【大模型微调】

一文掌握 7 种大模型微调的方法

一、大型模型微调的基础理论

大型语言模型(LLM)的训练过程通常分为两大阶段：

阶段一：预训练阶段

在这个阶段，大型模型会在大规模的无标签数据集上接受训练，目标是使模型掌握语言的统计特征和基础知识。此期间，模型将掌握词汇的含义、句子的构造规则以及文本的基本信息和上下文。

需特别指出，预训练实质上是一种无监督学习过程。完成预训练的模型，亦即基座模型(Base Model)，拥有了普遍适用的预测能力。例如，GLM-130B 模型、OpenAI 的四个主要模型均属于基座模型。

阶段二：微调阶段

预训练完成的模型接下来会在针对性的任务数据集上接受更进一步的训练。这一阶段主要涉及对模型权重的细微调整，使其更好地适配具体任务。最终形成的模型将具备不同的能力，如 gpt code 系列、gpt text 系列、ChatGLM-6B 等。

那么，何为大型模型微调？

直观上，大型模型微调即是向模型“输入”更多信息，对模型的特定功能进行“优化”，通过输入特定领域的数据集，使模型学习该领域知识，从而优化大模型在特定领域的 NLP 任务中的表现，如情感分析、实体识别、文本分类、对话生成等。

为何微调至关重要？

其核心理由是，微调能够“装备”大模型以更精细化的功能，例如整合本地知识库进行搜索、针对特定领域问题构建问答系统等。

以 VisualGLM 为例，作为一个通用多模态模型，当应用于医学影像判别时，就需要输入医学影像领域的数据集以

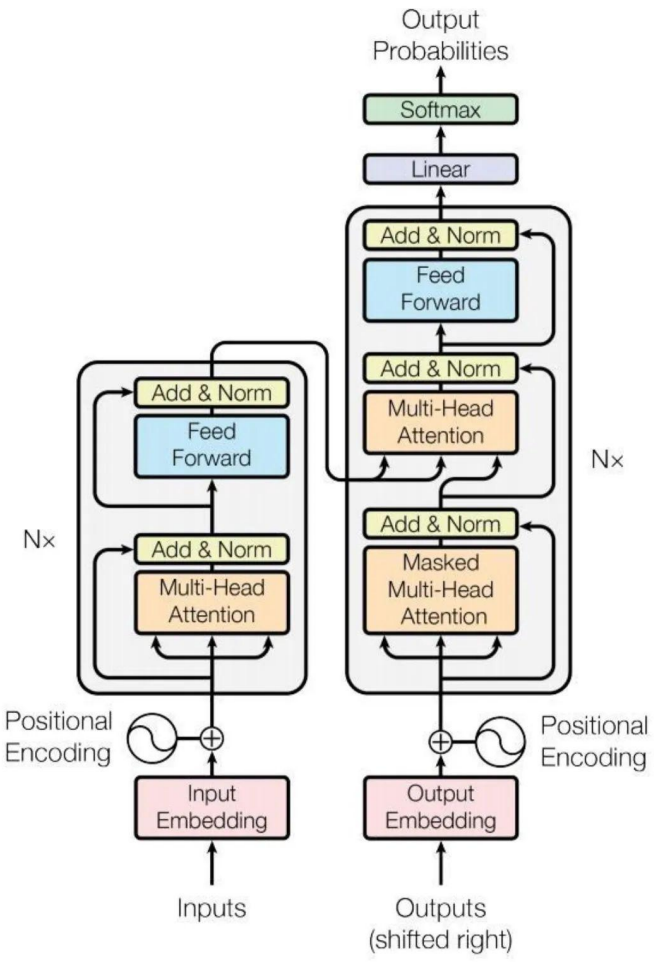
进行微调，以此提升模型在医学影像图像识别方面的表现。

这与机器学习模型的超参数优化类似，只有在调整超参数后，模型才能更好地适应当前数据集；同时，大型模型可以经历多轮微调，每次微调都是对模型能力的优化，即我们可以在现有的、已经具备一定能力的大模型基础上进一步进行微调。

