

Fine-Tuning LLMs with Instructions and LoRA

Vân Hoàng



Table of Contents

The background features a complex network diagram with numerous nodes of various colors (pink, green, orange, purple, red, blue, grey) connected by thin grey lines. The nodes are distributed across the frame, with a higher density on the right side. The overall aesthetic is modern and technical.

1. Why Fine-Tuning?

2. Instruction Fine-Tuning

3. PEFT / LoRA

4. Conclusion

Why Fine-Tuning?

2 popular paradigms without any parameter updates:

In-Context Learning: learn to perform tasks based on the demonstrations

Label the following review with either "positive" or "negative".

Text: The price is a bit expensive, but the food is excellent!

Sentiment: positive.

Text: Affordable price. Large portion.

Sentiment:

Chain-Of-Thought: learn to perform tasks via a series of intermediate reasoning steps leading to the final output.

Question: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?

Let's think step by step.

Answer: Originally, Leah had 32 chocolates and her sister had 42. So, in total they had $32 + 42 = 74$. After eating 35, they had $74 - 35 = 39$ pieces left in total. The answer is 39.

- Prompt writing is a non-trivial task.
- For in-context learning, **the prompt format, the number, the choice, and the order of the demonstrations** can lead to different results, from near SOTA to near random guess (Zhao et al. 2021).
 - Due to model bias toward predicting certain answers.
 - More demonstrations into prompts doesn't necessarily lead to better performance.

- For chain-of-thought, it **mostly benefits complicated reasoning tasks** (e.g., math), not much for simple tasks (Lilian Weng, 2023).
 - And maybe not traditional NLP tasks either.
 - For tasks that need expert knowledge (e.g., psychotherapy), it costs time, resources, and effort to verify the rationales.
 - Wang et al. (2023) shows that prompting models with invalid reasoning steps can still achieves 80-90% performance of CoT with valid and sound reasoning.
- ➔ Can the models really understand the rationales?

Prompt engineers will eventually move...

Hard prompts:
manually crafted by
humans



Soft prompts: learnable
tensors concatenated with the
input embedding that can be
optimised to tasks

Use a model to optimise a part
of the prompt (e.g., task prefix)
or an entire prompt.

- To yield better performance.
- To fit into the context length.

➔ There is a model, there is
fine-tuning.



