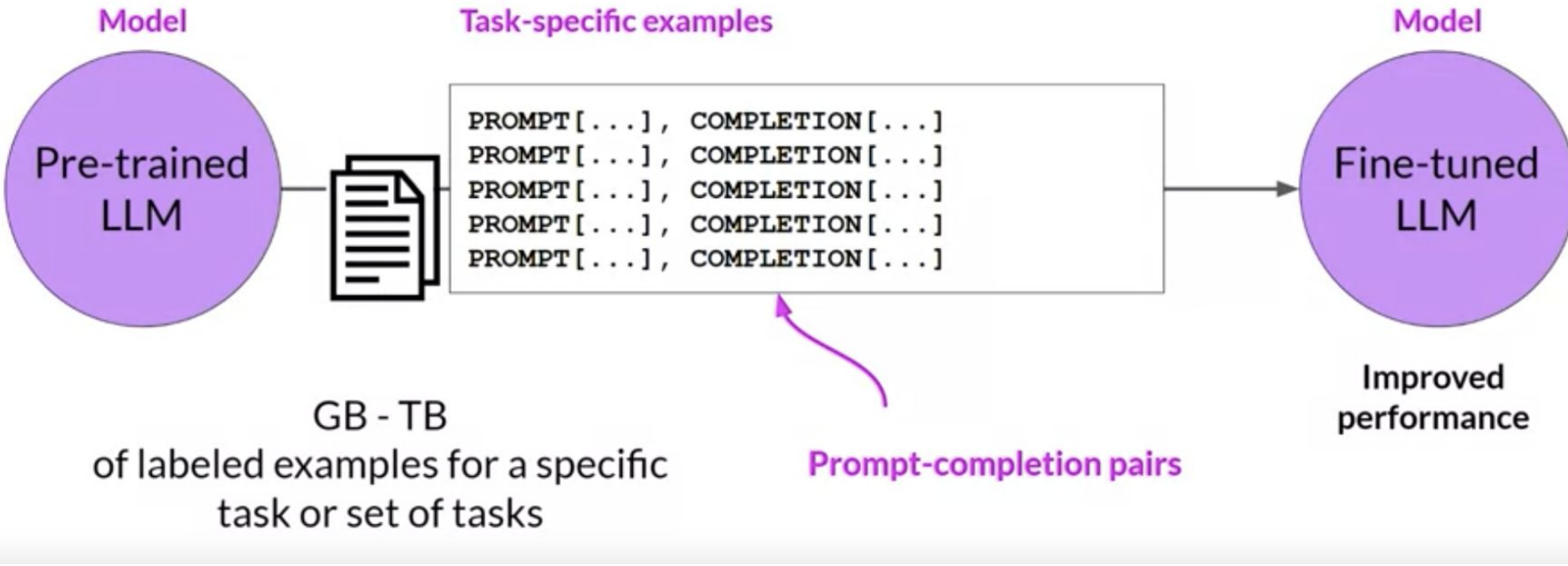
Fine-Tuning LLMs

Arin Ghazarian
Chapman University

LLM Fine-Tuning



source

Catastrophic Forgetting

- Catastrophic forgetting, also known as catastrophic interference, is a phenomenon in machine learning, particularly in neural networks, where a model forgets previously learned information upon learning new information.
- When a neural network is trained on a new task or dataset, the weights of the network are updated to minimize the loss for this new task. However, these updates can interfere with the weights associated with previously learned tasks, leading to a loss of previously acquired knowledge.
- This is a problem when we want to fine-tune a model for a task or in continual learning scenarios

How to Avoid Catastrophic Forgetting

- Fine-tune on multiple tasks
- Use Parameter-efficient fine-tuning (PEFT)

Fine-tuned LAnguage Net (FLAN)

- Chung et al., "Scaling Instruction-Finetuned Language Models"
- A specific set of instructions which is used to perform instruction fine-tuning

T0-SF

- Commonsense Reasoning,
- Question Generation,
- Closed-book QA,
- Adversarial QA,
- Extractive QA

...