二、大型模型的经典网络结构

以 GPT 系列中的 Transformer 为例，这种深度学习模型结构通过自注意力机制等技巧解决了相关问题。正是得益于 Transformer 架构，基于 GPT 的大型语言模型取得了显著的进展。

Transformer 模型架构包含了众多模块，而我们讨论的各种微调技术通常是对这些模块中的特定部分进行优化，以实现微调目的。



要深入理解各类微调手段，首先需要对网络架构有一个基本的认识。以下以 Transformer 为例，阐述各个模块的作用：

1. 输入嵌入层（Input Embedding）

- 输入（Inputs）：模型的输入环节，通常为单词或符号序列。
- 输入嵌入（Input Embedding）：此步骤将输入序列（例如句中的每个单词）转化为嵌入表示，即能够表征单词语义信息的高维向量。
- 位置编码（Positional Encoding）：鉴于 Transformer 不依赖序列，位置编码旨在提供序列中单词位置的信息，这些编码添加到输入嵌入中，确保模型即便同时处理输入也能够利用单词的顺序信息。

2. 编码器层（Encoder，左边）

- Nx：指示有 N 个相同的编码器层叠加而成。每个编码器层包括两个主要子层：多头自注意力机制和前馈神经网络。
- 多头自注意力（Multi-Head Attention）：注意力机制允许模型在处理每个单词时考虑到输入序列中的所有单词。多头部分表示模型并行学习输入数据的不同表示。
- 残差连接和归一化（Add & Norm）：注意力层后面跟着残差连接和层归一化，有助于防止深层网络中的梯度消失问题，并稳定训练过程。
- 前馈神经网络（Feed Forward）：全连接神经网络处理自注意力层的输出，包含两个线性变换和一个非线性激活函数。

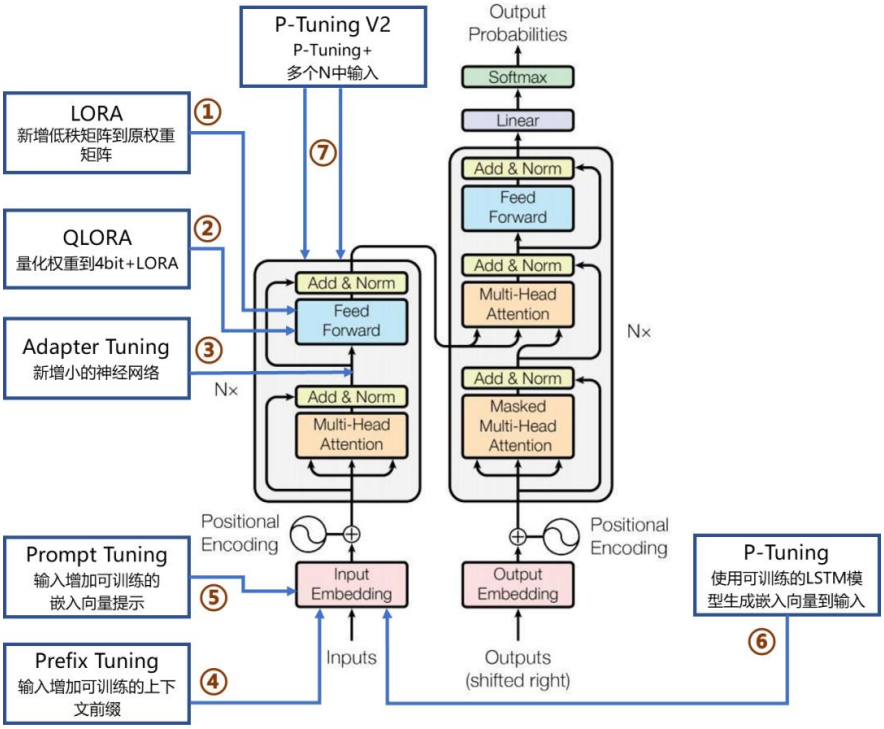
3. 解码器层（Decoder，右侧）

- 解码器亦包含多个相同的层，每层包括三个主要子层：掩蔽的多头自注意力机制、多头自注意力机制和前馈神经网络。

- 掩蔽多头自注意力（Masked Multi-Head Attention）：与编码器的多头自注意力机制类似，但为确保解码顺序性，掩蔽操作确保预测仅依赖于之前的输出。
- 前馈神经网络（Feed Forward）：与编码器相同，每个子层之后也有加法和归一化步骤。

