Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey

背景

微调对于增强LLM在看不见的用户数据集和任务上的性能至关重要。随着模型规模的增长（例如GPT-2中的**1.5**B到GPT-3中的**175**B），标准的全微调范式需要数千个GPU并行工作，这是非常低效和不可持续的。新算法，即参数高效微调（PEFT），目的是在下游任务上调整最少的参数以获得比全微调更好的性能。

具体来说，**PEFT 是指调整预训练大型模型的参数以使其适应特定任务或领域，同时最大限度地减少引入的附加参数或所需的计算资源的数量的过程。**

广泛采用的微调LLM的策略为参数高效微调，选择性地调整一小部分参数，同时保持其余参数不变。

此外，PEFT的应用超出了NLP领域，并迅速吸引了CV社区的兴趣，用于处理具有大参数的微调视觉模型，如视觉transformer（ViT）和扩散模型，以及视觉语言模型。

PEFT分类

根据PEFT算法的操作将其分为**增量微调**、**选择性微调**、**重参数化微调**和**混合微调**。

1：如图所示，三种主要的增量微调算法：(1)**适配器**（2）**软提示**（3）**其他**.它们之间的区别在于不同的附加可调模块或参数。

2：选择性微调不需要任何额外的参数，它从主干模型中选择一小部分参数，并仅使它们可调，在微调期间保持大多数参数不变。根据所选参数的分组情况对选择性微调进行了分类：(1) **非结构性屏蔽**；(2) **结构性屏蔽。**

3：重参数化表示在两种等效形式之间转换模型参数。具体来说，重参数化微调在训练过程中引入额外的低秩可训练参数，然后与原始模型整合进行推理。主要分为两种策略：**低秩分解**和 **LoRA 衍生。**

4：混合微调探索了不同 PEFT 方法的设计空间，并结合了它们的优势。

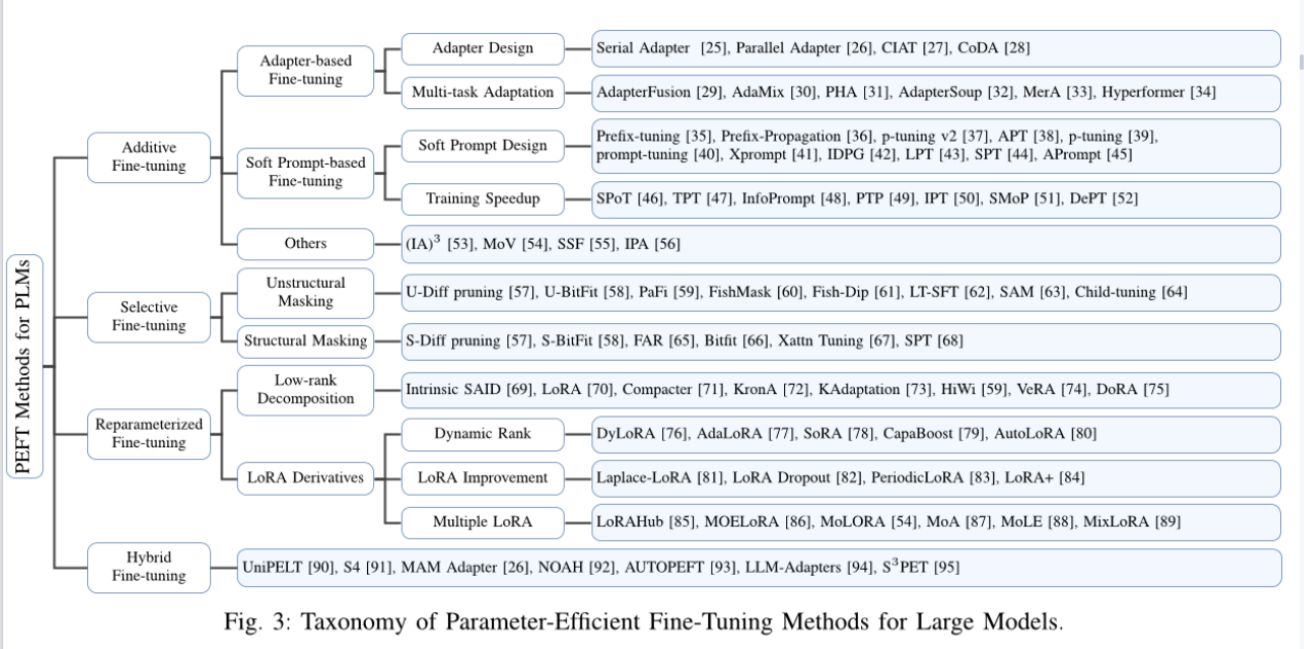


Fig. 3: Taxonomy of Parameter-Efficient Fine-Tuning Methods for Large Models.

