

























Day 26: LoRA vs PEFT

Feature	PEFT (General)  / 	LoRA (Specific PEFT)  / 
Definition	 Broad category of parameter-efficient fine-tuning methods	 Not a broad method, but a specific PEFT approach
Trainable Parameters	 Small fraction of the model (varies by method)	 Even fewer parameters due to low-rank matrices
Efficiency	 More efficient than full fine-tuning, but method-dependent	 One of the most memory-efficient PEFT techniques
Base Model Changes	 Some PEFT methods modify parts of the base model	 LoRA never modifies the base model (only adds trainable matrices)
Implementation Complexity	 Some PEFT methods are complex to implement	 LoRA is relatively easy to integrate into Transformers
Storage Cost	 Lower than full fine-tuning	 Extremely low, as it only stores small adapter matrices
Performance	 Can reach near full fine-tuning performance	 Comparable to full fine-tuning with much lower compute
Multi-Task Adaptability	 Some PEFT methods require separate models per task	 LoRA allows adapter swapping without retraining
Best Use Cases	 General-purpose fine-tuning	 Ideal for Transformer-based models (e.g., LLaMA, GPT)
Fine-Tuning Speed	 Some PEFT methods still require significant compute	 LoRA is one of the fastest fine-tuning methods

With the rise of large language models (LLMs), fine-tuning them efficiently has become a major challenge. Traditional fine-tuning requires updating all model parameters, which is expensive and memory-intensive. This is where Parameter-Efficient Fine-Tuning (PEFT) techniques come in, with LoRA being one of the most popular methods.

Let's break down their differences in concept, efficiency, use cases, and performance.

What is PEFT?

- PEFT (Parameter-Efficient Fine-Tuning) refers to a category of methods that adapt a pre-trained model without updating all parameters.
- Instead of modifying the entire model, PEFT techniques introduce lightweight trainable components that make fine-tuning more efficient in terms of memory, compute, and storage.

Popular PEFT Techniques

- LoRA (Low-Rank Adaptation)
- Prefix Tuning
- Adapter Layers
- Prompt Tuning

Key Features of PEFT

- Reduces memory usage by keeping most model weights frozen
- Faster training with fewer trainable parameters
- Maintains performance close to full fine-tuning

What is LoRA

- LoRA (Low-Rank Adaptation) is a specific PEFT method that decomposes weight updates into low-rank matrices, reducing the number of trainable parameters.

























How LoRA Works

- Instead of modifying the original weights, LoRA injects low-rank trainable matrices into the pre-trained model's layers (typically in attention layers of Transformers).
- These matrices capture task-specific knowledge while keeping the base model unchanged, allowing for efficient adaptation.

Key Features of LoRA

- Uses low-rank matrices to adapt pre-trained weights
- Reduces GPU memory requirements significantly
- Performs well across multiple tasks with minimal fine-tuning

PEFT vs LoRA – Key Differences

Feature	PEFT (General)  / 	LoRA (Specific PEFT)  / 
Definition	 Broad category of parameter-efficient fine-tuning methods	 Not a broad method, but a specific PEFT approach
Trainable Parameters	 Small fraction of the model (varies by method)	 Even fewer parameters due to low-rank matrices
Efficiency	 More efficient than full fine-tuning, but method-dependent	 One of the most memory-efficient PEFT techniques
Base Model Changes	 Some PEFT methods modify parts of the base model	 LoRA never modifies the base model (only adds trainable matrices)
Implementation Complexity	 Some PEFT methods are complex to implement	 LoRA is relatively easy to integrate into Transformers
Storage Cost	 Lower than full fine-tuning	 Extremely low, as it only stores small adapter matrices
Performance	 Can reach near full fine-tuning performance	 Comparable to full fine-tuning with much lower compute
Multi-Task Adaptability	 Some PEFT methods require separate models per task	 LoRA allows adapter swapping without retraining
Best Use Cases	 General-purpose fine-tuning	 Ideal for Transformer-based models (e.g., LLaMA, GPT)
Fine-Tuning Speed	 Some PEFT methods still require significant compute	 LoRA is one of the fastest fine-tuning methods

When to Use PEFT vs LoRA?

Use PEFT when

- You need a general framework for efficient fine-tuning
- You want flexibility in choosing methods (prompt tuning, adapters, LoRA, etc.)
- Your model has limited compute or memory for training

Use LoRA when

- You are fine-tuning Transformer-based models efficiently
- You need to fine-tune multiple tasks efficiently while keeping the base model unchanged
- You want to deploy fine-tuned adapters without storing multiple full model copies

Stay Tuned for **Day 27** of

Mastering LLMs