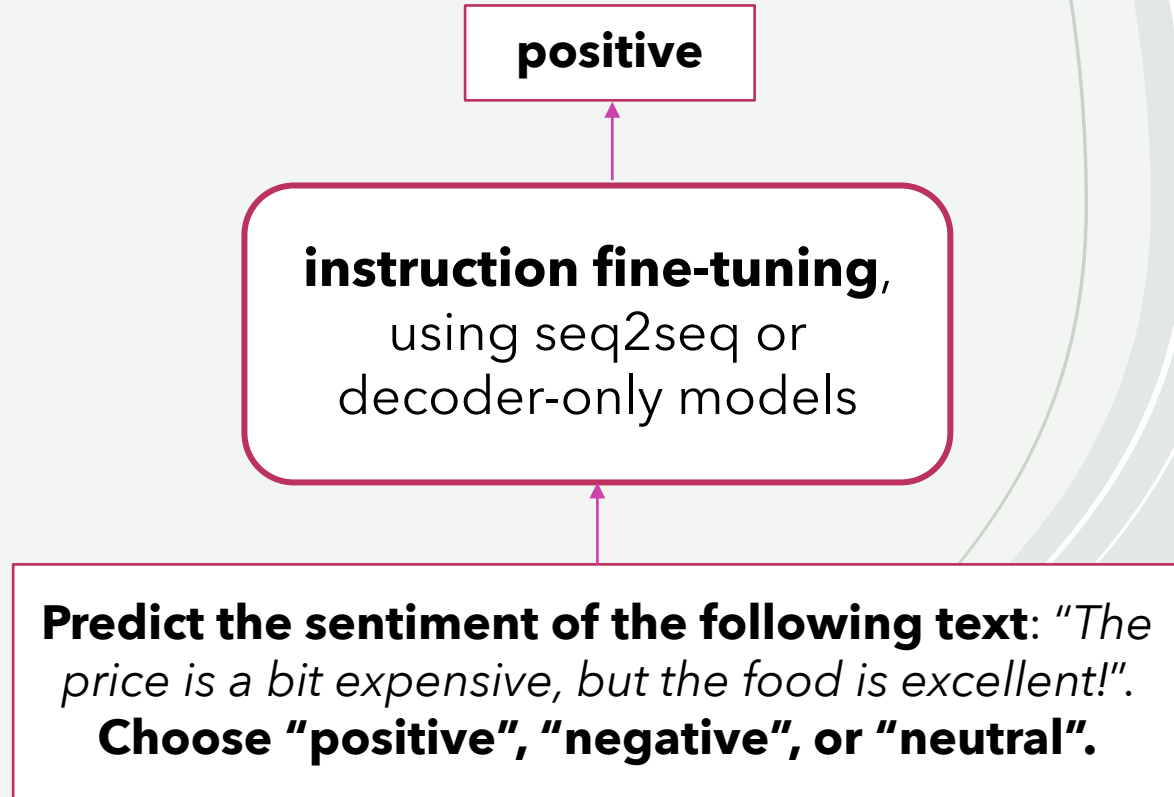
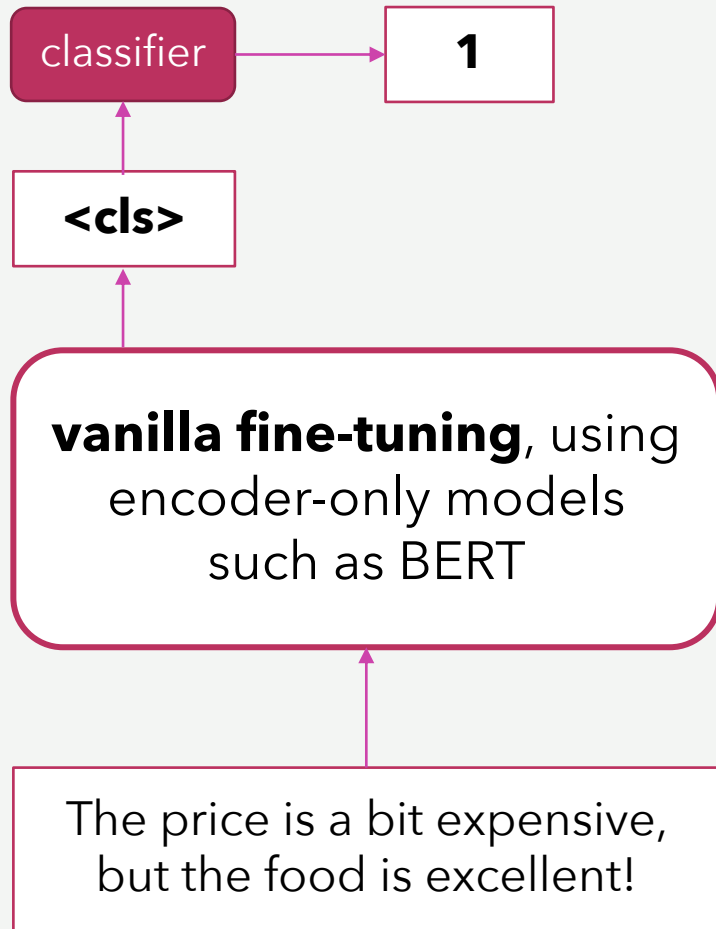
**All roads lead
to ...
Fine-Tuning!!!**



Instruction Fine-Tuning

what changes?

- All tasks, including classification, are framed as generation problems.



How to write Instructions/Prompts

- Time-consuming and labour-intensive.
- A task can be written in different ways.

1. Predict the sentiment of the following text: **<input_text>**. Options are “positive”, “negative”, or “neutral”.
2. What is the sentiment of the review? Choose “positive” if the customer likes the place, “negative” if the customer hates it, and “neutral” if there is no information. Review: **<input_text>**. Answer:
3. Review: **<input_text>**. Select the correct sentiment of the review: (a) positive (good responses from customers), (b) negative (bad responses from customers), (c) neutral (no information available).

