$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

2. **训练**：
   - 在训练过程中，仅更新 A 和 B 的值。
   - 随着训练的进行， B 的值逐渐变得非零。
3. **微调权重计算**：
   - 训练结束后，我们得到 A 和 B 的最终值，计算 $\Delta W = BA$ 。
   - 将 ΔW 加到 $W_{\text{pretrained}}$ 上，得到微调后的权重矩阵 $W_{\text{finetuned}}$ 。

4.3 选择哪些权重矩阵进行适配?

在有限参数预算下，应选择哪些权重矩阵进行适配以最大化下游任务性能?

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

表5：在GPT-3中应用LoRA到不同类型的注意力权重后，在WikiSQL和MultiNLI上的验证准确性，考虑到可训练参数的数量相同。同时调整W_q和W_v能够获得最佳的整体表现。我们发现给定数据集的随机种子间的标准偏差是一致的，我们在第一列中报告了这一点。

主要发现：

- 同时适配 W_q 、 W_k、W_v、W_o 提供了最佳性能。
- 低秩（例如rank为4）足以在 ΔW 中捕获足够信息，表现优于单一类型权重适配但具有更高秩的策略。

4.4 为什么 LoRA 在 Q, K, V, O 上有效?

LoRA（Low-Rank Adaptation）在 Transformer 模型的 Q（Query）、K（Key）、V（Value）和 O（Output）矩阵上有效果的原因可以归结为这些矩阵在注意力机制中的核心作用以及 LoRA 方法在降低参数数量的同时保持或提升模型性能的能力。具体原因如下：

4.4.1 Self-Attention

- **Q（Query）矩阵**：用于生成查询向量，决定模型在注意力机制中对输入的关注程度。
- **K（Key）矩阵**：用于生成键向量，与查询向量计算相似度，帮助确定注意力分布。
- **V（Value）矩阵**：用于生成数值向量，实际传递注意力机制计算的输出。
- **O（Output）矩阵**：用于将多头注意力的输出合并并映射回原始维度。

4.4.2 信息传播的关键路径

Q、K、V 和 O 矩阵在信息传播和特征表示中起着关键作用：

- **查询与键的交互**：Q 和 K 的交互决定了注意力分布，影响模型对输入序列的不同部分的关注度。
- **数值的加权求和**：V 矩阵通过加权求和操作，将注意力分布转化为具体的输出。
- **多头输出的整合**：O 矩阵整合多头注意力的输出，提供最终的特征表示。

4.4.3 LoRA 的低秩近似

LoRA 通过将权重矩阵分解为两个低秩矩阵（例如 $W \approx BA$），减少了参数数量，降低了计算和存储成本，同时保持模型性能：

- **参数压缩**：Q、K、V 和 O 矩阵通常包含大量参数，LoRA 的低秩分解显著减少了需要优化的参数数量。
- **性能保持**：低秩矩阵能够捕捉到原始矩阵的主要信息，确保模型性能不受显著影响。

4.5 LoRA中的最优秩 r 的选择

作者探讨了不同秩 r 对模型性能的影响，并确定最小的有效秩，即"内在秩"。

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

表5：在GPT-3中应用LoRA到不同类型的注意力权重后，在WikiSQL和MultiNLI上的验证准确性，考虑到可训练参数的数量相同。同时调整W_q和W_v能够获得最佳的整体表现。我们发现给定数据集的随机种子间的标准偏差是一致的，我们在第一列中报告了这一点。

	Weight Type	$r=1$	$r=2$	$r=4$	$r=8$	$r=64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

表6：在WikiSQL和MultiNLI上使用不同等级r的验证准确性。令我们惊讶的是，等级为一就足以适应这些数据集上的 W_q 和 W_v ，而单独训练 W_q 则需要更高的 r 。我们在第H.2节中对GPT-2进行了类似的实验。

主要发现：

- LoRA 即使在非常小的秩 r 下也展现出了竞争力的性能。
- 在同时适配 W_q 和 W_v 时比仅适配 W_q 表现更好。
- 增加 r 并没有覆盖更多有意义的子空间，说明低秩适配矩阵已足够。

4.6 LoRA可以应用到模型中的哪些层?

LoRA（Low-Rank Adaptation）可以插入到模型的多个地方，具体取决于需要微调的模型部分和任务要求。以下是一些常见的插入位置及其原因：

4.6.1 线性层（全连接层）

位置：

在深度神经网络的全连接层（Linear Layer）中，通常会使用线性变换 $Wx+b$ 。

原因：

- **主要参数集中**：全连接层通常包含大量参数，通过在这些层中应用LoRA，可以显著减少需要微调的参数数量。
- **计算密集型**：全连接层的计算量较大，通过低秩近似可以有效降低计算复杂度。

4.6.2 注意力层

位置：

在Transformer模型的多头自注意力（Multi-head Self-Attention）机制中，包括查询（Query）、键（Key）和值（Value）矩阵的线性投影部分。

原因：

- **关键功能组件**：自注意力机制是Transformer模型的核心组件，对模型性能影响重大。对这些矩阵进行低秩近似可以显著影响模型的表达能力。
- **参数量大**：这些投影矩阵包含大量参数，使用LoRA可以减少参数数量，降低计算和存储需求。

4.6.3 嵌入层

位置：

在NLP任务中，嵌入层用于将离散的词汇表映射到连续的向量空间。

原因：

- **高维稀疏表示**：嵌入层通常包含大量高维向量，通过LoRA可以有效降低维度，减少计算量和内存占用。
- **提升训练效率**：低秩分解可以使嵌入层的训练更加高效。

5. 案例实战

```
# 本机实验环境
1. ubuntu20.04
2. Python 3.10.14
3. pytorch 1.13.0
4. CUDA
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

01_LoRA_案例实战.ipynb

02_LoRA_微软开源项目

第1步：环境配置

```
conda create -n lora python=3.10
# conda init bash && source /root/.bashrc
conda activate lora
# conda install ipykernel
# ipython kernel install --user --name=lora   # 设置kernel，--user表示当前用户，lora为虚拟环境名称
```