55 Datasets
14 Categories
193 Tasks

Muffin

- Natural language inference,
- Code instruction gen,
- Code repair
- Dialog context generation,
- Summarization (SAMSum)

...

69 Datasets
27 Categories
80 Tasks

CoT (reasoning)

- Arithmetic reasoning,
- Commonsense reasoning
- Explanation generation,
- Sentence composition,
- Implicit reasoning,

...

9 Datasets
1 Category
9 Tasks

Natural Instructions

- Cause effect classification,
- Commonsense reasoning,
- Named Entity Recognition,
- Toxic Language Detection,
- Question answering

...

372 Datasets
108 Categories
1554 Tasks

Source: Chung et al. 2022, "Scaling Instruction-Finetuned Language Models"

FLAN-T5

- An enhanced version of T5 that has been fine-tuned with a mixture of tasks.

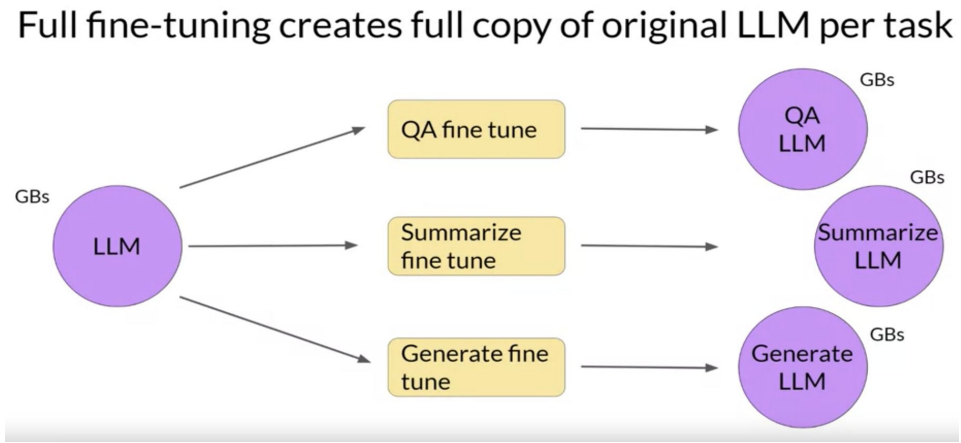
```
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer


model = AutoModelForSeq2SeqLM.from_pretrained("google/flan-t5-small")
tokenizer = AutoTokenizer.from_pretrained("google/flan-t5-small")
```

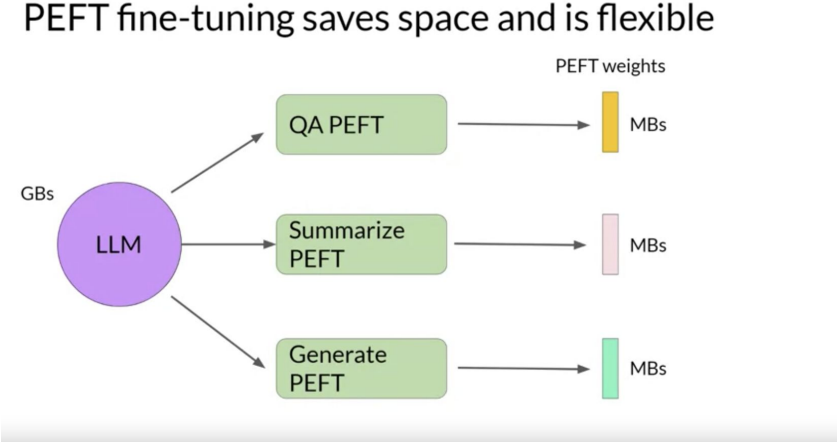
Parameter-Efficient Fine-Tuning (PEFT)