4. 输出嵌入层和输出过程

- 解码器端的嵌入层将目标序列转换为向量形式。
- 线性层（Linear）和 Softmax 层：解码器的输出通过线性层映射到一个更大的词汇空间，Softmax 函数将输出转换为概率分布。



三、大型模型微调的技术手段

1、LoRA

LoRA（Low-Rank Adaptation）是一种旨在微调大型预训练语言模型（如 GPT-3 或 BERT）的技术。其核心理念在于，在模型的决定性层次中引入小型、低秩的矩阵来实现模型行为的微调，而无需对整个模型结构进行大幅度修改。

这种方法的优势在于，在不显著增加额外计算负担的前提下，能够有效地微调模型，同时保留模型原有的性能水准。

LoRA 的操作流程如下：

- 确定微调目标权重矩阵：首先在大型模型（例如 GPT）中识别出需要微调的权重矩阵，这些矩阵一般位于模型

的多头自注意力和前馈神经网络部分。

- 引入两个低秩矩阵：然后，引入两个维度较小的低秩矩阵 A 和 B。假设原始权重矩阵的尺寸为 dd，则 A 和 B 的尺寸可能为 dr 和 r*d，其中 r 远小于 d。
- 计算低秩更新：通过这两个低秩矩阵的乘积 AB 来生成一个新矩阵，其秩（即 r）远小于原始权重矩阵的秩。这个乘积实际上是对原始权重矩阵的一种低秩近似调整。
- 结合原始权重：最终，新生成的低秩矩阵 AB 被叠加到原始权重矩阵上。因此，原始权重经过了微调，但大部分权重维持不变。这个过程可以用数学表达式描述为：新权重 = 原始权重 + AB。

以一个具体实例来说，假设我们手头有一个大型语言模型，它通常用于执行广泛的自然语言处理任务。现在，我们打算将其微调，使其在处理医疗健康相关的文本上更为擅长。

采用 LoRA 方法，我们无需直接修改模型现有的大量权重。相反，只需在模型的关键部位引入低秩矩阵，并通过这些矩阵的乘积来进行有效的权重调整。这样一来，模型就能更好地适应医疗健康领域的专业语言和术语，同时也避免了大规模权重调整和重新训练的必要。

2、QLoRA

QLoRA（Quantized Low-Rank Adaptation）是一种结合了 LoRA（Low-Rank Adaptation）方法与深度量化技术的高效模型微调手段。QLoRA 的核心在于：

- 量化技术：QLoRA 采用创新的技术将预训练模型量化为 4 位。这一技术包括低精度存储数据类型（4-bit NormalFloat，简称 NF4）和计算数据类型（16-bit BrainFloat）。这种做法极大地减少了模型存储需求，同时保持了模型精度的最小损失。
- 量化操作：在 4 位量化中，每个权重由 4 个比特表示，量化过程中需选择最重要的值并将它们映射到 16 个可能的值之一。首先确定量化范围（例如-1 到 1），然后将这个范围分成 16 个区间，每个区间对应一个 4-bit 值。然后，原始的 32 位浮点数值将映射到最近的量化区间值上。
- 微调阶段：在训练期间，QLoRA 先以 4-bit 格式加载模型，训练时将数值反量化到 bf16 进行训练，这样大幅减少了训练所需的显存。例如，33B 的 LLaMA 模型可以在 24 GB 的显卡上进行训练。

量化过程的挑战在于设计合适的映射和量化策略，以最小化精度损失对性能的影响。在大型模型中，这种方法可以显著减少内存和计算需求，使得在资源有限的环境下部署和训练成为可能。