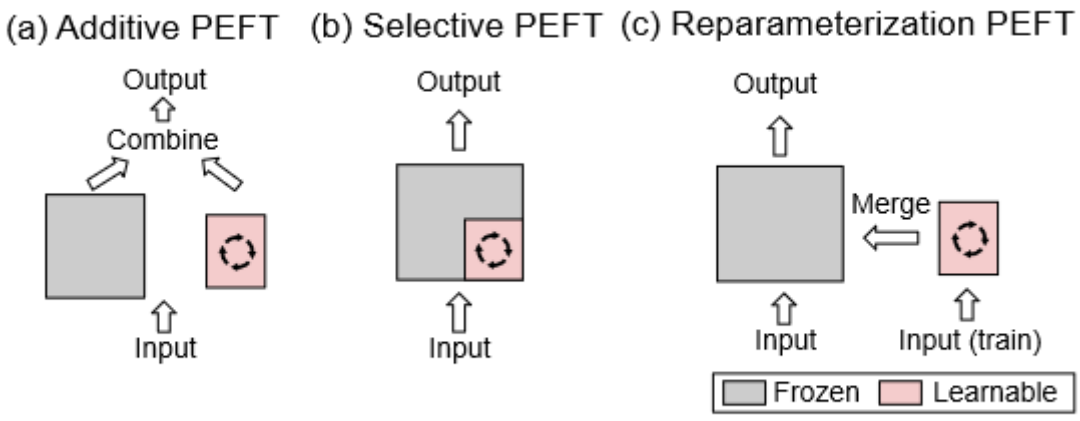


Fig. 4: Different types of PEFT algorithms.

增量微调

标准的全面微调需要大量的计算费用，还可能损害模型的泛化能力。为了缓解这一问题，一种被广泛采用的方法是保持预训练的骨干模块不变，只引入极少量的可训练参数，这些参数在模型架构中处于战略位置。

在针对特定下游任务进行微调时，只更新这些附加模块或参数的权重，从而大幅减少对存储、内存和计算资源的需求。由于这些技术都具有增加参数的特点，因此可将其称为 "增量微调"。接下来，我们将讨论主流的增量 PEFT 算法。

适配器：适配器是在Transformer模块中插入小型适配器层。适配器层由下投影矩阵W_{down}组成，然后是激活函数σ()，最后是上投影矩阵W_{up}。

适配器模块内的计算（具有残差）可以总结如下：

$$Adapter(x) = W_{\text{up}}\sigma(W_{\text{down}}x) + x. \qquad (7)$$

NLP领域适配器的概念最初是由**Serial Adapter**引入的，如图所示。在他们的方法中，通过添加两个适配器模块来增强每个Transformer model块，一个位于自注意力层之后，另一个位于FFN层之后。随后的研究旨在解决与适配器层相关联的额外计算成本。

之后引入了 **parallel adapter** (PA) 方法，如图所示，该方法将传统的Serial Adapter重组为并行侧网络，与每个Transformer子层并行运行。同样，CIAT、CoDA 和 KronA 也采用了并行适配器设计。

除了并行设计，**CoDA**（Conditional adapters） 还采用了稀疏激活机制来提高推理效率。少数token由冻结的Transformer层选择和处理，而所有的token由可调的适配器层处理。 CoDA发表于23年4月，NeurIPS 2023。

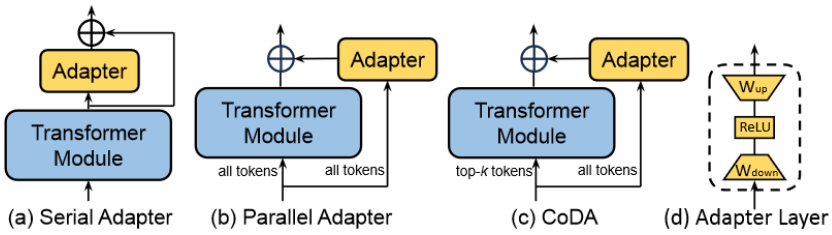


Fig. 5: Illustration of three representative adapter-based fine-tuning algorithms. Blue represents frozen, while yellow represents trainable.

为了提高适配器的性能和通用性，许多研究都采用了多任务学习策略，如 AdapterFusion 、AdaMix 、PHA 、AdapterSoup 、MerA 和 Hyperformer 。

AdapterFusion 在模型中保留了所有预先训练好的适配器，并采用融合模块来合并多任务信息。与 AdapterFusion 不同的是，**MerA** 通过基于权重和激活值的最佳传输，将预先训练好的适配器合并成一个。这种方法避免了引入任何额外的可训练参数，从而提高了计算效率。**Hyperformer** 将多任务信息存储在共享超网络中，并根据任务和层 ID 嵌入生成特定于任务和层的适配器参数。对于新任务，只需学习额外的任务嵌入，从而减少了训练参数的数量。