- Adjusts only a small subset of the model's parameters during fine-tuning, rather than the entire model.
- This subset could be specific layers, heads, or adapters, which makes the process more computationally efficient.
- By fine-tuning only a small portion of the model or introducing new parameters that interact with the existing ones, PEFT helps retain the pre-trained knowledge and capabilities of the model and avoids catastrophic forgetting

source

Full Fine-Tuning vs PEFT





source

PEFT Taxonomy

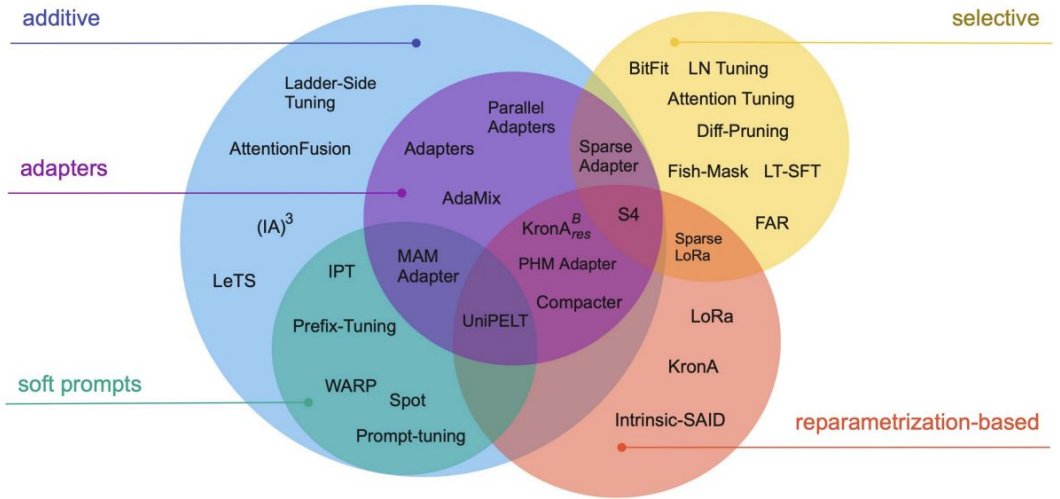


Figure 2: Parameter-efficient fine-tuning methods taxonomy. We identify three main classes of methods: **Addition**-based, **Selection**-based, and **Reparametrization**-based. Within additive methods, we distinguish two large included groups: **Adapter-like** methods and **Soft prompts**.

source

Hard Prompting vs Soft Prompting

- Soft prompts are learnable tensors that are concatenated with input embeddings and can be optimized to a dataset
- There are many methods to train the soft prompts
  - Prompt tuning
  - Prefix tuning
  - P-tuning
  - Multitask prompt tuning.

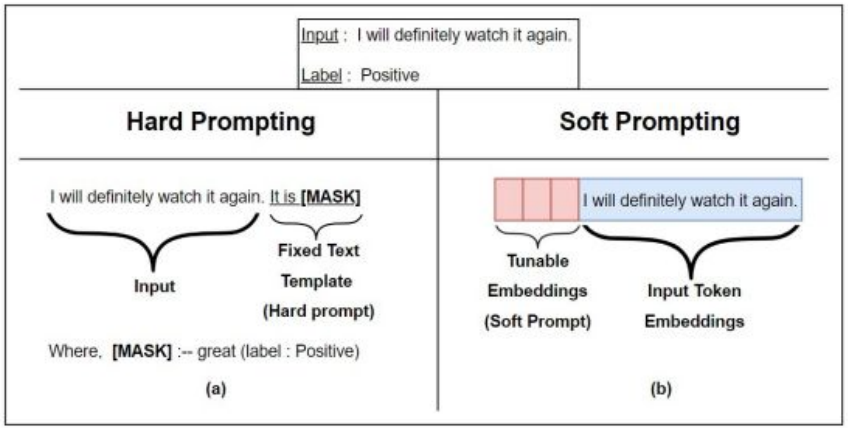
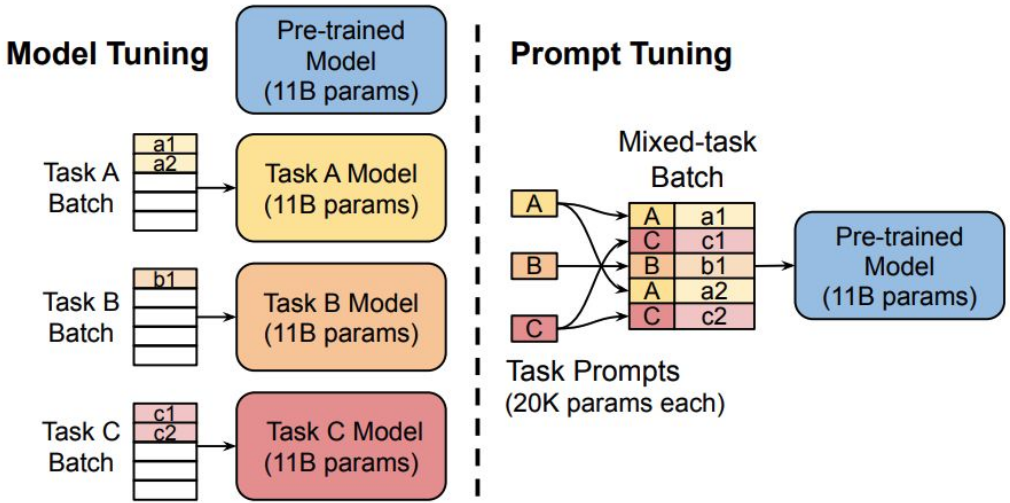


