

15 LLM Fine-Tuning Techniques

WITH THE PAPER BEHIND EACH ONE

What each technique changes, when to reach for it, and the original paper, one click away.

What do you teach?

The training signal

EXAMPLES OF GOOD ANSWERS

SFT

WHICH ANSWER IS BETTER

RLHF · PPO · RLAIIF · DPO · ORPO

+

Which weights change?

The cost

ALL OF THEM

FFT

A SMALL ADD-ON

PEFT · LoRA · QLoRA · DoRA · Adapters

ONLY SOFT PROMPTS

Prompt · Prefix · P-Tuning

They aren't 15 alternatives. **Pick one from each side.**

SFT + LoRA · DPO + QLoRA · ORPO + LoRA · SFT + FFT

Two questions, not fifteen.

CHEAT SHEET

#	TECHNIQUE	ANSWERS	IN ONE LINE	PAPER
01	SFT	TEACH	Train on example prompts paired with ideal answers.	2021
02	RLHF	TEACH	Learn a reward model from human rankings, then optimise the LLM against it.	2022
03	PPO	TEACH	The RL algorithm classic RLHF uses to update the model.	2017
04	RLAIF	TEACH	RLHF where an AI model, not people, labels the preferences.	2023
05	DPO	TEACH	Learn from chosen/rejected pairs directly. No reward model, no RL.	2023
06	ORPO	TEACH	SFT and preference alignment in one run, no reference model.	2024
07	FFT	WEIGHTS	Update every weight in the model.	2018
08	PEFT	WEIGHTS	The family: train a small set of parameters, freeze the rest.	2023
09	LoRA	WEIGHTS	Freeze the model; train a small low-rank update beside chosen layers.	2021
10	QLoRA	WEIGHTS	LoRA on top of a 4-bit quantised, frozen model.	2023
11	DoRA	WEIGHTS	Split weights into magnitude and direction; LoRA on the direction.	2024
12	Adapters	WEIGHTS	Insert small trainable layers inside each transformer block.	2019
13	Prompt Tuning	SOFT PROMPT	Learn a few "virtual token" embeddings added to every input.	2021
14	Prefix Tuning	SOFT PROMPT	Learn trainable vectors prepended at every layer.	2021
15	P-Tuning	SOFT PROMPT	Learn continuous prompt embeddings through a small prompt encoder.	2021

TEACH techniques decide what the model learns. **WEIGHTS** and **SOFT PROMPT** techniques decide what it costs. Most real projects combine one of each, so "LoRA or DPO?" is the wrong question.

FIG. 02 • BEFORE YOU TRAIN ANYTHING

Is fine-tuning even the fix?

ASK IN THIS ORDER

- 01

Is it missing or changing knowledge?
Prices, policies, your own documents, anything that updates.

→ USE RAG. STOP.
- 02

Would a better prompt or a few examples fix it?
Always try this first. It costs nothing to undo.

→ PROMPTING. STOP.
- 03

Can you write down what good answers look like?
A consistent format, tone, or task the model keeps getting wrong.

→ SFT
- 04

Is "good" easier to judge than to write?
You can say which of two answers is better: helpfulness, safety, style.

→ DPO • ORPO • RLHF
- 05

Now pick the cost.
Start cheap and only move right if quality demands it.

QLoRA → LoRA → FFT

FIG. 03 • SENSIBLE DEFAULTS

IF YOU NEED TO...	START WITH
Follow a format, tone or task reliably	SFT + LoRA
Do the same with only one modest GPU	SFT + QLoRA
Prefer better answers once SFT is done	DPO + LoRA
Do SFT and alignment in a single run	ORPO + LoRA
Get preference labels without people	RLAIF labels, then DPO or PPO
Serve many tasks from one base model	One LoRA or adapter per task
Move to a new domain with lots of data and budget	SFT + FFT

01 SFT Supervised Fine-Tuning

TEACH

WHAT IT DOES

Trains on pairs of a prompt and its ideal answer, with the normal next-token loss. Instruction tuning is SFT across many tasks phrased as instructions.

USE IT WHEN

You can write down what a good answer looks like. Almost always the first step.

Wei et al., 2021 • Finetuned Language Models Are Zero-Shot Learners (FLAN) • arxiv.org/abs/2109.01652

02 RLHF Reinforcement Learning from Human Feedback

TEACH

WHAT IT DOES

People rank pairs of answers, a reward model learns those rankings, and the LLM is optimised with RL (usually PPO) to score well, with a penalty for drifting too far from where it started.

USE IT WHEN

"Good" is easier to judge than to write. The most powerful option, and the most complex to run.

Ouyang et al., 2022 • Training language models to follow instructions with human feedback • arxiv.org/abs/2203.02155
Origin: Christiano et al., 2017 • Deep reinforcement learning from human preferences • arxiv.org/abs/1706.03741

03 PPO Proximal Policy Optimization

TEACH

WHAT IT DOES

The RL algorithm inside classic RLHF. It nudges the model towards higher reward, but clips each update so the model only changes a little per step.

USE IT WHEN

You run RLHF or RLAIIF with a reward model. If all you have is preference pairs, DPO is simpler.

Schulman et al., 2017 • Proximal Policy Optimization Algorithms • arxiv.org/abs/1707.06347

04 RLAIIF Reinforcement Learning from AI Feedback

TEACH

WHAT IT DOES

The RLHF pipeline, but an off-the-shelf LLM decides which answer is better instead of people.

USE IT WHEN

Human labelling is too slow or costly. Check the AI judge against a small human-labelled sample first.