3、适配器调整（Adapter Tuning）

与 LoRA 技术类似，适配器调整的目标是在保留预训练模型原始参数不变的前提下，使模型能够适应新的任务。适配器调整的方法是在模型的每个层或选定层之间插入小型神经网络模块，称为“适配器”。这些适配器是可训练的，而原始模型的参数则保持不变。

适配器调整的关键步骤包括：

- 以预训练模型为基础：初始阶段，我们拥有一个已经经过预训练的大型模型，如 BERT 或 GPT，该模型已经学习了丰富的语言特征和模式。
- 插入适配器：在预训练模型的每个层或指定层中，我们插入适配器。适配器是小型的神经网络，一般包含少量层次，并且参数规模相对较小。
- 维持预训练参数不变：在微调过程中，原有的预训练模型参数保持不变。我们不直接调整这些参数，而是专注于适配器的参数训练。
- 训练适配器：适配器的参数会根据特定任务的数据进行训练，使适配器能够学习如何根据任务调整模型的行为。
- 针对任务的调整：通过这种方式，模型能够对每个特定任务进行微调，同时不影响模型其他部分的通用性能。适配器有助于模型更好地理解和处理与特定任务相关的特殊模式和数据。
- 高效与灵活：由于只有部分参数被调整，适配器调整方法相比于全模型微调更为高效，并且允许模型迅速适应新任务。

例如，如果我们有一个大型文本生成模型，它通常用于执行广泛的文本生成任务。若要将其微调以生成专业的金融报告，我们可以在模型的关键层中加入适配器。在微调过程中，仅有适配器的参数会根据金融领域的数据进行更新，使得模型更好地适应金融报告的写作风格和术语，同时避免对整个模型架构进行大幅度调整。