软提示（Soft Prompt）：人们普遍认为，软提示的连续嵌入空间本质上包含更多信息，而不是通过上下文学习来优化离散标记表征 。研究人员从这一概念中汲取灵感，直接将可调整向量（称为软提示）附加到输入序列的起始位置。具体表现如下

$$\mathbf{X}^{(l)} = [\mathbf{s}_1^{(l)}, \ldots, \mathbf{s}_{N_S}^{(l)}, \mathbf{x}_1^{(l)}, \ldots, \mathbf{x}_{N_X}^{(l)}] \qquad (8)$$

$\mathbf{X}^{(l)}$是层L的输入token序列，前面是软提示token$\mathbf{S}_i^{(l)}$，后跟原始输入token$\mathbf{x}_i^{(l)}$。

Prefix-tuning 引入了可学习向量，这些向量被添加到所有Transformer层的键 k 和值 v 的前缀中。为了确保优化过程中的稳定性，**Prefix-tuning**采用了重参数化策略，利用 MLP 层生成这些前缀向量，而不是直接对其进行优化。微调后，仅保存前缀向量用于推理。

一些研究对这一技术进行了调整和改进 。例如，**p-tuning-v2** 取消了重参数化，并将其应用扩展到更广泛的模型规模和 NLP 任务。**APT**（自适应前缀调整）通过引入自适应门机制来控制每一层的前缀重要性，从而增强了前缀调整功能。

p-tuning 和 **prompt-tuning** 只在初始词嵌入层应用可学习向量，以提高训练和推理效率。需要强调的是，**prompt-tuning** 主要是在大型模型，特别是参数超过 110 亿的模型中显示出其有效性。

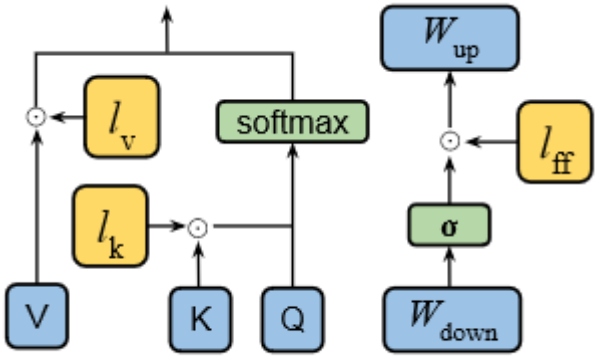
软提示已被用于各种下游任务，但是它们的训练容易不稳定和收敛缓慢。

- 为了解决这个问题，**SPoT**使用从一个或多个任务中学习到的源提示来初始化新任务的提示。
- 同样，**TPT**（transferable prompt tuning）中也提出了从一个任务中迁移软提示来初始化另一个任务的方法，该方法证明了：更好的提示初始化会带来很大的训练收敛速度提升。
- **SMoP**（Sparse Mixture-of-Prompts） 通过使用简短的软提示来降低训练和推理成本。在训练过程中，会对多个简短的软提示进行训练，每个提示都是针对数据集的特定子集量身定制的。在推理过程中，**SMoP** 集成了一种门控机制，可将每个输入实例发送到适当的短提示。这种技术不仅提高了训练和推理阶段的效率，而且还能保持与使用较长软提示时相当的性能。（发表于23年，EMNLP）

其他增量微调方法：除了上述方法之外，还出现了在微调过程中战略性地结合附加参数的其他方法。

例如，**(IA)^3**引入了三个可学习的重新缩放向量，分别重新缩放键、值和FFN激活值，

Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning 发表于2022 NIPS



(a) (IA)^3

自注意力块内的操作如下所示：

$$SA(x) = Softmax(\frac{Q(l_k \odot K^T)}{\sqrt{d_{head}}})((l_v \odot V). \qquad (9)$$

在FFN中，重新缩放可以表示为：

$$FFN_{Transfomer}(x) = W_{up}(l_{\text{ff}} \odot \sigma(W_{down}x)), \qquad (10)$$

选择性微调

与增加更多参数以增加模型复杂度的增量式PEFT不同，选择性PEFT仅微调现有参数的子集以提升模型在下游任务中的性能。 最早的选择性PEFT是仅微调网络的几个顶层（冻结前层），现代方法通常基于层的类型（Cross-Attention is All You Need）或内部结构，仅微调模型的偏置（BitFit）或仅特定的行（Efficient Fine-Tuning of BERT Models on the Edge）。

选择性PEFT一般通过采用二进制掩码M对应所有参数，每个M为0或1，表示是否选择相应的参数进行微调。1表示该参数选中被微调，0则表示未被选择。

非结构化掩码：

- **Diff pruning**是一项代表性工作，它在微调期间将可学习的二进制掩码应用于模型权重。为了实现参数效率，该掩码通过对L0范数惩罚的可微近似进行正则化。
- **SAM**提出一种二阶近似方法，该方法用一个可解析求解的优化函数来近似原始问题，以帮助决定参数掩码。（*On the Effectiveness of Parameter-Efficient Fine-Tuning* 发表于2023 AAAI）