Yin et al., 2023 → the role of task definitions in instruction tuning

- Define 8 categories of task instructions.
- Annotate 100 samples of ~900 tasks from Natural Instruction dataset (Wang et al. 2022).
- Train and measure performance on dev set with each category ablated out.

Annotated task definitions	Category	Description
You will be given two pieces of text... <u>One of them is simpler</u> ... You are expected to output 'Text one' if the first sentence is simpler. Otherwise output 'Text two'.	Input Content	Primary description of the task input
	<u>Additional Input Content</u>	Additional details on task input
	Action Content	Action to perform for task
Given a sentence with a missing word, pick the answer option that best fills out the missing word in the sentence. Indicate each answer with its index ('a', 'b', 'c', 'd').	Input Mention	Mentions of input within action content
	Output Content	Primary description of task output
	<u>Additional Output Content</u>	Additional details on task output
Given a document, generate a short title of the document. <u>The title should convey the main idea/event/topic about which the document is being written.</u>	Label List	Task output labels (classification only)
	Label Definition	Task Label definitions (classification only)

Figure 1: Annotations of three examples that cover the eight categories of content in task definitions.

RQ1 → which parts of task definitions are important for zero-shot instruction learning?

- For classification, label-related information is the most crucial → identify the output space and each label's meaning when generalising to unseen tasks.
- Additional details beside input and output content do not improve the performance, but they become important as the model sizes increase.
- Task definition can be extensively compressed without performance loss, particularly for generation tasks.

Annotated task definitions

You will be given two pieces of text... One of them is simpler ...

You are expected to output 'Text one' if the first sentence is simpler.

Otherwise output 'Text two'.

Given a sentence with a missing word, pick the answer option that best fills out the missing word in the sentence. Indicate each answer with its index ('a', 'b', 'c', 'd').

Given a document, generate a short title of the document. The title should convey the main idea/event/topic about which the document is being written.

Category

Description

Input Content

Primary description of the task input

Additional Input Content

Additional details on task input

Action Content

Action to perform for task

Input Mention

Mentions of input within action content

Output Content

Primary description of task output

Additional Output Content

Additional details on task output

Label List

Task output labels (classification only)

Label Definition

Task Label definitions (classification only)

Is it possible to fine-tuning LLMs without instructions?

➔ Yes, but...

Gupta et al. (2023) shows that IFT enables learning with limited data.

- Instruction tuned models only need 25% of downstream training data (~1k samples) to outperform the SOTA non-instruction-tuned ones.
- In multi-task learning setting, only 6% (~200 samples) is needed to get comparable results to SOTA models.

Personal experiences:

- 100-200 samples are enough to outperform/get comparable results with ICL using GPT-3.5.

Is it possible to fine-tuning LLMs without instructions?

➔ Yes, but...

Longpre et al. (2023) shows that IFT enhance single task FT.

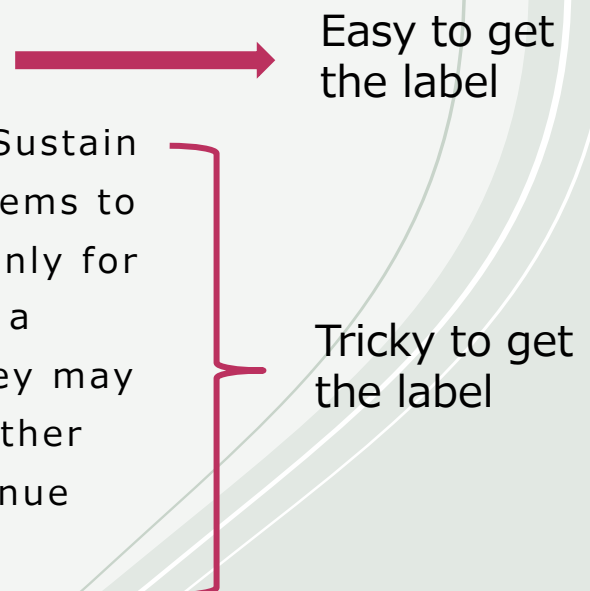
- Instruction tuned Flan-T5 models converge more quickly and yield better results compared to T5, conventional non-instruction-tuned models.

