

Fine-tuning and PEFT

The adaptation spectrum, LoRA, and aligning a model to people

Ramaseshan Ramachandran

Fine-Tuning and PEFT

THE ADAPTATION SPECTRUM

How do we make a general pre-trained model good at *our* task?

Method	Weights changed	When to use
Prompting / few-shot	none	task is common, examples fit in context
RAG	none	knowledge is external and changes often
PEFT (LoRA, ...)	< 1%	new <i>behavior/style/domain</i> , modest compute
Full fine-tuning	all	large data, maximal quality, big budget
Pre-training	all, from scratch	you are an AI lab

Rule of thumb: RAG adds **knowledge**; fine-tuning adds **behavior**. They are complementary, not competitors.

FINE-TUNING: THE OBJECTIVE

Take the pre-trained parameters θ_0 and continue training on task data $\{(x^{(i)}, y^{(i)})\}$ with the same cross-entropy loss:

$$\mathcal{L}(\theta) = - \sum_i \sum_t \log p_{\theta} \left(y_t^{(i)} \mid x^{(i)}, y_{<t}^{(i)} \right), \quad \theta \text{ initialized at } \theta_0 \quad (1)$$

- ▶ **Domain adaptation**: continue next-token prediction on in-domain text (legal, medical, code)
- ▶ **Instruction / supervised fine-tuning (SFT)**: (instruction, response) pairs. This is the first step of alignment we saw earlier
- ▶ **Task fine-tuning**: classification, extraction, translation pairs
- ▶ Use a **small learning rate** (10^{-5} vs. 10^{-3} from scratch): we want to *nudge* θ_0 , not erase it

WHY FULL FINE-TUNING DOES NOT SCALE

For a 7B-parameter model in 16-bit precision:

Weights	14 GB
Gradients	14 GB
Adam optimizer states ($2\times$, in fp32)	56 GB
Total (before activations!)	~84 GB

- ▶ Beyond any single consumer GPU (24 GB)
- ▶ One full copy of the model *per task*. Ten tasks = 140 GB of checkpoints
- ▶ **Catastrophic forgetting**: aggressive updates overwrite general abilities learned in pre-training

Can we adapt the model by training only a *tiny* number of parameters?

LOW-RANK ADAPTATION (LORA)¹

Hypothesis: the *change* needed to adapt a weight matrix has low intrinsic rank. It is the same low-rank intuition as SVD/LSA earlier in the course.

$$h = W_0x + \Delta Wx = W_0x + \underbrace{B}_{d \times r} \underbrace{A}_{r \times k} x, \quad r \ll \min(d, k) \quad (2)$$

- ▶ **Freeze** the pre-trained W_0 ; train only A and B
- ▶ A initialized with small random values, B initialized to zero $\Rightarrow \Delta W = BA = 0$ at the start: training begins exactly at the pre-trained model
- ▶ Typically applied to the attention projection matrices ($W_Q, W_V, \dots$)
- ▶ At deployment, merge $W = W_0 + BA$ for **zero extra inference cost**

¹HU ET AL., “LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS”, ICLR 2022.

LORA: HOW MUCH DO WE SAVE?

One attention projection in Llama-2-7B: $d = k = 4096$.

Full ΔW : $4096 \times 4096 \approx 16.8\text{M}$ params

LoRA ($r = 8$): $4096 \times 8 + 8 \times 4096 \approx 65\text{K}$ params ($\approx 0.4\%$)

- ▶ Across the whole model, LoRA typically trains 0.1–1% of the parameters
- ▶ Optimizer states shrink proportionally. A 7B model becomes trainable on a single 24 GB GPU
- ▶ Each task's adapter is a few **megabytes**: keep one base model, swap adapters per task/customer
- ▶ Freezing W_0 also mitigates catastrophic forgetting

THE PEFT FAMILY

- ▶ **LoRA** is a low-rank update to frozen weights (the current default)
- ▶ **QLoRA** quantizes the frozen base model to 4-bit and trains LoRA adapters in 16-bit on top: fine-tune a 65B model on one 48 GB GPU²
- ▶ **Adapters** are small bottleneck layers inserted between transformer sublayers (they add inference latency, unlike LoRA)
- ▶ **Prompt / prefix tuning** learns continuous “virtual token” embeddings prepended to the input, leaving weights untouched
- ▶ **BitFit** trains only the bias terms

All share one design: **freeze the knowledge, train a small steering wheel.**

²Dettmers et al., “QLoRA: Efficient Finetuning of Quantized LLMs”, NeurIPS 2023.

A PRACTICAL FINE-TUNING RECIPE

1. **Try prompting first.** Fine-tune only when few-shot prompting demonstrably falls short
2. **Data before compute:** 1,000 clean, diverse, deduplicated examples beat 100,000 noisy ones
3. Hold out a test set *before* looking at the data (the same discipline as every model in this course)
4. Start with LoRA: $r \in \{8, 16\}$, learning rate $\sim 10^{-4}$, 1–3 epochs
5. Evaluate on the held-out set *and* on a general benchmark. Watch for forgetting and regression
6. Compare against the RAG alternative: if the failure is *missing knowledge*, fine-tuning is the wrong tool

SUMMARY AND EXERCISES

- ▶ Fine-tuning continues pre-training on task data from θ_0 ; full fine-tuning is memory-hungry and forgets
- ▶ LoRA trains a low-rank $\Delta W = BA$ on frozen weights: $< 1\%$ of parameters, mergeable, swappable
- ▶ Choose by failure mode: knowledge $\rightarrow$ RAG; behavior/format/style $\rightarrow$ PEFT

Exercises

1. Compute total LoRA parameters for Llama-2-7B with $r = 16$ applied to W_Q and W_V in all 32 layers. What fraction of 7B is that?
2. Fine-tune a small model (e.g., 1–3B) with LoRA on 500 instruction pairs; report held-out accuracy vs. the zero-shot baseline.
3. Ablate $r \in \{2, 8, 64\}$: plot quality vs. trainable parameters. Where does it saturate, and why is that consistent with the low-rank hypothesis?