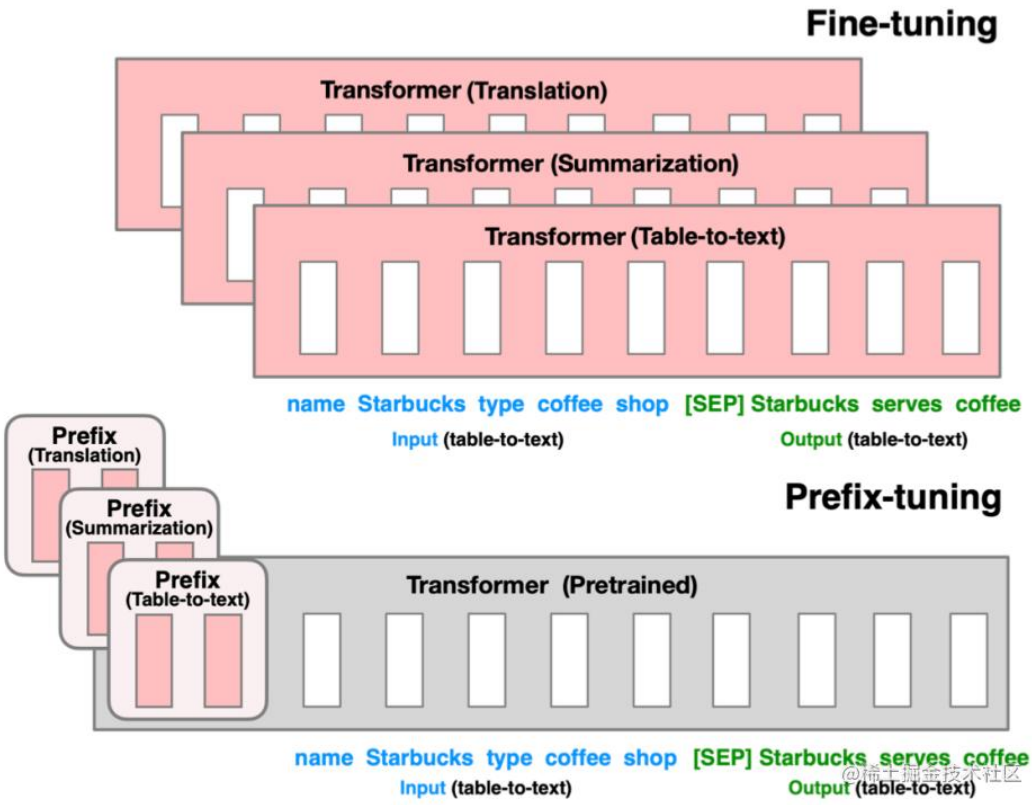
LoRA 与适配器调整的主要区别在于：

- LoRA：在模型的权重矩阵中引入低秩矩阵来实现微调。这些低秩矩阵作为原有权重矩阵的修改项，在实际计算时对原有权重矩阵进行调整。

- 适配器调整：通过在模型各层中添加小型神经网络模块，即“适配器”，来实现微调。适配器独立于模型的主体结构，仅适配器的参数在微调过程中更新，而模型的其他预训练参数保持不变。

4、前缀调整（Prefix Tuning）

与传统的微调范式不同，前缀调整提出了一种新的策略，即在预训练的语言模型（LM）输入序列前添加可训练、任务特定的前缀，从而实现针对不同任务的微调。这意味着我们可以为不同任务保存不同的前缀，而不是为每个任务保存一整套微调后的模型权重，从而节省了大量的存储空间和微调成本。



前缀实际上是一种连续可微的虚拟标记（Soft Prompt/Continuous Prompt），与离散的 Token 相比，它们更易于优化并且效果更佳。这种方法的优势在于不需要调整模型的所有权重，而是通过在输入中添加前缀来调整模型的行为，从而节省大量的计算资源，同时使得单一模型能够适应多种不同的任务。前缀可以是固定的（即手动设计的静态提示）或可训练的（即模型在训练过程中学习的动态提示）。

5、提示调整（Prompt Tuning）

提示调整是一种在预训练语言模型输入中引入可学习嵌入向量作为提示的微调方法。这些可训练的提示向量在训练过程中更新，以指导模型输出更适合特定任务的响应。

提示调整与前缀调整都涉及在输入数据中添加可学习的向量，这些向量是在输入层添加的，但两者的策略和目的不同：

- 提示调整：旨在模仿自然语言中的提示形式，将可学习向量（通常称为提示标记）设计为模型针对特定任务生成特定类型输出的引导。这些向量通常被视为任务指导信息的一部分，倾向于使用较少的向量来模仿传统的自然语言提示。
- 前缀调整：可学习前缀更多地用于提供输入数据的直接上下文信息，作为模型内部表示的一部分，可以影响整个模型的行为。

以下是两者的训练示例，以说明它们的不同：

- 提示调整示例：

输入序列: [Prompt1][Prompt2] “这部电影令人振奋。”

问题: 评价这部电影的情感倾向。

答案: 模型需要预测情感倾向（例如“积极”）

提示: 没有明确的外部提示，[Prompt1][Prompt2]作为引导模型的内部提示，这里的问题是隐含的，即判断文本中表达的情感倾向。

- 前缀调整示例：

输入序列: [Prefix1][Prefix2][Prefix3] “I want to watch a movie.”

问题: 根据前缀生成后续的自然语言文本。

答案: 模型生成的文本，如“that is exciting and fun.”

提示: 前缀本身提供上下文信息，没有单独的外部提示。

6、P-Tuning

P-Tuning（基于提示的微调）和提示调整都是为了调整大型预训练语言模型（如 GPT 系列）以适应特定任务而设计的技术。两者都利用预训练的语言模型执行特定的下游任务，如文本分类、情感分析等，并使用某种形式的“提示”或“指导”来引导模型输出，以更好地适应特定任务。

提示调整与 P-Tuning 的主要区别在于：

- 提示调整：使用静态的、可训练的虚拟标记嵌入，在初始化后保持固定，除非在训练过程中更新。这种方法相对简单，因为它只涉及调整一组固定的嵌入参数，在处理多种任务时表现良好，但可能在处理特别复杂或需要细粒度控制的任务时受限。
- P-Tuning：使用一个可训练的 LSTM 模型（称为提示编码器 prompt_encoder）来动态生成虚拟标记嵌入，允许根据输入数据的不同生成不同的嵌入，提供更高的灵活性和适应性，适合需要精细控制和理解复杂上下文的任务。这种方法相对复杂，因为它涉及一个额外的 LSTM 模型来生成虚拟标记嵌入。

