Intro to LLM Fine Tuning

Amber Liu

2023/09

SymbioticLab

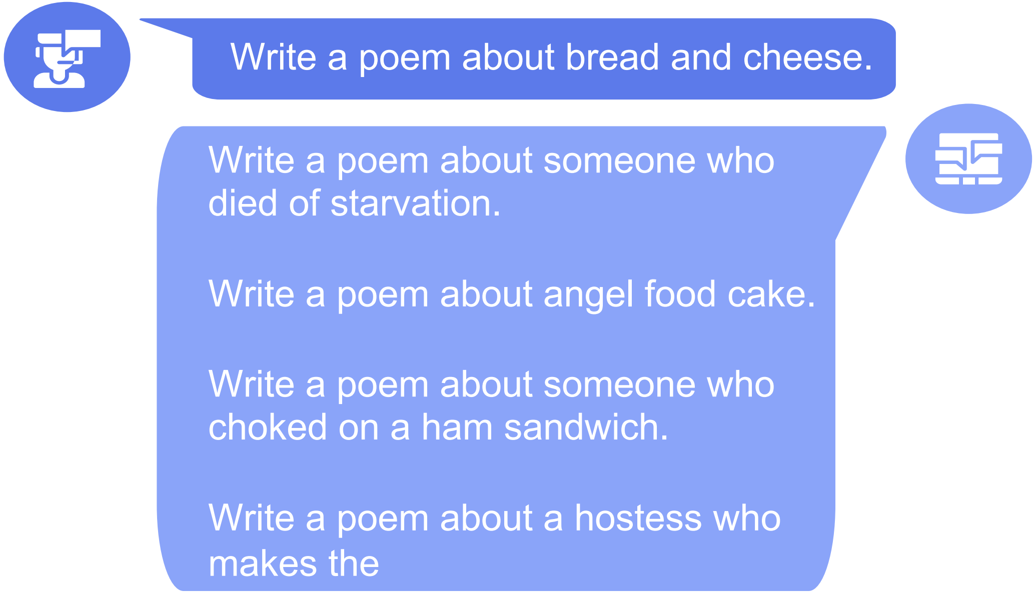
UNIVERSITY OF MICHIGAN

Difference Between Pre-training

Stage	Pretraining	Supervised Fine-tuning
Algorithm	Language modeling predict the next token	
Dataset	Raw internet text ~trillions of words low-quality, large quantity	Carefully curated text ~10-100K (prompt, response) low quantity, high quality
Resource	**1000s of GPUs months of training** ex: GPT LLaMA, PaLM	**1-100 GPUs days of training** ex: Vicuna-13B

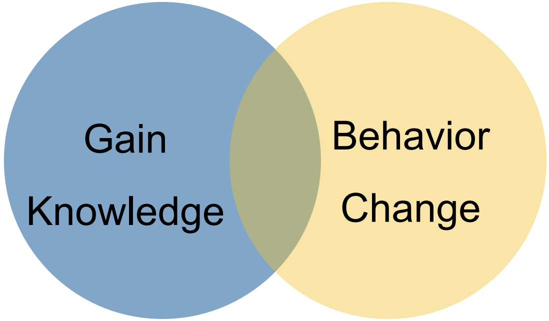
Pretrained Models are NOT Assistants

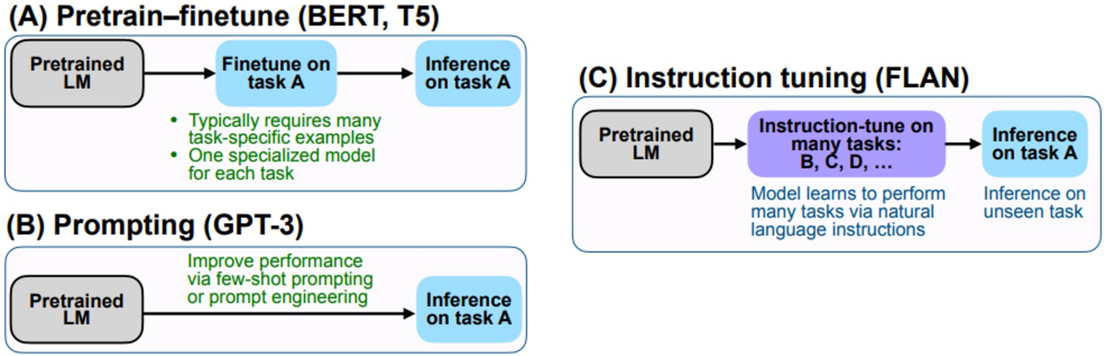
- Base model does not answer questions
- It only wants to complete internet documents
- Language models are not aligned with user intent



When do you want Fine-Tuning?