Alignment

A CAPABLE MODEL IS NOT YET A USEFUL ONE

Ask a *base* language model a question and it may answer. It may also continue your list of questions, because that is a perfectly good continuation of the text you gave it.

You type	What a base model may produce
What is the capital of France?	What is the capital of Spain? What is the capital of Italy?

Nothing has gone wrong. The model knows the answer, and nobody ever told it that a question should be followed by an answer rather than by another question.

Closing that gap is what alignment means.

THREE STAGES, THREE DIFFERENT THINGS

Stage	Data	Teaches
Pre-training	raw text, trillions of tokens	language and facts
Instruction tuning	instruction/response pairs	the shape of a response
Preference tuning	human rankings of responses	which response is better

- ▶ Pre-training is enormously expensive and happens once. The other two are moderate, and both are repeated.
- ▶ These are **not alternatives**. Each assumes the one before it, and each teaches something the previous stage could not.

INSTRUCTION TUNING

Continue training on pairs of instructions and good responses. **The data is what matters:** a few tens of thousands of well-written pairs turn a text continuer into something that answers questions, follows formats, and stops when it is done.

- ▶ Note what has *not* happened. **No new knowledge entered the model.** The capital of France was already there.
- ▶ What changed is the model's **default behaviour** when it sees a question.
- ▶ This is why the stage is cheap relative to pre-training, and why **data quality dominates data quantity** here.

PREFERENCE TUNING: WHY IT IS NEEDED

Instruction tuning teaches the *shape* of a response. It cannot easily teach which of two well-shaped responses is better.

- ▶ Nobody can write a loss function for helpfulness.
- ▶ Yet people can reliably say **which of two answers they prefer**.
- ▶ That comparison is enough.

RLHF IN THREE STEPS[1]

Reinforcement learning from human feedback turns comparisons into a training signal:

1. Collect pairs of model responses and have people **rank** them.
2. Train a **reward model** to predict those rankings.
3. Optimise the language model against the reward model, with a **penalty for drifting too far** from where it started.

The penalty term is not an afterthought. Without it the model finds outputs the reward model scores highly and humans do not. That is **reward hacking**, and it happens reliably.

DPO: THE SHORTER ROAD[3]

Direct preference optimisation reaches a similar destination with less machinery.

RLHF	DPO
separate reward model	none
reinforcement learning loop	none
optimise against a learned proxy	optimise the preference data directly

DPO is simpler to implement and to debug, and the two are close enough in practice that **the choice is usually made on engineering grounds**.

Constitutional AI [2] replaces much of the human labelling with a written set of principles, a different answer to the same problem.

WHAT ALIGNMENT ACTUALLY CHANGES

The word carries more weight than the technique does, so be precise.

- ▶ Alignment changes **which of the model's existing behaviours are likely**.
- ▶ It neither adds knowledge nor removes capability.

A model that could write malware before alignment can still write malware after it. What changed is that the aligned model's **most probable** response to the request is a refusal.

A behaviour made unlikely is not a behaviour removed. That one distinction explains most of what follows.

WHAT STILL GOES WRONG

Four failure modes survive all three stages, and each has a known cause.

Failure	Cause
Hallucination	trained to produce <i>likely</i> text, and a plausible falsehood is likely text
Sycophancy	preference tuning working correctly on data where raters preferred agreeable answers
Jailbreaks	alignment shifted probabilities rather than removing capabilities, so a new framing shifts back
Inherited bias	the pre-training corpus; alignment reduces expression without removing the source

None of the four is a bug in the sense of a mistake in the code. Each is a predictable consequence of the objective, which is why each needs **a system around the model rather than a fix inside it.**

SUMMARY AND EXERCISES

- ▶ Pre-training teaches language, instruction tuning teaches the shape of a response, preference tuning teaches which response is better.
- ▶ RLHF needs a reward model because helpfulness has no loss function, and a penalty term because reward hacking is reliable.
- ▶ Alignment changes probabilities, not capabilities.

Exercises

1. Write three prompts on which a base model and an instruction-tuned model would behave very differently. For each, say why the base model's output is a *correct* continuation.
2. Explain what the RLHF penalty term protects against, and predict what its absence would produce.
3. Instruction tuning changes behaviour without adding knowledge. Design an experiment that distinguishes the two, using only prompts and no access to the weights.

REFERENCES I

- [1] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL]. URL: <https://arxiv.org/abs/2203.02155>.
- [2] Yuntao Bai et al. *Constitutional AI: Harmlessness from AI Feedback*. 2022. arXiv: 2212.08073.
- [3] Rafael Rafailov et al. “Direct Preference Optimization: Your Language Model is Secretly a Reward Model”. In: *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. 2023.