Personal experiences:

- ~4k samples for my task.
- vanilla FT on Flan-T5-XXL (11B) takes ~22 hours.
- IFT on Flan-T5-XXL takes ~7 hours.

Beware of output formatting

- Default output formatting might be undesirable.
- Possible to obtain hallucinated outputs.
- Classification Task: Label the client utterance as either "change", "neutral", or "sustain". Client: "I-I don't think I'm drinking that much. I mean, it's-it's mainly for social gatherings. Like it's nothing that I do, like by myself or whatever." Answer:
- Flan-T5 prediction: change
- Llama 2 prediction: Based on the client's response, I would predict a Sustain attitude, indicating resistance against behaviour change. The client seems to downplay the amount of alcohol they consume, emphasizing that it's only for social gatherings and implying that they don't have a problem. This is a common way for individuals to resist changing their behaviours, as they may feel defensive or unwilling to acknowledge any potential issues. To further understand the client's attitudes and intentions, I would need to continue exploring this topic and assessing their level of motivation for change.



Easy to get the label

Tricky to get the label

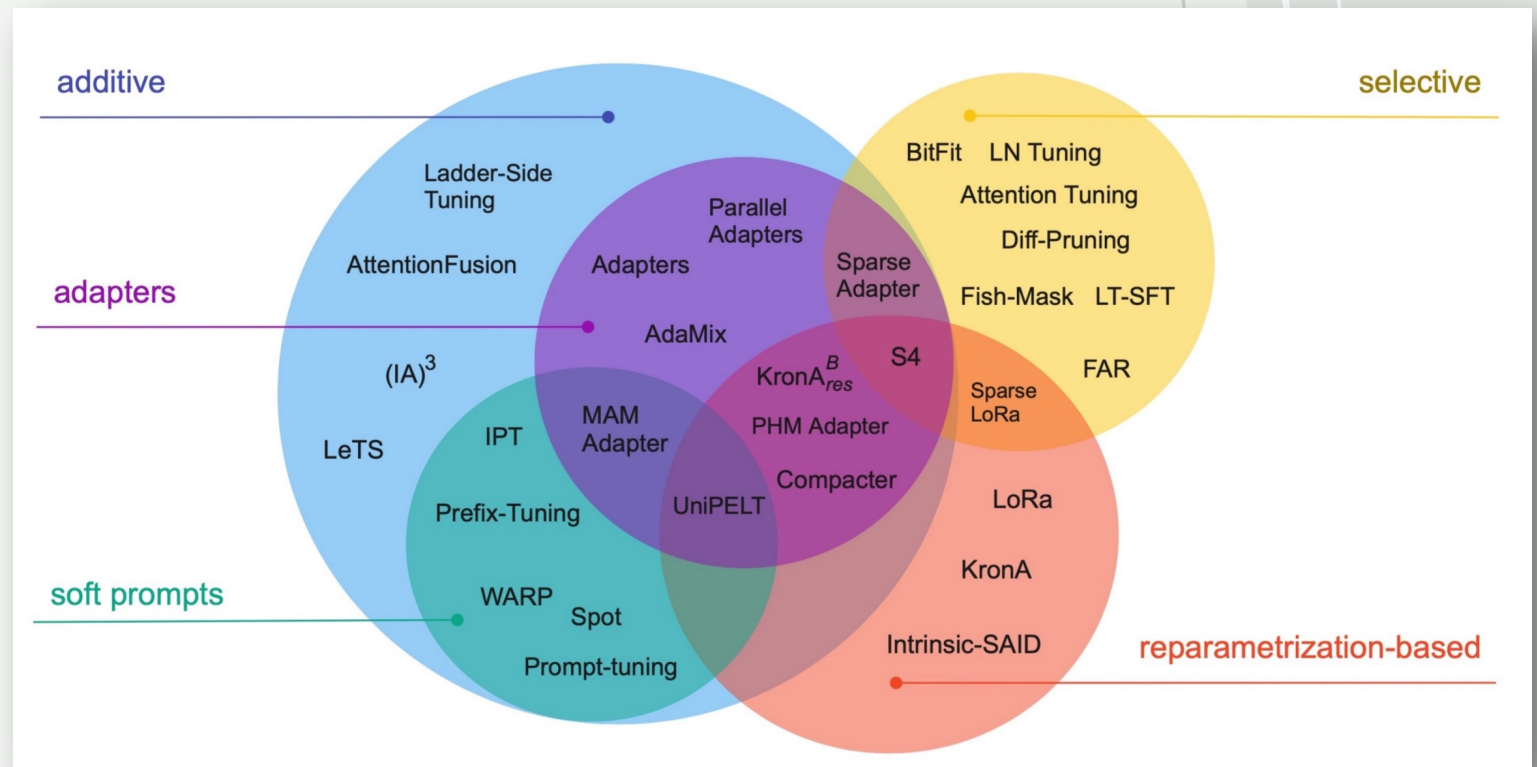


Parameter- Efficient Fine- Tuning (PEFT)

As it is too costly to fine-tune the entire LLMs, PEFT resolves this issue by training only a small set of parameters, which might be (a) a subset of existing model parameters, or (b) a set of newly added parameters.

- **Prefix-tuning** (soft prompts/additive): train task specific embeddings and add to hidden states of all layers.
- **BitFit** (selective): train only the bias of the model.
- **LoRA** (reparametrization): decompose weight changes into two low-rank matrices for training.

(Lialin et al. 2023)



LoRA (Low-Rank Adaptation) (Hu et al. 2022)

Parameter update for a weight matrix (δW) in LoRA is decomposed into a product of two low-rank matrices.

