Parameter-Efficient Fine-Tuning (PEFT): Adapters, LoRA, and QLoRA

Fine-tuning Large Language Models with Minimal Resources

Introduction to Fine-Tuning and PEFT

Fine-tuning a pre-trained language model adjusts its weights for a specific task. However, updating billions of parameters is costly in memory and computation. **Parameter-Efficient Fine-Tuning (PEFT)** addresses this by adding or modifying a small subset of parameters while keeping the majority of the pre-trained model frozen.

Why it matters:

- **Efficiency**: Requires less GPU memory and compute.
- **Speed**: Faster to train and iterate.
- **Modularity**: Enables multiple task-specific adapters without duplicating the full model.

Common PEFT methods include **Adapters**, **LoRA (Low-Rank Adaptation)**, and **QLoRA**.

1. Adapters

Concept

Adapters insert small neural network modules (bottleneck layers) between existing layers of a frozen model. During fine-tuning, only adapter weights are updated, leaving the main model parameters unchanged.

Adapter structure:

1. **Down projection**: Reduce dimensionality (d → r).
2. **Non-linearity**: Apply activation (e.g., ReLU).
3. **Up projection**: Restore dimension (r → d).

Equation:

$$extAdapter(h) = W_{extup}(extReLU(W_{extdown}h + b_{extdown})) + b_{extup}$$

- **h**: input representation (dimension d)
- **r**: bottleneck dimension (r ≪ d)

Short Example (Pseudo-Code)

```
# h: input hidden state of size d
# r: bottleneck size
down = Linear(d, r)(h)        # project down
activated = ReLU(down)       # non-linearity
```

```
up = Linear(r, d)(activated)  # project up
output = h + up               # residual connection
```

Reflection

- **Key Takeaway**: Adapters let you fine-tune by adding ~0.1–1% extra parameters.
- **Pitfall**: Choosing bottleneck size r involves a trade-off between capacity and efficiency.
- **Applications**: Task-specific adapters for translation, summarization, and more.

2. LoRA (Low-Rank Adaptation)

Concept

LoRA updates pre-trained weight matrices by learning low-rank decomposition matrices. Instead of updating W (d×d), it learns two smaller matrices A (d×r) and B (r×d) such that:

$$W' = W + BA$$

Only A and B are trained, reducing trainable parameters by a factor of d/r.

Short Example (Pseudo-Code)

```
# Original weight W: d×d
# LoRA matrices A: d×r, B: r×d

delta = B @ A               # low-rank update
output = (W + delta) @ h    # apply updated weights
```

Reflection

- **Key Takeaway**: LoRA fine-tunes with minimal parameters (O(2dr)).
- **Pitfall**: The choice of rank r affects expressiveness vs. efficiency.
- **Applications**: Fine-tuning large models like GPT and BERT for various tasks.

3. QLoRA (Quantized LoRA)

Concept

QLoRA combines LoRA with model quantization (e.g., 4-bit) to further reduce memory usage. The base model weights are quantized, and only the LoRA matrices (in low precision or full precision) are updated.

Workflow:

1. **Quantize** main model weights to 4-bit.
2. **Freeze** quantized weights.
3. **Train** LoRA adapters with minimal overhead.

Benefits and Trade-offs

- **Benefits**: Dramatic reduction in GPU memory, enabling fine-tuning on commodity hardware.
- **Trade-offs**: Slightly lower numerical precision; careful hyperparameter tuning needed.

Overall Reflection and Best Practices

- **Choosing PEFT method**: Adapters for modular multi-task environments; LoRA for straightforward low-rank updates; QLoRA for extreme memory constraints.
- **Hyperparameter tuning**: Bottleneck size (r), quantization bits, learning rates for added modules.
- **Avoiding pitfalls**: Monitor task performance vs. parameter budget; test different ranks and precision levels.

Real-World Applications:

- **Custom chatbots**: Fine-tune GPT variants with LoRA on domain-specific data.
- **Mobile deployment**: Use QLoRA to adapt large models on-device.
- **Research**: Quickly test novel architectures by swapping adapters.

This guide provides a clear, beginner-friendly overview of parameter-efficient fine-tuning methods. Download the PDF for a fully formatted chapter ready to publish.