Fig. 1. The comparison between the hard prompting (part (a)) where the input is transformed into a predefined text template according to the attributes of the input (label) and soft prompting (part (b)) where a tunable piece of embedding is prepended to the input

source

Soft Prompting: Prompt Tuning

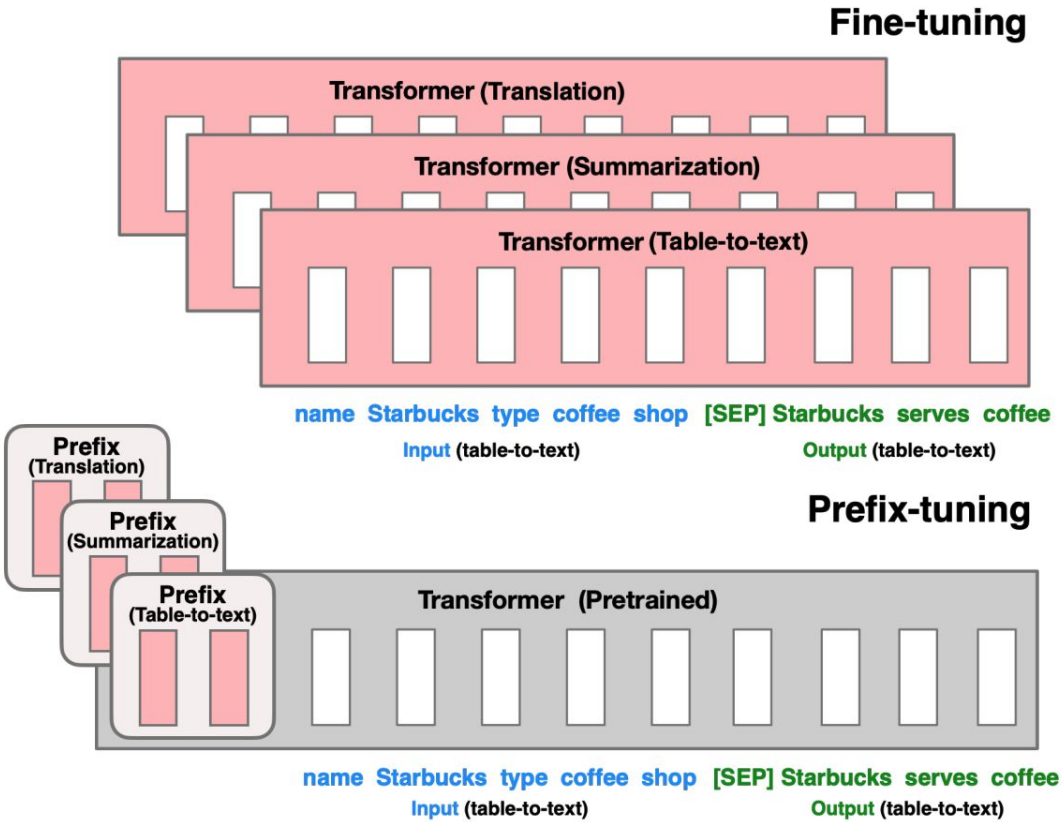
- Prompt tuning converts text classification tasks into text generation tasks by generating tokens that represent class labels, with prompts added to the input as tokens.
- Unlike traditional methods, prompt tuning keeps the model parameters fixed and updates only the parameters of the prompt tokens independently.
- The approach achieves results comparable to full model training and improves performance as the model size increases.