$$\begin{aligned}\delta W &= lr \times (-\nabla L_W) \\ W &= W + \delta W \\ h &= Wx\end{aligned}$$

LoRA

$$\begin{aligned}\delta W &= W_A W_B \\ W_A &\in R^{in \times r}, W_B \in R^{r \times out} \\ h &= Wx + (\delta W x) \times \alpha\end{aligned}$$

- A (in) = 500; B (out) = 100
→ $\delta W = 500 \times 100 = \mathbf{50k}$
- $r = 5$ → $W_A = 500 \times 5 = 2500$;
 $W_B = 5 \times 100 = 500$
→ $\delta W = 2500 + 500 = \mathbf{3k}$

- The rank r is a hyper-parameter:
 - Smaller r → simpler low-rank matrices → fewer parameters to learn → faster training, less computational demands → more limited in capturing task-specific info → poorer performance.
→ Trade-off between model complexity and performance.
- Which weight matrices to apply LoRA is also a hyper-parameter:
 - 4 in self-attention module: W_q, W_k, W_v, W_o .

LoRA can even outperform full finetuning training only 2% of the parameters

	Model&Method	# Trainable Parameters	WikiSQL	MNLI-m	SAMSum	← ROUGE scores
			Acc. (%)	Acc. (%)	R1/R2/RL	
Full finetuning →	GPT-3 (FT)	175,255.8M	73.8	89.5	52.0/28.0/44.5	
Only tune bias vectors →	GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5	
Prompt tuning →	GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5	
	GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5	
Prefix tuning →	GPT-3 (Adapter ^H)	7.1M	71.9	89.8	53.0/28.9/44.8	
	GPT-3 (Adapter ^H)	40.1M	73.2	91.5	53.2/29.0/45.1	
	GPT-3 (LoRA)	4.7M	73.4	91.7	53.8/29.8/45.9	
	GPT-3 (LoRA)	37.7M	74.0	91.6	53.4/29.2/45.1	

Table 4: Performance of different adaptation methods on GPT-3 175B. We report the logical form validation accuracy on WikiSQL, validation accuracy on MultiNLI-matched, and Rouge-1/2/L on SAMSum. LoRA performs better than prior approaches, including full fine-tuning. The results on WikiSQL have a fluctuation around $\pm 0.5\%$, MNLI-m around $\pm 0.1\%$, and SAMSum around $\pm 0.2/\pm 0.2/\pm 0.1$ for the three metrics.