当实现PEFT时，非结构化参数掩蔽导致非零掩蔽的不均匀分布和降低的硬件效率。如图7所示，结构化掩码以规则的方式组织参数掩码，不像非结构化掩码那样随机应用，因此可以提高训练时的计算效率和硬件效率。

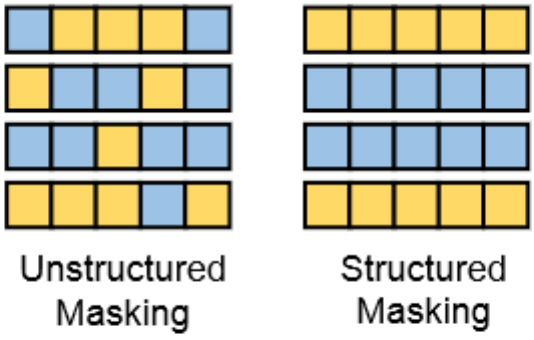


Fig. 7: Illustration of how two parameter masking methods. Blue represents frozen, while yellow represents trainable.

结构化掩码：

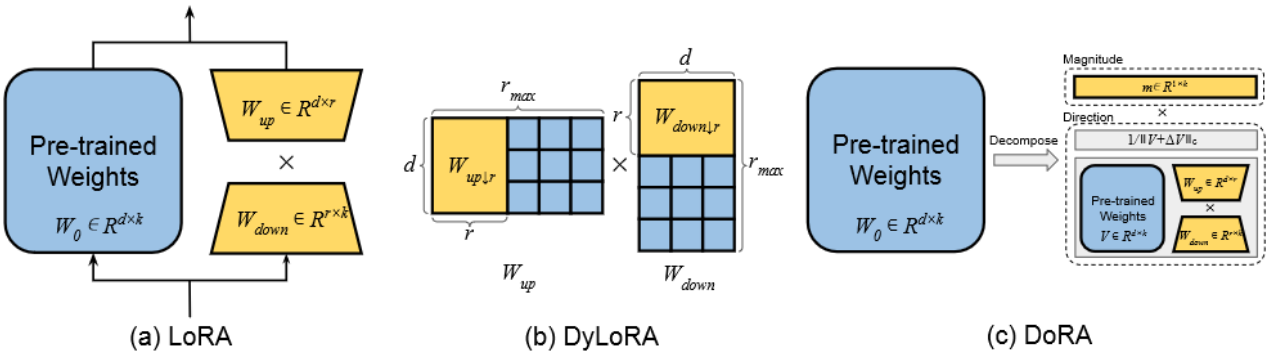
- **Diff pruning**通过将权重参数分割为局部组并有策略地一起删除，提出了一种结构化的修剪策略。
- **Bitfit** 只对每个DNN层的bias参数进行微调，对于小型模型可以取得不错的效果。但是，这种方法无法处理大型模型。
- **SPT**（sensitivity-aware visual parameter-efficient fine-tuning）识别出在微调过程中由损失减少所衡量的敏感参数。这种敏感度是通过在微调前进行一次前向和后向传递后，基于一阶泰勒展开计算得出的。接下来，SPT找到权重矩阵中敏感参数数量超过预设阈值的权重矩阵，然后应用选定的PEFT技术（例如LoRA和Adapter）对这些目标权重进行结构化调整，以实现微调。 (发表于2023 ICCV Oral)

重参数化微调

重参数化微调在训练期间引入了额外的低秩可训练参数，然后将其与原始模型集成来推理以保持推理速度。该方法分为**低秩分解**和 **LoRA 衍生**两大类。

早期的研究**Intrinsic SAID**（*Intrinsic dimensionality explains the effectiveness of language model finetuning*）表明，常见的预训练模型表现出极低的内在维度。换句话说，可以找到对整个参数空间进行有效微调的低维重参数化。

Intrinsic SAID 是在 LLM 微调过程中研究内在维度特征的开创性工作。然而，最广泛认可的重参数化技术还是 LoRA (Low Rank Adaptation)。



(a) LoRA (b) DyLoRA (c) DoRA

对于给定的预训练权重矩阵，LoRA 引入了两个可训练的权重矩阵与预训练权重并行操作，从而实现结果对预训练组件输出的增量更新，从而封装了特定于任务的知识。在训练开始时，使用随机高斯分布对降维矩阵进行初始化，而升维矩阵初始化为零，确保增量最初保持为零值。微调完成后，LoRA 的自适应权重将与预先训练的主干权重无缝集成。这种集成确保 LoRA 保持模型的效率，在推理过程中不会增加额外的负担。