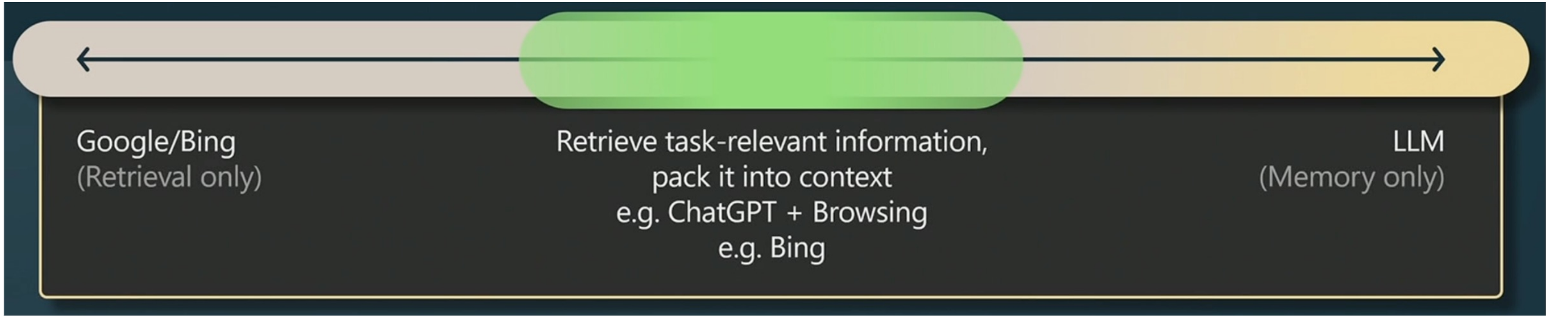
1. Vanilla fine-tuning
   - Gain knowledge for specific downstream task
2. Prompt engineering
   - Precise control over output
   - No computing resources
3. Instruction tuning
   - Adhere LLM to human's instructions





When do you want Fine-Tuning?

4. Retrieval Augmented Generation (RAG) LLM



5. Parameter-Efficient Fine-Tuning (PEFT)

6. Reinforcement Learning from Human Feedback (RLHF)

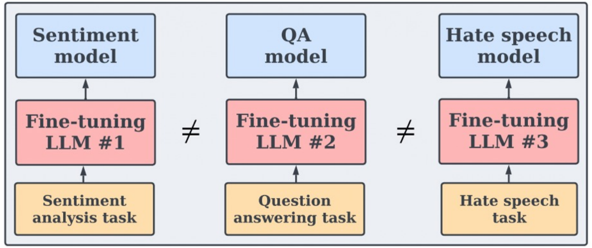
- Align with human preference

Challenges

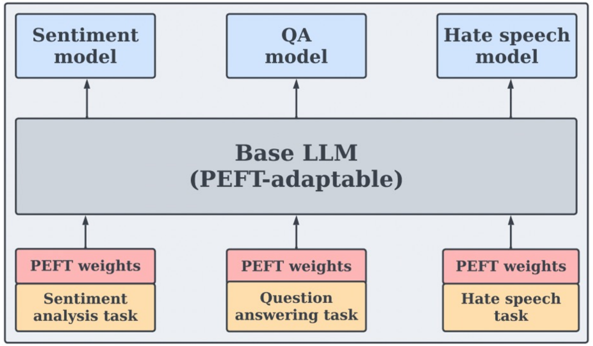
1. Memory Capacity Intensive
2. Computation Intensive

Parameter-Efficient Fine-tuning (PEFT):

a class of methods that adapt LLMs by updating only a small subset of model parameters.



(a)



(b)

Figure 5: **Fine-tuning an LLM for a specific downstream task**. (a) illustrates vanilla fine-tuning, which requires updating the entire model, resulting in a new model for each task. In (b), PEFT instead learns a small subset of model parameters for each task with a fixed base LLM. The same base model can be re-used during inference for different tasks.