P-Tuning 中使用 LSTM（长短期记忆网络）作为生成虚拟标记嵌入的工具，利用了 LSTM 的以下优势：

- 更好的适应性和灵活性：LSTM 可以捕捉输入数据中的时间序列特征，更好地理解和适应复杂的、顺序依赖的任务，如文本生成或序列标注。
- 改进的上下文理解：LSTM 因其循环结构，擅长处理和理解长期依赖关系和复杂的上下文信息。

- 参数共享和泛化能力：在 P-Tuning 中，LSTM 模型的参数可以在多个任务之间共享，这提高了模型的泛化能力，并减少了针对每个单独任务的训练需求。而在提示调整中，每个任务通常都有其独立的虚拟标记嵌入，这可能限制了跨任务泛化的能力。

这些特性使得 LSTM 特别适合处理复杂任务和需要细粒度控制的应用场景。然而，这些优势也伴随着更高的计算复杂度和资源需求，因此在实际应用中需要根据具体需求和资源限制来权衡使用 LSTM 的决策。

7、P-Tuning v2

P-Tuning v2 是 P-Tuning 的进一步改进版，在 P-Tuning 中，连续提示被插入到输入序列的嵌入层中，除了语言模型的输入层，其他层的提示嵌入都来自于上一层。这种设计存在两个问题：

- 第一，它限制了优化参数的数量。由于模型的输入文本长度是固定的，通常为 512，因此提示的长度不能过长。

- 第二，当模型层数很深时，微调时模型的稳定性难以保证；模型层数越深，第一层输入的提示对后面层的影响难以预测，这会影响模型的稳定性。

P-Tuning v2 的改进在于，不仅在第一层插入连续提示，而是在多层都插入连续提示，且层与层之间的连续提示是相互独立的。这样，在模型微调时，可训练的参数量增加了，P-Tuning v2 在应对复杂的自然语言理解(NLU)任务和小型模型方面，相比原始 P-Tuning 具有更出色的效能。

除了以上 PEFT，当前还存在 PILL（Pluggable Instruction Language Learning）、SSF（Scaling & Shifting Your Features）等其他类型的微调方法。

PILL 是 PEFT 的一个特定实现，特别关注于如何通过插入可训练的模块或插件来提升模型的任务适应性。这些插件被设计为与原始模型协同工作，以提高模型在处理特定任务时的效率和效果。

SSF 核心思想是对模型的特征（即模型层的输出）进行缩放（Scaling）和位移（Shifting）。简单来说，就是通过调整特征的比例和偏移量来优化模型的性能。

这种方法可以在改善模型对特定任务的响应时，不需要调整或重新训练模型中的所有参数，从而在节省计算资源的同时保持或提升模型性能。这对于处理大规模模型特别有效，因为它减少了训练和调整所需的资源和时间。

总结：大模型的微调策略

综上所述，微调是一种强大的工具，它能够使大型预训练模型适应于特定的任务和应用场景。正确选择和应用微调策略对于实现高效且有效的模型性能至关重要。

1、微调与迁移学习：微调实际上是迁移学习的一个实例，其中预训练的模型（通常在大型通用数据集上训练）被用作特定任务的起点。这种方法使得即使是对于小数据集的任务，也可以实现高效的学习

2、选择微调策略：选择哪种微调方法取决于多个因素，包括任务的复杂性、可用的数据量、计算资源和期望的性能。

例如，对于需要细粒度控制的复杂任务，P-Tuning v2 或 LSTM 基础的 P-Tuning 可能更适合。而对于计算资源有限的情况，可以选择 LoRA 或 Adapter Tuning 等方法。

3、微调与模型泛化能力：微调时需要注意的一个关键问题是保持模型的泛化能力。过度的微调可能会导致模型对特定训练数据过拟合，而忽略了其在实际应用中的泛化能力。