在 LoRA 训练中，选择合适的 rank 一直是一个具有挑战性的问题。为了解决这个问题，出现了一些工作：

- **DyLoRA**在预定义的训练预算内训练了一系列的具有不同秩的 LoRA 模块。DyLoRA 在训练过程的每次迭代中动态选择一个秩。则两个矩阵针对所选的秩定制，从而产生两个空间截取的版本，并且本次迭代期间后续的前向和反向传播将被限制在选定位置，而不是完整权重。通过这种动态且免搜索的方法，DyLoRA 显著减少了为特定任务找到最佳且固定的 LoRA 秩所需的训练时间。
- **AdaLoRA**使用奇异值分解 (SVD) 重新表述整个增量变换，表示为 $\Delta W = P\Lambda Q$，三个矩阵都设置可学习。其中 $P \in \mathbb{R}^{d\times r}$ 和 $Q \in \mathbb{R}^{r\times k}$ 正交， $\Lambda \in \mathbb{R}^{r\times r}$ 是包含奇异值的对角矩阵。训练过程中，奇异值被根据其重要性分数进行迭代修剪，这些重要性分数是根据梯度权重乘积大小的移动平均值构建的。并构建额外的正则化项确保 P, Q 之间的正交性。这种自适应方法使模型能够动态调整每个 LoRA 模块内的秩，根据权重矩阵的重要性有效管理其参数计数。
- **SoRA**(Sparse Low-rank Adaptation)中认为 AdaLoRA 中使用的重要性分数是启发式构建的，缺乏严格的理论动机。此外，移动平均运算和额外的正则项的计算在训练期间引入了额外的计算成本。为了解决这个问题，SoRA 消除正交性前提，直接应用和优化在降维输出后插入的门控向量 ，其通过 L1 损失的近端梯度迭代的变体进行更新。(EMNLP 2023)

除了 **LoRA** 以外，其他几种具有巨大潜力的重参数化技术正在兴起。

- **Compacter** 将降维和升维权重进一步简化为$W = \sum_{i=1}^{n} Ai \otimes Bi$,二者使用 Kronecker 积组合。通过将 A_i 指定为共享参数并使用两个低秩矩阵的乘积重新参数化 B_i来进一步减少参数数量，从而有效地将参数复杂度从 $O(rd)$降低到 $O(r + d)$。
- **KronA** 和**KAdaptation** 也采用 Kronecker 乘积来重新参数化适配器权重，旨在实现参数减少。
- **DoRA** (Weight-Decomposed Low-Rank Adaptation) 将模型权重分解为幅度向量 m 和方向矩阵 V 的组合。随后 DoRA 对 m 和 V 采用独特的微调策略。虽然两者都是可调的，但实际只有 V 进行 LoRA 形式的重参数化，对其加上基于降维升维组合的附加权重。DoRA 在各种任务和模型中始终优于 LoRA。(ICML 2024 Oral)

混合微调

各种 PEFT 方法的功效在不同的任务中可能存在显着差异。因此，许多研究旨在结合不同 PEFT 方法的优点，或通过分析这些方法之间的相似性来寻求建立统一的视角。

- **UniPELT**将 LoRA, prefix-tuning, 和 adapters 集成到每个 Transformer 块中。为了控制哪些 PEFT 子模块应该被激活，他们还引入了门控机制。该机制由三个小 FFN 组成，每个 FFN 产生一个 0~1 的标量值 G，然后分别应用于 LoRA、前缀和适配器矩阵。在各种设置中，UniPELT 始终显示出准确度的提高，范围为 1% 到 4%。

- **S4**（ICLR 2023）探索了几种 PEFT 方法即 Adapter (A), Prefix (P), BitFit (B), LoRA (L)的设计空间，以揭示底层设计模式。经过一系列实验，他们发现，第一：对 Transformer 层应用主轴分组划分后得到的四个层组具有如下特点，组中的层具有相似的行为，这意味着应该应用相似的 PEFT 策略。第二：将可训练参数的数量均匀分配到各层。第三：调整所有组。第四：在不同组中分配不同的PEFT 策略，由此产生的具有最佳性能的设计空间是（$G1:A,L,G2:A,P,G3:A,P,B,G4:P,B,L$）。 文章：*Parameter-Efficient Fine-Tuning Design Spaces*
- **LLM-Adapters**（EMNLP 2023）构建了一个易于使用的框架，将各种 PEFT 技术合并到 LLM 中。通过跨多个数据集的全面基准测试，该研究揭示了几个关键见解：

  1：series adapters, parallel adapters, LoRA 最有效的位置分别是在 MLP 层之后、MLP 层旁边，和同时跟随注意力层和 MLP 层。

  2：与规模较大的 LLM 相比，利用 PEFT 的规模较小的 LLM 可以在某些任务上取得有竞争力甚至更好的结果。

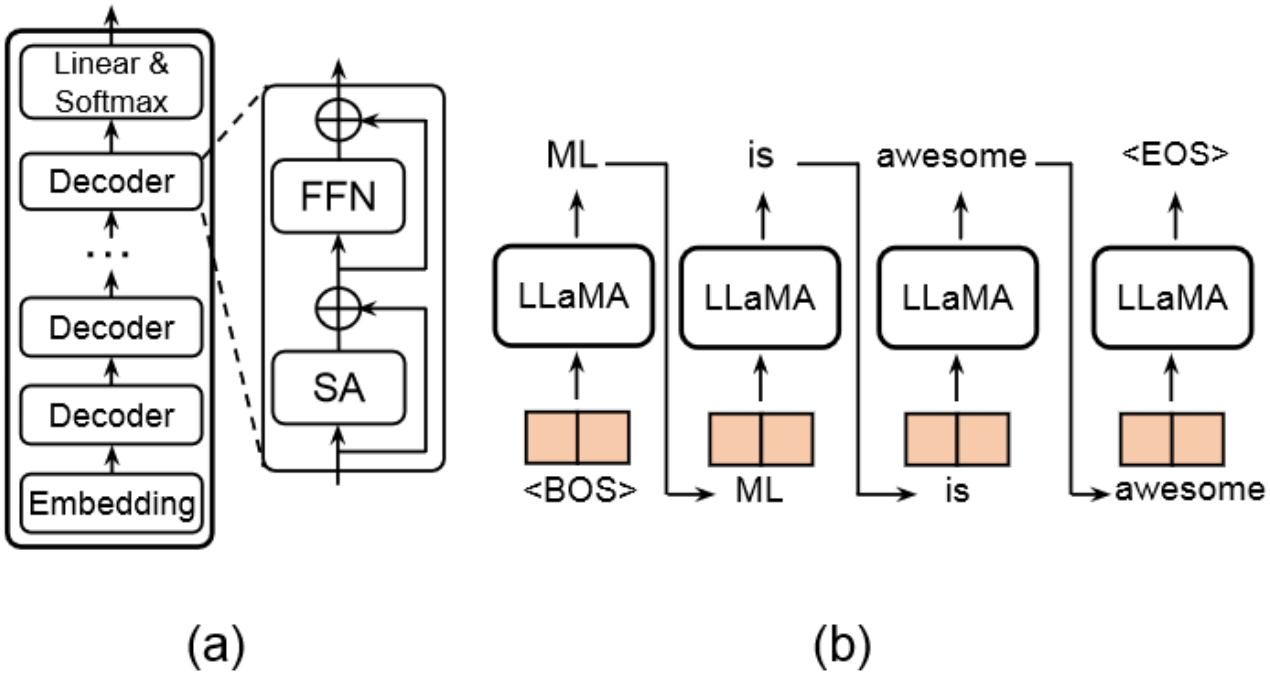
  3：通过适当的分布内数据微调，较小的模型能够在特定任务的性能上超越较大的模型。
- **AUTOPEFT**（TACL 2024）首先建立一个搜索空间，其中包括 serial adapters, parallel adapters, prefix tuning。之后提出了一种基于高维多维贝叶斯优化的有效 NAS 方法。

高效PEFT设计

从计算的角度来看，处理延迟和峰值内存开销是需要考虑的关键因素。以下几种策略都旨在最小化模型资源消耗的同时增强模型性能。

提高PEFT效率的KV缓存管理

LLM的核心是一个自回归的Transformer模型，如图所示。自回归特性成为设计推理系统的主要挑战，因为每次生成新的token时，整个LLM模型必须将所有权重从不同的内存转移到图形处理器的内存中，这对单用户任务调度或多用户工作负载平衡非常不友好。



服务自回归范式的挑战部分是必须缓存和保存所有先前的序列以供下一次进行迭代，从先前序列生成的缓存激活存储为键值缓存(KV-cache)。KV-cache 的存储将消耗内存空间和 IO 性能，导致工作负载内存有限，系统的计算能力利用率不足 。

之前的工作提出了一系列解决方案，例如 KV-cache 控制管理【Efficient Memory Management for Large Language Model Serving with PagedAttention】或 KV-cache 压缩【High-throughput Generative Inference of Large Language Model with a Single GPU】，以提高吞吐量或减少延迟。

在设计 PEFT 方法时，考虑 KV-cache 的特性以补充其功能至关重要。例如，当在推理阶段应用 soft prompt 时，有效利用 KV-cache 来处理这些附加输入可以通过确保 prompt 相关的数据易于访问从而帮助加快响应时间。

PEFT的剪枝策略

什么是剪枝：深度学习网络模型从卷积层到全连接层存在着大量冗余的参数，大量神经元激活值趋近于0，将这些神经元去除后可以表现出同样的模型表达能力，这种情况被称为过参数化，而对应的技术则被称为模型剪枝。