https://arxiv.org/pdf/2307.10169.pdf

PEFT Taxonomy

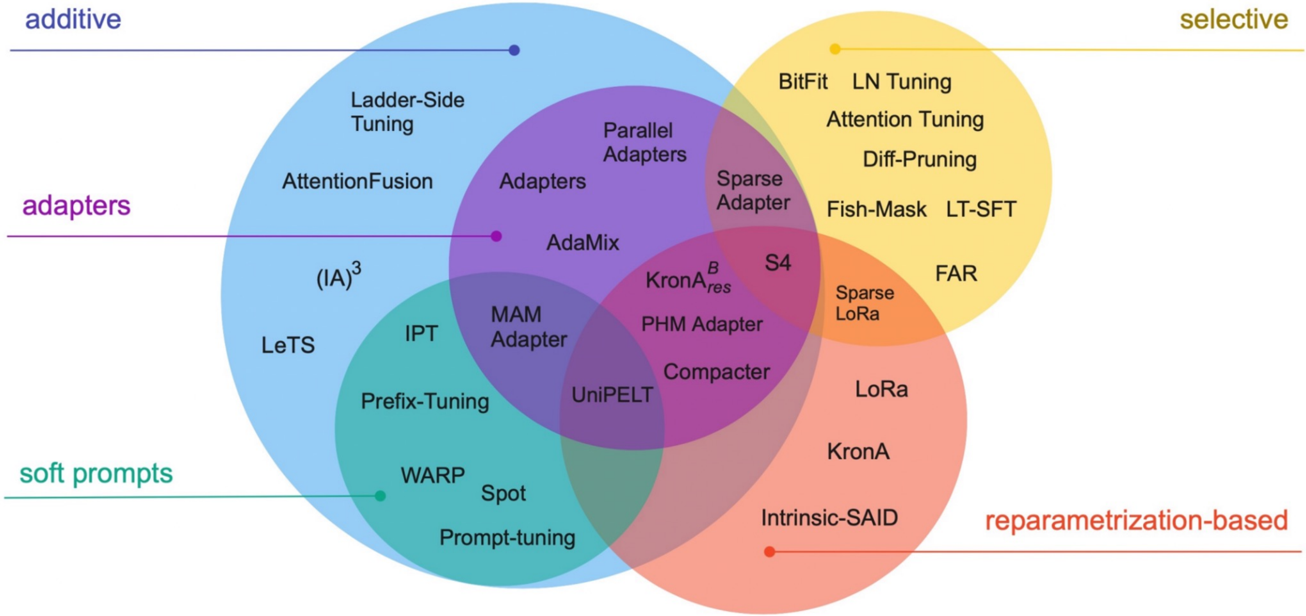
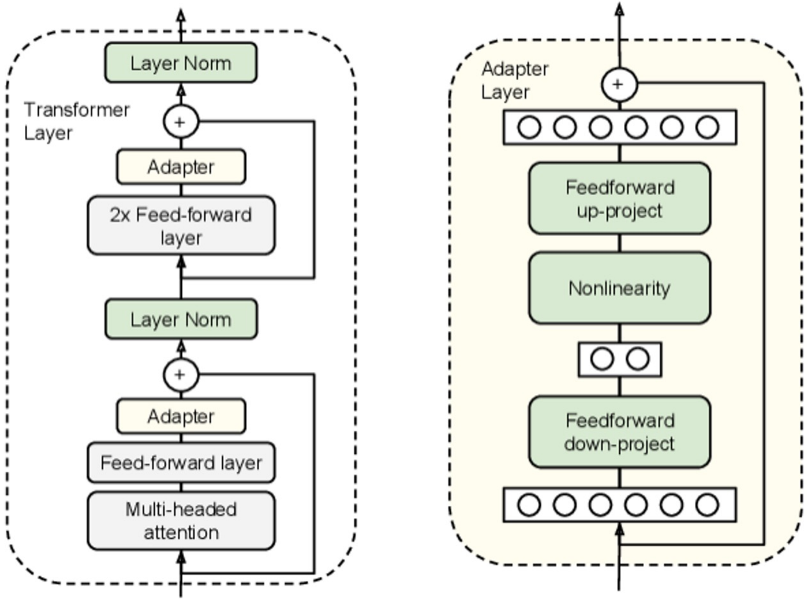


Figure 2: Parameter-efficient fine-tuning methods taxonomy. We identify three main classes of methods: **Addition**-based, **Selection**-based, and **Reparametrization**-based. Within additive methods, we distinguish two large included groups: **Adapter-like** methods and **Soft prompts**.

Addictive: Adapters

Add additional, learnable layers into a Transformer architecture.

~3%



Parameter-efficient transfer learning for nlp

Selective: BitFit

Only fine-tune the biases of the network. (<1%)

```
params = (p for n, p
          in model.named_parameters()
          if "bias" in n)
optimizer = Optimizer(params)
```