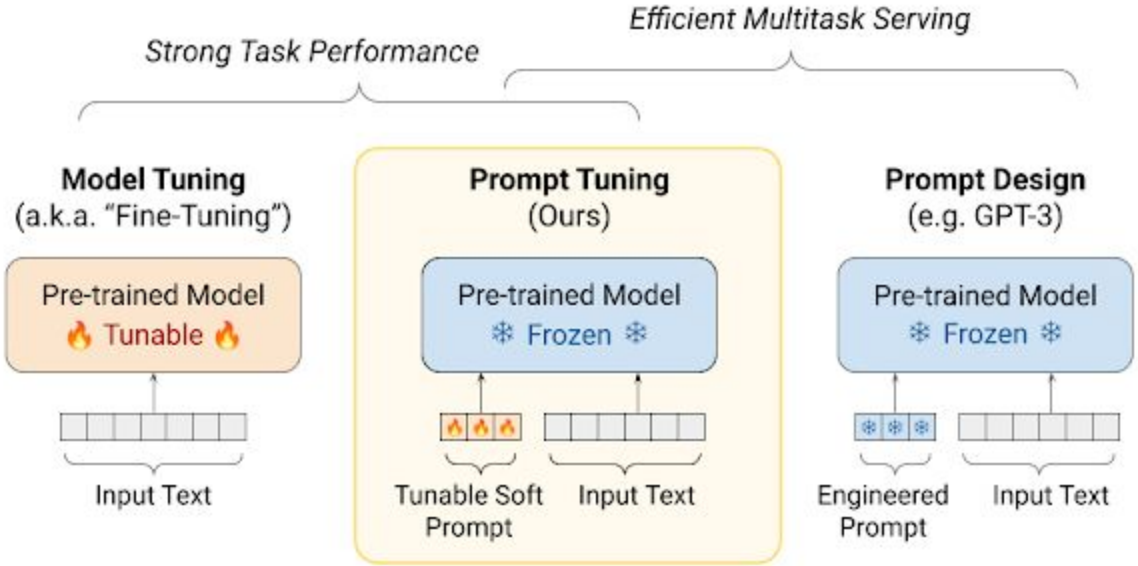
source

Soft Prompting: Prefix tuning

- Very similar to prompt tuning
- Prepends a sequence of task-specific vectors to the input that can be trained and updated while keeping the rest of the pretrained model's parameters frozen.
- The main difference is that the prefix parameters are inserted in all of the model layers, whereas prompt tuning only adds the prompt parameters to the model input embeddings.