模型剪枝主要分为**结构化剪枝**和**非结构化剪枝**，非结构化剪枝去除不重要的神经元，相应地，被剪除的神经元和其他神经元之间的连接在计算时会被忽略。由于剪枝后的模型通常很稀疏，并且破坏了原有模型的结构，所以这类方法被称为非结构化剪枝。与非结构化剪枝相对应的是结构化剪枝，结构化剪枝通常以滤波器或者整个网络层为基本单位进行剪枝。一个滤波器被剪枝，那么其前一个特征图和下一个特征图都会发生相应的变化，但是模型的结构却没有被破坏，仍然能够通过 GPU 或其他硬件来加速，因此这类方法被称之为结构化剪枝。

剪枝可以大大提高PEFT的效率。

- **AdapterDrop**探索了从 AdapterFusion 中删除浅层 Transformer 层适配器和多任务适配器，这表明剪枝可以提高训练和推理效率，同时性能下降最小。
- **SparseAdapter**研究了不同的修剪方法，发现高稀疏率(80%)可以优于标准适配器。此外，大稀疏配置在增加瓶颈维度的同时保持恒定的参数预算（例如以 50% 的稀疏度加倍维度），大大增强了模型的容量，从而提高了性能。
- **SPLoRA**对 LoRA 中降维和升维的权重采用基于通道的剪枝。这种修剪不仅影响源权重，而且影响 LoRA 的这两个权重。

- **LoRAPruning** 不仅对预训练模型权重采用结构化剪枝，而且对 LoRA 权重也采用结构化剪枝。非结构化 LoRA 剪枝方法主要侧重于稀疏模型权重，同时保持 LoRA 权重的密集性，从而使权重合并难以实现，与此相反，LoRAPruning 可以轻松合并权重。此外，这项工作还引入了一种新颖的标准，利用 LoRA 梯度作为预训练权重梯度的近似值，从而能够估计权重重要性。
- **ProPETL**（ACL 2023）跨层和任务构建单个共享原型（例如 adapter, prefix, LoRA）。此外，ProPETL 学习二进制掩码来修剪不同层和任务中的不同子网络。因此，参数可以在跨层和不同任务中重用，大大提高了参数效率。

PEFT的量化策略

量化，是一种在保证模型效果基本不降低的前提下，通过降低参数的精度，来减少模型对于计算资源的需求的方法。

具体来说，量化技术通过将模型的权重和激活值从浮点数转换为低精度的整数或定点数，从而大大减少了推理所需的计算资源和内存。

- **BI-Adapter**(ICCV 2023)通过调查适配器的损耗情况，发现适配器对参数空间中的噪声具有抵抗力。基于这一见解，作者引入了一种基于聚类的量化方法。值得注意的是，他们证明了适配器的 1-bit 量化不仅可以最大限度地减少存储需求，而且可以在所有精度设置中实现卓越的性能。
- **PEQA** (Parameter-Efficient and Quantization-aware Adaptation) 使用两阶段管道来实现参数高效且具有量化感知的微调。第一阶段，预训练的 FFN 权重矩阵被量化为一个通道放缩量构成的向量与量化权重之间的乘积。第二阶段，量化权重保持固定，仅对通道放缩向量微调。这种方法不仅保证了内存效率，而且有利于参数效率。
- **QLoRA** 提出了几种新技术，包括 4-bit NormalFloat、双重量化和分页优化器，将 4-bit 量化预训练语言模型反向传播到 LoRA 中。这些技术可以在单个 48GB GPU 上对 65B 语言模型进行微调，同时保持与完整 16-bit 微调类似的性能。与原始 LoRA 类似，QLoRA 将固定的零初始化 LoRA 权重附加到量化的预训练模型作为训练起点。

然而，当应用极低 bit（例如 2-bit）量化时，巨大的量化误差会对 LoRA 微调的初始化产生负面影响。为了解决这个问题，研究者们又提出了几种量化策略来消除量化误差。

- **LoftQ** (LoRA-Fine-Tuningaware Quantization) 提出了一种创新框架，为后续的 LoRA 微调提供了量化主干权重和 LoRA 权重的优越初始化点。该方法通过在网络初始化期间优化 Frobenius 范数目标来解决由量化引起的差异，该目标考虑了 LoRA 权重和量化的预训练骨干网络。LoftQ 在 2-bit 量化方面表现出优于 QLoRA 的性能，并且对下游任务具有更强的泛化能力。
- **QA-LoRA** 解决了 QLoRA 的另一个限制，即在微调后难以保留其量化属性。在 QLoRA 中，量化的预训练权重（NF4）必须恢复为 FP16，以在权重合并期间匹配 LoRA 权重精度（FP16）。相反，QA-LoRA 使用 INT4 量化并引入分组运算符以在推理阶段实现量化，因此与 QLoRA 相比提高了效率和准确性。
- **BitDelta** （2024.02）引入了一种新颖的 1-bit 训练后量化方法，该方法作用于微调模型与底层预训练模型之间的权重增量。通过利用单个全精度基本模型以及高效批处理的 1-bit 增量，显著简化了共享服务器上多个微调模型的部署。