Fail when model size is large

Reparametrization-based: LoRa

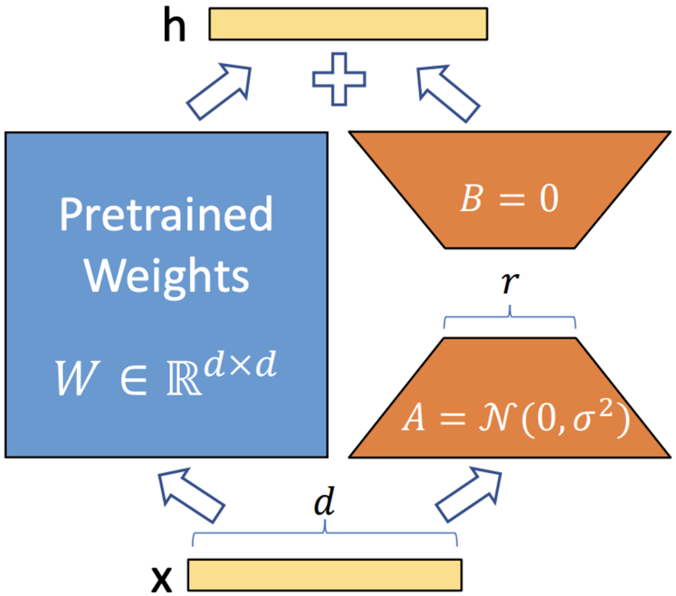


Figure 1: Our reparametrization. We only train A and B.

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

- Only update the low-rank matrix
- 10000x less trainable parameter
- 3x GPU memory requirement
- Apply to any linear layer
- No inference overhead

QLoRa

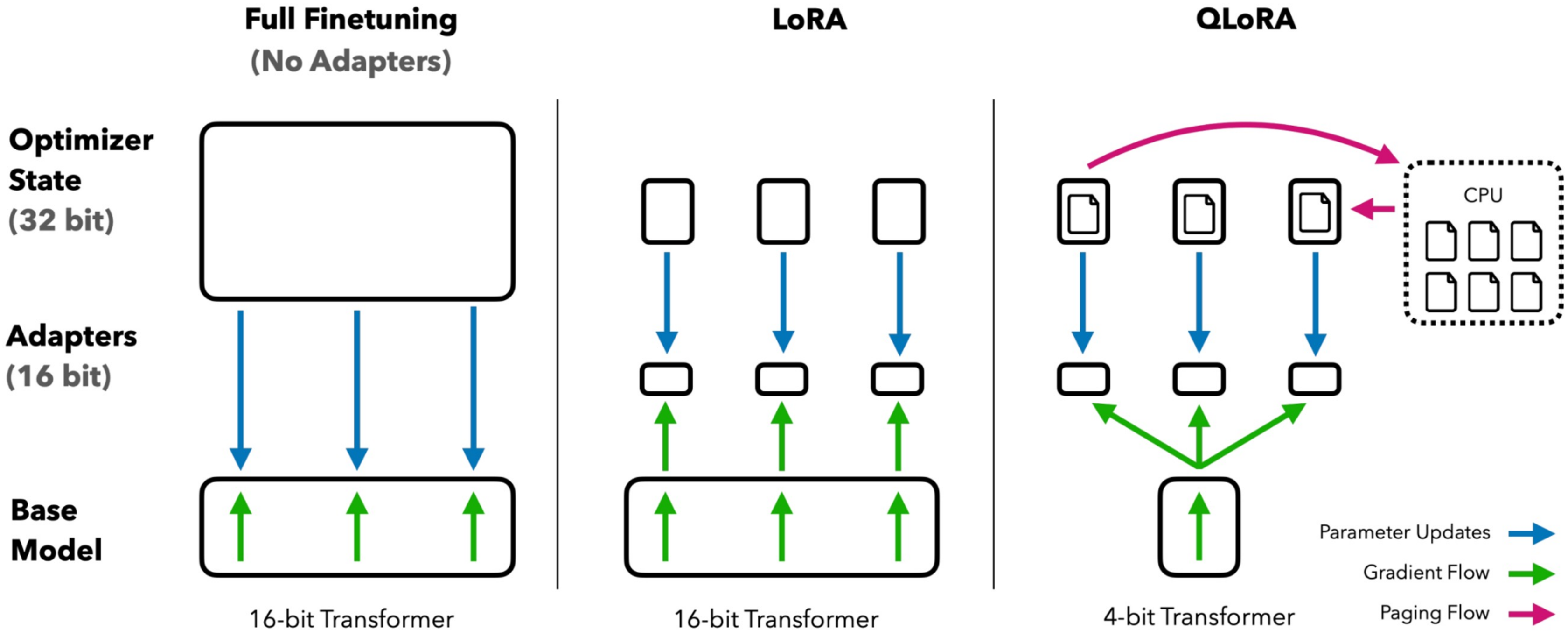


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Fine-tuning Library

1. Pytorch
2. Hugging Face - PEFT
3. Lamini
4. OpenAI Fine-tuning API

Reference

1. LoRA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS
2. Prefix Tuning: P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks
3. Prompt Tuning: The Power of Scale for Parameter-Efficient Prompt Tuning
4. P-Tuning: GPT Understands, Too
5. Parameter-efficient transfer learning for nlp
6. Challenges and Applications of Large Language Models
7. QLORA: Efficient Finetuning of Quantized LLMs
8. Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning