

# Finetune

---

## 1. 全参数微调(Full Fine-tuning)

- 操作：更新模型的所有参数
- 特点：计算和存储成本高，但通常效果最好
- 应用场景：有足够计算资源且需要最佳性能时

## 2. 参数高效微调(PEFT - Parameter-Efficient Fine-Tuning)

### LoRA (Low-Rank Adaptation)

- 操作：为原始权重矩阵添加低秩分解矩阵
- 特点：仅训练低秩适应矩阵，冻结原始参数
- 优势：显著减少可训练参数数量，同时保持性能

#### 参数更新公式

假设权重矩阵为  $W \in R^{r \times k}$ ，Lora的更新为：  $h = Wx + \triangle Wx = Wx + BAx$

其中：  $B \in R^{d \times r}$ ，  $A \in R^{r \times k}$ ，  $r \ll \min(d, k)$

- 需要注意的是原始权重的矩阵是接近满秩的，不能进行低秩近似
- 也就是把权重矩阵的变化分解为两个低秩矩阵相乘，对原始的权重矩阵进行冻结
- 更新参数的时候可以保证调整的参数数量大幅度减小
- 疑惑：为什么  $\Delta W$  可以分解成两个低秩矩阵相乘
  - 因为参数更新的  $\triangle W$  本质上是低秩的，可以用低秩矩阵去近似原来的权重矩阵，从而减少调整的参数数量
  - $\triangle W = U \Sigma V^T$ ，参数矩阵可以进行奇异值分解，而前  $r$  个大的奇异值包含了矩阵的主要信息

初始化的时候：

- A用高斯初始化（均值为0），B初始化为全零矩阵，保证训练开始时  $W$  为0

### Adapter微调

- 操作：在Transformer层之间插入小型可训练模块

- **特点：**冻结原始模型，仅训练**Adapter层**
- **优势：**参数效率高，便于多任务切换

## 插入位置

通常插入在Transformer的**两个核心子层之后**：**自注意力层和前馈神经网络**

## 模块设计

每个Adapter层包含：

- 一个降维层：将输入特征从**维度d压缩到r**( $r \ll d$ )
- 一个**非线性激活函数**：（如ReLU或者GELU）
- 一个升维层：将特征从r恢复为d

## 工作原理

**数学表达：**  $\text{Adapter}(x) = x + W_{\text{up}} \cdot \text{ReLU}(W_{\text{down}} \cdot x)$  其中：

- $W_{\text{down}} \in \mathbb{R}^{r \times d}$ ,  $W_{\text{up}} \in \mathbb{R}^{d \times r}$ , 它们为**可训练参数**
- 借鉴了**残差连接**思想，保证初始状态不影响模型

## Bias-only微调

### 工作原理

- **选择性参数更新：**仅更新神经网络中的偏置项(bias)参数，完全冻结所有权重矩阵
- **极低参数量：**偏置通常只占模型总参数的不到1%，大大减少了可训练参数