source

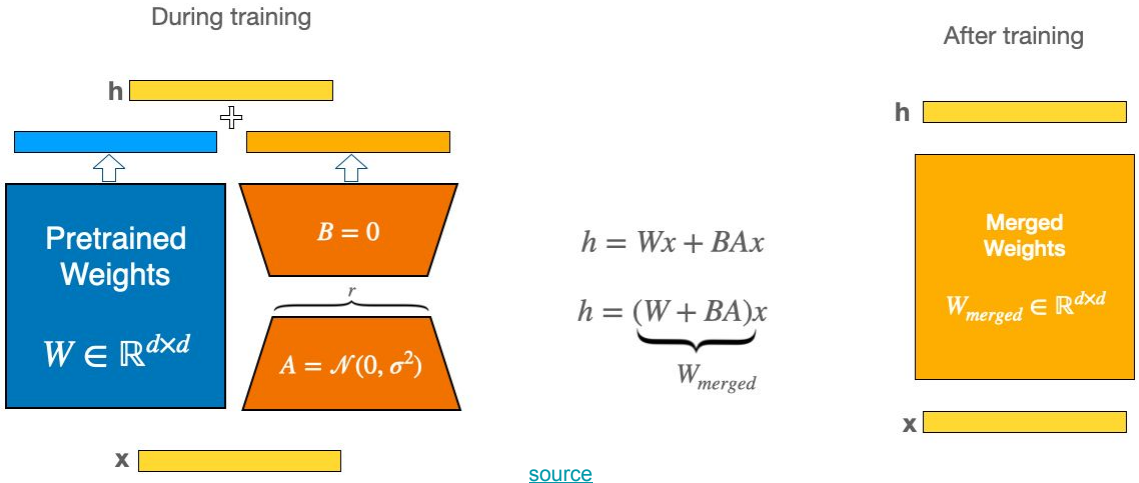

Strong Task Performance
Efficient Multitask Serving
Model Tuning
(a.k.a. "Fine-Tuning")
Pre-trained Model
🔥 Tunable 🔥
Input Text
Prompt Tuning
(Ours)
Pre-trained Model
❄ Frozen ❄
Tunable Soft Prompt
Input Text
Prompt Design
(e.g. GPT-3)
Pre-trained Model
❄ Frozen ❄
Engineered Prompt
Input Text

Soft Prompting: Demo

- https://github.com/peremartra/Large-Language-Model-Notebooks-Course/blob/main/5-Fine%20Tuning/Prompt_Tuning_PEFT.ipynb
- https://github.com/kipgparker/soft-prompt-tuning/blob/main/example.ipynb

Low Rank Adaptation (LORA)

- Hu et al., "LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS"
- LoRA introduces trainable low-rank matrices to the existing layers of a pre-trained model. Instead of fine-tuning all the parameters, it updates these low-rank matrices to capture task-specific adjustments.
- The original pre-trained weights are kept fixed, and only the low-rank adaptation parameters are trained. This helps retain the model's foundational capabilities while enabling it to adapt to new tasks or domains.
- QLoRA (Quantized Low-Rank Adaptation) is an extension of LoRA that incorporates quantization into the low-rank adaptation framework. Here's a brief overview:



source

LORA: Hugging Face Example

- We can specify the amount of parameters to be trained, in this example 0.16%

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
from peft import LoraConfig, get_peft_model, prepare_model_for_int8_training, TaskType
model = AutoModelForSeq2SeqLM.from_pretrained("google/flan-t5-xxl", load_in_8bit=True, device_map="auto")
lora_config = LoraConfig(
 r=16,
 lora_alpha=32,
 target_modules=["q", "v"],
 lora_dropout=0.05,
 bias="none",
 task_type=TaskType.SEQ_2_SEQ_LM
)
model = prepare_model_for_int8_training(model)
model = get_peft_model(model, lora_config)
model.print_trainable_parameters()
# trainable params: 18874368 || all params: 11154206720 || trainable%: 0.16921300163961817
```

r and lora_alpha parameters

"r" (Rank):

- A smaller "r" means fewer trainable parameters, making fine-tuning more efficient but potentially limiting model capacity.
- Higher "r" values allow for more complex adaptations but can lead to overfitting if not carefully managed.

"lora_alpha" (Scaling factor):

- Controls the degree to which the LoRA adapters influence the model's output during computation.
- Usually scaled by "r" during calculations (e.g., "lora_alpha/r").
- Adjusting "lora_alpha" can help fine-tune the balance between the original model and the LoRA adaptation.

LORA: DEMO

- Image classification using peft lora
- https://github.com/peremartra/Large-Language-Model-Notebooks-Course/blob/main/5-Fine%20Tuning/5_2_LoRA_Tuning.ipynb