"Comparable performance to RLHF"

Lee et al., 2023 • RLAIIF vs. RLHF: Scaling RL from Human Feedback with AI Feedback • arxiv.org/abs/2309.00267
Origin: Bai et al., 2022 • Constitutional AI: Harmlessness from AI Feedback • arxiv.org/abs/2212.08073

FIG. 04 • WHAT YOU TEACH, CONTINUED

05 DPO Direct Preference Optimization

TEACH

WHAT IT DOES

Learns straight from preference pairs (chosen vs rejected) with a simple classification-style loss. No separate reward model and no RL loop; a frozen reference copy of the model keeps it from drifting.

USE IT WHEN

You have better/worse answer pairs and want alignment without RL infrastructure. The usual step after SFT.

Rafailov et al., 2023 • Direct Preference Optimization: Your Language Model is Secretly a Reward Model • arxiv.org/abs/2305.18290

06 ORPO Odds Ratio Preference Optimization

TEACH

WHAT IT DOES

Adds a preference penalty, based on the odds of the chosen vs the rejected answer, to the normal SFT loss, so one training run does both. No reference model.

USE IT WHEN

You want SFT and preference alignment in a single, cheaper stage.

Hong et al., 2024 • ORPO: Monolithic Preference Optimization without Reference Model • arxiv.org/abs/2403.07691

FIG. 05 • WHICH WEIGHTS CHANGE

07 FFT Full Fine-Tuning

WEIGHTS

WHAT IT DOES

Updates every weight. Needs memory for the weights, their gradients and the optimiser state, several times the size of the model itself, and gives you a full new copy per task.

USE IT WHEN

A big shift to a new domain or language, lots of data, and the budget to match.

Howard & Ruder, 2018 • Universal Language Model Fine-tuning for Text Classification (ULMFiT) • arxiv.org/abs/1801.06146

08 PEFT Parameter-Efficient Fine-Tuning

WEIGHTS

WHAT IT DOES

Not one method but the family: freeze most of the model and train a small set of parameters. LoRA, QLoRA, DoRA, adapters and the three prompt methods are all PEFT. Also the name of Hugging Face's library for them.

USE IT WHEN

You need fine-tuning on limited hardware, or many task versions of one base model.

Lialin et al., 2023 • Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning • arxiv.org/abs/2303.15647

09 LoRA Low-Rank Adaptation

WEIGHTS

WHAT IT DOES

Freezes the pre-trained weights and learns a small low-rank update (two thin matrices) beside chosen layers. The update can be merged back in, so serving costs nothing extra.

USE IT WHEN

The default way to fine-tune, for SFT, DPO or any other objective, on a limited GPU budget.

On GPT-3 175B: 10,000× fewer trainable parameters, 3× less GPU memory

Hu et al., 2021 • LoRA: Low-Rank Adaptation of Large Language Models • arxiv.org/abs/2106.09685

10 QLoRA Quantized LoRA

WEIGHTS

WHAT IT DOES

Loads the frozen base model in 4-bit ("4-bit NormalFloat", NF4) and trains LoRA adapters on top of it.

USE IT WHEN

The model barely fits, or doesn't fit, in your GPU memory.

A 65B model fine-tuned on a single 48 GB GPU

Dettmers et al., 2023 • QLoRA: Efficient Finetuning of Quantized LLMs • arxiv.org/abs/2305.14314

11 DoRA Weight-Decomposed Low-Rank Adaptation

WEIGHTS

WHAT IT DOES

Splits each pre-trained weight into a magnitude and a direction. The magnitude is trained directly; the direction is updated with LoRA.

USE IT WHEN

Plain LoRA falls short of full fine-tuning on your task. It's a drop-in option in common libraries such as Hugging Face PEFT.

Liu et al., 2024 • DoRA: Weight-Decomposed Low-Rank Adaptation • arxiv.org/abs/2402.09353

12 Adapters Bottleneck adapter layers

WEIGHTS

WHAT IT DOES

Inserts small bottleneck layers inside each transformer block and trains only those. The original weights stay frozen.

USE IT WHEN

Many tasks share one base model, each as a small swappable module.

Within 0.4% of full fine-tuning on GLUE, adding 3.6% parameters per task

Houlsby et al., 2019 • Parameter-Efficient Transfer Learning for NLP • arxiv.org/abs/1902.00751

13 Prompt Tuning

SOFT PROMPT

WHAT IT DOES

Freezes the whole model and learns a handful of "virtual token" embeddings added to the start of every input.

"Closes the gap" with full fine-tuning once models exceed billions of parameters

USE IT WHEN

Very large models, simpler tasks, and each task stored as a tiny file.

Lester et al., 2021 • The Power of Scale for Parameter-Efficient Prompt Tuning • arxiv.org/abs/2104.08691

14 Prefix Tuning

SOFT PROMPT

WHAT IT DOES

Learns trainable vectors that are prepended at every layer, not just at the input, while the model stays frozen.

Learns only 0.1% of the parameters

USE IT WHEN

Generation tasks on a frozen model, with very little stored per task.

Li & Liang, 2021 • Prefix-Tuning: Optimizing Continuous Prompts for Generation • arxiv.org/abs/2101.00190

15 P-Tuning

SOFT PROMPT

WHAT IT DOES

Learns continuous prompt embeddings, produced by a small prompt encoder, that are placed into the input. The model's own weights are untouched.

USE IT WHEN

Improving a frozen model on understanding tasks without changing its weights.

Liu et al., 2021 • GPT Understands, Too • arxiv.org/abs/2103.10385

Short on time? Read three papers: **LoRA** (how cheap fine-tuning works), **InstructGPT** (how SFT and RLHF fit together) and **DPO** (alignment without RL). Every link in this PDF is clickable.

Found this useful?

Follow SKD Neuron for more explainers on RAG, AI agents and AI system design. Save this for your next interview.

@SKDNEURON
SKDNEURON.COM