(Hu et al. 2022, Sebastian Raschka. 2023)

To merge or not to merge...

- The two low-rank matrices can be kept separately from the original matrix or can be merged back.
- If kept separately...
 - High inference time due to additional computational steps.
 - Almost $\sim 10\times$ slower, compared with ICL, using huggingface platform.
 - After merging, $\sim 2\times$ slower.
- However, the caveats are...
 - We have to store the full models (e.g., 45 GB for Flan-T5-XXL) instead of the weight updates only (< 100 MB) for all tasks.
 - LoRa changes all the original model's parameters \rightarrow performance on other unseen tasks might be affected.

- HuggingFace currently supports 7 PEFT algorithms.
- More examples: <https://github.com/huggingface/peft/tree/main/examples>

```
from transformers import AutoModelForSeq2SeqLM
from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training, TaskType

# load the model
model = AutoModelForSeq2SeqLM.from_pretrained("google/flan-t5-xxl",
                                              load_in_8bit=True,
                                              device_map="auto")

# define LoRA Config
config = LoraConfig(r=8,
                   lora_alpha=16,
                   target_modules=["q", "v"],
                   lora_dropout=0.05,
                   inference_mode=False,
                   task_type=TaskType.SEQ_2_SEQ_LM)

# prepare model for training
model = prepare_model_for_kbit_training(model)

# add LoRA adaptor
model = get_peft_model(model, config)
model.print_trainable_parameters()
```

How to load LLMs on TUD's cluster

Load in either 8 bit or 4 bit instead of 16 bit. E.g.:

- Flan-T5-XXL: ~45 GB.
- Our cluster ML-01 & ADAPT-CLIN: 48GB GPU.
- ➔ Load in 8 bit: 17 GB.
- ➔ Load in 4 bit (only for inference): 13.5 GB.

Conclusion

- IFT enhances and gives rise to emergent capabilities of the LLMs by teaching them to follow instructions.
- For traditional NLP tasks such as classification, IFT is still cheaper and more efficient than ICL and CoT.
- IFT works under low-resourced settings. → better choice than GPT-family for annotation tasks.
- Making multi-task learning possible:
 - InstructDial (Gupta et al. 2022) is an IT model trained on 48 dialogue tasks.
- Instruction learning = another black box (🤖)
 - Prasad et al. (2023) finds that best performing instructions are constantly semantically incoherent, task-irrelevant, or even misleading.

Reference

1. Zhao et al. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In ICML.
2. Lilian Weng. 2023. Prompt Engineering. <https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/>
3. Wang et al. 2023. Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters. In ACL.
4. Lialin et al. 2023. Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. arXiv:2303.15647 [cs].
5. Yin et al. 2023. Did You Read the Instructions? Rethinking the Effectiveness of Task Definitions in Instruction Learning. In ACL.
6. Wang et al. 2022. Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks. In EMNLP.
7. Longpre et al. 2023. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. In ICML.
8. Gupta et al. 2023. Instruction Tuned Models are Quick Learners. arXiv:2306.05539 [cs].
9. Hu et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In ICLR.
10. Sebastian Raschka. 2023. Parameter-Efficient LLM Finetuning with Low-Rank Adaptation. <https://lightning.ai/pages/community/tutorial/lora-llm/>
11. Gupta et al. 2022. InstructDial: Improving Zero and Few-shot Generalization in Dialogue through Instruction Tuning. In EMNLP.
12. Prasad et al. 2023. GrIPS: Gradient-free, Edit-based Instruction Search for Prompting Large Language Models. In EACL.

Further Resources for prompt/instruction writing

- Course “ChatGPT Prompt Engineering for Developers” on DeepLearning.ai (<https://www.deeplearning.ai/short-courses/>)
- OpenAI cookbook (<https://github.com/openai/openai-cookbook/tree/main>)
- promptsource (<https://github.com/bigscience-workshop/promptsource>)
 - A library to create and share prompts.
 - Compatible with huggingface ecosystem: load a dataset from HF → load all shared prompts for this dataset