在不同模块和层之间存在显著差异，并且只训练了Attention，没有训练FFN，事实上FFN更重要。

AdaLoRA是对LoRA的一种改进，它根据重要性评分动态分配参数预算给权重矩阵。具体做法如下：

调整增量矩分配。AdaLoRA将关键的增量矩阵分配高秩以捕捉更精细和任务特定的信息，而将较不重要的矩阵的秩降低，以防止过拟合并节省计算预算。

以奇异值分解的形式对增量更新进行参数化，并根据重要性指标裁剪掉不重要的奇异值，同时保留奇异向量。由于对一个大矩阵进行精确SVD分解的计算消耗非常大，这种方法通过减少它们的参数预算来加速计算，同时，保留未来恢复的可能性并稳定训练。



在训练损失中添加了额外的惩罚项，以规范奇异矩阵P和Q的正交性，从而避免SVD的大量计算并稳定训练。

## QLoRA

目的：减少资源消耗

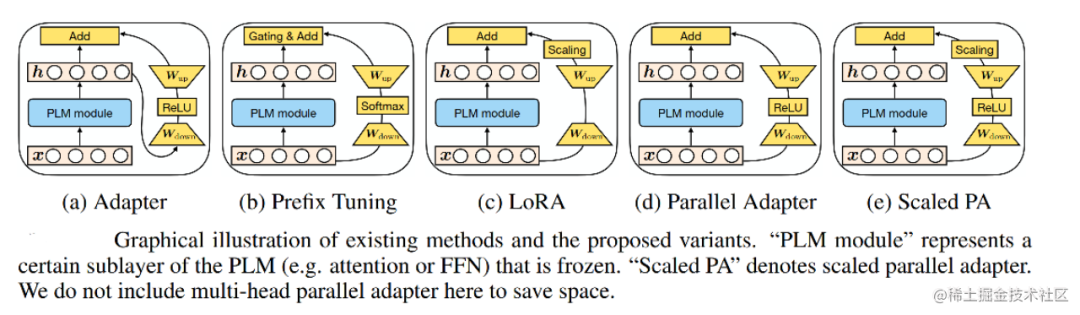
优化点如下：

4bit NormalFloat（NF4）：对于正态分布权重而言，一种信息理论上最优的新数据类型，该数据类型对正态分布数据产生比 4 bit整数和 4bit 浮点数更好的实证结果。

双量化：对第一次量化后的那些常量再进行一次量化，减少存储空间。

分页优化器：使用NVIDIA统一内存特性，该特性可以在在GPU偶尔OOM的情况下，进行CPU和GPU之间自动分页到分页的传输，以实现无错误的 GPU 处理。该功能的工作方式类似于 CPU 内存和磁盘之间的常规内存分页。使用此功能为优化器状态（Optimizer）分配分页内存，然后在 GPU 内存不足时将其自动卸载到 CPU 内存，并在优化器更新步骤需要时将其加载回 GPU 内存。

## MAM Adapter



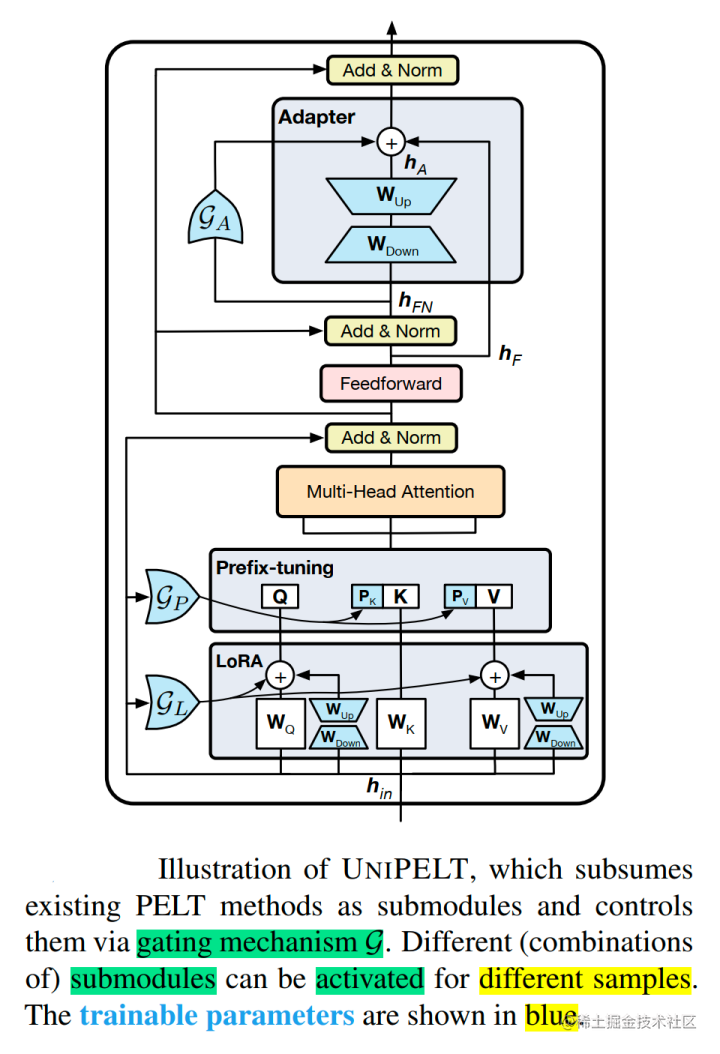
作者对Adapter的放置和软提示（soft prompt）进行了详细的调查。得出如下结论：

1.并行放置的Adapter优于顺序放置的Adapter，并且与 FFN 并行放置的Adapter优于多头注意力（MHA）并行放置的Adapter（模型修改的位置如下图中所示，蓝色表示修改Attention、红色表示修改FFN）。

2.软提示可以通过仅更改 0.1% 的参数来有效地修改注意力。

## UniPELT

使用门控结构，结合Prefix-tuning、LoRA、Adapter三种方法，调控最适合当前数据或者任务最适宜的微调方法。结构如下：



其中：GP门结构调控Prefix-tuning、GL调控LoRA结构、GA调控Adapter结构。