PEFT应用

针对LLM的应用：

- 视觉指导（Visual Instruct Following）：VL-BART、MiniGPT-4和LLaVA在内的几项研究已经成功地扩展了最初为纯文本设计的LLM的能力，以理解和生成对视觉输入的响应。这些增强的模型，即Visual Instruct Follow LLMs，可以处理图像和文本以产生文本响应，这些响应可以在图像字幕和视觉问答（VQA）等任务上进行基准测试。
- 持续学习（Continual Learning）：CL 的目标是在一个模型中随着时间的推移学习一系列新任务，该模型在对话系统、信息提取系统和问答系统等场景中具有广泛应用。CL 的主要挑战是灾难性遗忘。基于架构的方法，通过维护特定于任务的参数来解决 CL 模型中每个新任务的输入。因此，很自然地利用 PEFT 方法来执行 CL 任务。
- 上下文窗口拓展（Context Window Extension）：LLM 通常使用预定义的上下文大小进行训练。如 LLaMA 和 LLaMA2 的预定义上下文大小分别为 2048 和 4096 个 token。旋转位置编码 （RoPE ）具有较弱的外推属性，这意味着在输入长度超过预定义上下文长度的情况下，性能明显下降。为了解决这个问题，一个简单的解决方案是将预训练的 LLM 微调到更长的上下文。然而，这会导致计算成本随上下文大小呈二次方上升，从而导致内存和处理资源紧张。

针对Vision Transformer(VIT)的应用：

- 图像分类：视觉数据集上的图像分类是一种常见的需求，并且具有广泛的应用。而预训练然后微调范式是一种广泛的策略。多种方法利用 PEFT 技术来实现高效的模型调整。
- 视频识别：一些工作考虑了更具挑战性的适应问题，将 ViT 转移到具有更大域隔阂的下游任务。

针对Vision-Language Alignment (VLA)的应用：

VLA 如 CLIP、ALIGN、DeCLIP 和 FLAVA，旨在学习在统一表示中对齐的良好图像和文本特征空间。每个 VLA 通常由提取各自特征的单独图像和文本编码器组成。这些模型利用对比学习来有效地对齐图像和文本特征。利用微调来提高 VLA 在特定数据集或任务上的性能。

但微调整个模型需要大量计算。例如，微调 CLIP-RN50x64 需要 32,768 的批大小以及在 592 块V100 GPU 上进行 18 天的训练。此外，对较小数据集的全面微调通常会导致灾难性的遗忘。

为了应对这些挑战，并从 PEFT 技术在 NLP 中的成功中汲取灵感，一系列 PEFT 策略被提出并在 VLA 模型中实现，例如语义分割、点云理解、视频理解、视觉推理、时间动作检测等。

针对Diffusion Models的应用：

等等…….

总结

在当前以大型模型和大型数据集为主导的时代，PEFT脱颖而出，可以有效地使模型适应下游任务。该技术通过解决传统全模型微调带来的重大挑战而获得吸引力，传统全模型微调通常对普通用户提出难以满足的计算和数据需求。

未来可能的方向：

1. 简化超参数调整：PEFT 的有效性通常对其超参数敏感，例如适配器瓶颈尺寸、LoRA 秩以及不同附加性 PEFT 层的放置。手动调整这些超参数将花费大量精力。因此，未来的努力可以集中在开发更少依赖手动调整这些参数的方法，或者自动找到最佳的超参数设置。
2. 建立统一的基准：尽管存在像 HuggingFace 的 PEFT 和 AdapterHub 这样的库，但仍然缺乏全面的 PEFT 基准。这种差距阻碍了公平比较不同 PEFT 方法的性能和效率的能力。
3. 提升训练效率：PEFT 的假定参数效率并不总是与训练期间的计算和内存节省一致。鉴于可训练参数在预训练模型架构中交织在一起，因此在微调期间通常需要计算和存储完整模型的梯度。潜在的解决方案在于集成模型压缩技术，例如剪枝和量化，以及专门设计用于在 PEFT 调整期间优化内存的创新。进一步提高 PEFT 方法的计算效率的研究势在必行。
4. 探索缩放定律：最初为较小的 Transformer 模型开发的 PEFT 方法的设计和有效性不一定适用于较大的模型。 随着基础模型规模的增加，识别和调整保持有效的 PEFT 策略至关重要。
5. 服务更多模型和任务：根据模型的独特特征设计定制的 PEFT 方法，例如 Sora、Mamba 和 LVM，可以解锁新的应用场景和机会。