LLM Fine-tuning

Om Sai Krishna Madhav Lella

April 23, 2025

Overview

1. Introduction to LLMs
2. Types of LLMs by Architecture
3. Fine-tuning Encoder-based Models
4. Fine-tuning Decoder-based Models
5. LoRA Fine-tuning (for Decoder Models)
6. RoBERTa vs Mistral-7B (Encoder vs Decoder)

Introduction to LLMs

→ **Large Language Models (LLMs)** are deep neural networks trained on massive text corpora.

→ Built using the Transformer architecture (Vaswani et al., 2017).

→ Capable of generalizing across many NLP tasks with minimal task-specific data.

→ Common tasks include:

- Text classification, summarization
- Question answering, translation
- Code generation, chat interfaces

→ Notable LLMs: GPT-4, BERT, T5, LLaMA, Claude

Types of LLMs by Architecture

→ **Encoder-only (e.g., BERT)**:

- Bidirectional context
- Suited for understanding tasks (e.g., classification, NER)

→ **Decoder-only (e.g., GPT)**:

- Autoregressive generation (left-to-right)
- Suited for text generation tasks

→ **Encoder-Decoder (e.g., T5, BART)**:

- Encoder encodes input; decoder generates output
- Best for sequence-to-sequence tasks (e.g., translation, summarization)

Fine-tuning Encoder-based Models

- → Start with a pre-trained encoder (e.g., BERT).
- → Add a task-specific output layer (e.g., classification head).
- → Use supervised learning on labeled examples.
- → Loss function: typically cross-entropy for classification tasks.
- → Advantages:
  - Efficient to fine-tune on small datasets
  - Useful embeddings for downstream tasks

Use Cases: Sentiment analysis, NER, document classification

Fine-tuning Decoder-based Models

- → Fine-tune a decoder-only model (e.g., GPT) using domain-specific prompts and completions.
- → Training objective: minimize loss in predicting the next token (causal language modeling).
- → Requires more data and compute than encoder models.
- → Fine-tuning methods:
  - **Instruction tuning**: train on a variety of task-format prompts.
  - **Reinforcement Learning from Human Feedback (RLHF)**: align outputs with human preferences.
- → **Use Cases:** Dialogue systems, code assistants, creative writing

LoRA (Low-Rank Adaptation) Fine-tuning (for Decoder Models)

What is LoRA?

- → **Parameter-efficient fine-tuning** technique.
- → Instead of updating all model weights, LoRA inserts **trainable low-rank matrices** into each layer.
- → Original weights are **frozen**; only the new matrices are trained.

Key Benefits:

- → Drastically reduces number of trainable parameters.
- → Lower memory and compute requirements.
- → Works well even with **large models** like Mistral or LLaMA.

When to Use:

- → You want to fine-tune **very large LLMs** on domain-specific data.
- → Limited compute or storage budget.
- → Multi-task fine-tuning with shared base model.

RoBERTa vs Mistral-7B (Encoder vs Decoder)

Aspect	**RoBERTa**	**Mistral-7B**
Inference Time	Fast (sub-second)	Slower (few seconds per input)
Model Size	125M–355M parameters	7B parameters
Hardware Needs	Runs on CPU / low-end GPU	Needs high-end GPU (16GB VRAM)
Fine-tuning Time	Few minutes on small sets	Could be hours to days

RoBERTa vs Mistral-7B (Encoder vs Decoder)

Aspect	**RoBERTa**	**Mistral-7B**
Output Type	Class probabilities (softmax)	Free-text response (tokens)
Task Alignment	Optimized for classification, regression	Versatile: QA, classification, text generation
Instruction Following	Not inherently designed for it	Strong instruction adherence
Recommendation	Use when speed + structure matter	Use when flexibility + context matter

Thank You!