PEFT techniques 2: Soft prompts

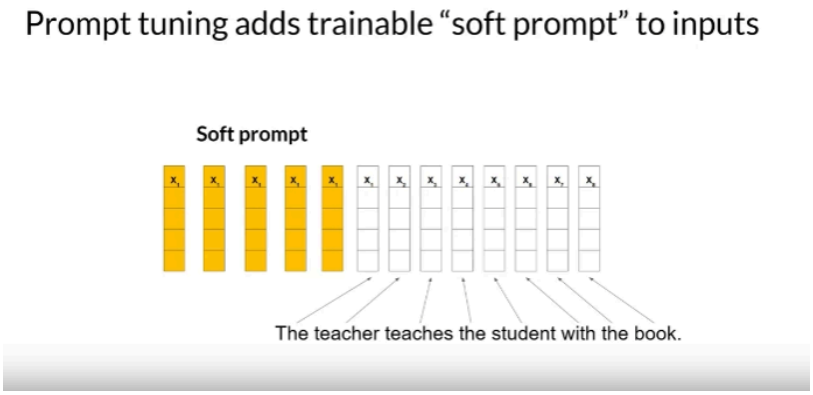
With LoRA, the goal was to find an efficient way to update the weights of the model without having to train every single parameter again. There are also additive methods within PEFT that aim to improve model performance without changing the weights at all.Here you'll explore a second parameter efficient fine tuning method called prompt tuning.

prompt tuning sounds a bit like prompt engineering, but they are quite different from each other. With prompt engineering, you work on the language of your prompt to get the completion you want. This could be as simple as trying different words or phrases or more complex, like including examples for one or Few-shot Inference. The goal is to help the model understand the nature of the task you're asking it to carry out and to generate a better completion.

prompt tuning sounds a bit like prompt engineering, but they are quite different from each other. With prompt engineering, you work on the language of your prompt to get the completion you want. This could be as simple as trying different words or phrases or more complex, like including examples for one or Few-shot Inference. The goal is to help the model understand the nature of the task you're asking it to carry out and to generate a better completion.

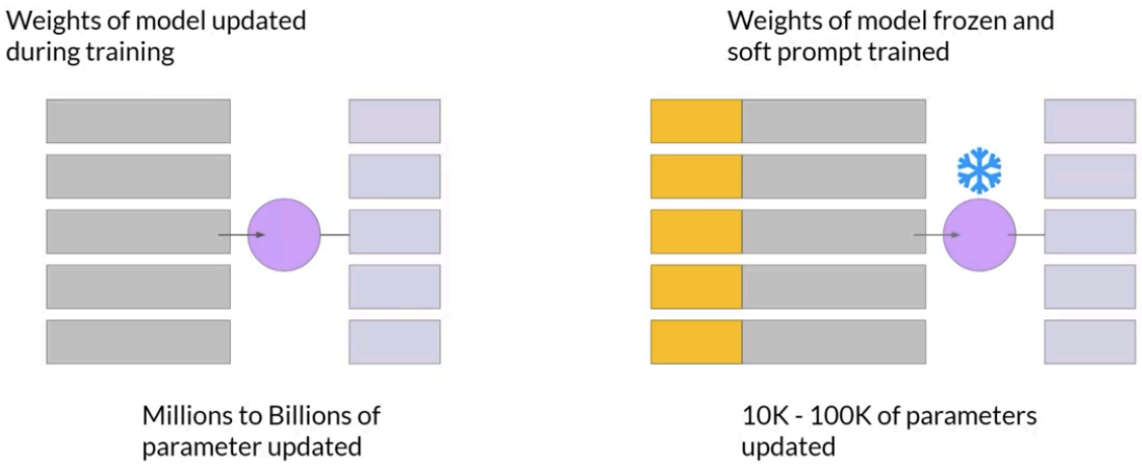
PROMPT TUNING

With prompt tuning, you add additional trainable tokens to your prompt and leave it up to the supervised learning process to determine their optimal values. The set of trainable tokens is called a soft prompt, and it gets prepended to embedding vectors that represent your input text.



- it gets prepended to embedding vectors that represent your input text. The soft prompt vectors have the same length as the embedding vectors of the language tokens. And including somewhere between 20 and 100 virtual tokens can be sufficient for good performance.
- The tokens that represent natural language are hard in the sense that they each correspond to a fixed location in the embedding vector space. However, the soft prompts are not fixed discrete words of natural language. Instead, you can think of them as virtual tokens that can take on any value within the continuous multidimensional embedding space. And through supervised learning, the model learns the values for these virtual tokens that maximize performance for a given task.
- In full fine tuning, the training data set consists of input prompts and output completions or labels. The weights of the large language model are updated during supervised learning.

Full Fine-tuning vs prompt tuning



- In contrast with prompt tuning, the weights of the large language model are frozen and the underlying model does not get updated. Instead, the embedding vectors of the soft prompt gets updated over time to optimize the model's completion of the prompt. Prompt tuning is a very parameter efficient strategy because only a few parameters are being trained. In contrast with the millions to billions of parameters in full fine tuning, similar to what you saw with LoRA.

- You can train a different set of soft prompts for each task and then easily swap them out at inference time. You can train a set of soft prompts for one task and a different set for another. To use them for inference, you prepend your input prompt with the learned tokens to switch to another task, you simply change the soft prompt. Soft prompts are very small on disk, so this kind of fine tuning is extremely efficient and flexible.
- The same LLM is used for all tasks, all you have to do is switch out the soft prompts at inference time.
- While Prompt Tuning offers a parameter-efficient approach to fine-tuning language models, it raises concerns regarding the interpretability of learned virtual tokens. Unlike traditional tokens that correspond to known words or phrases in the vocabulary, soft prompt tokens can assume any value within the continuous embedding vector space. This characteristic challenges the interpretability of the trained tokens, as they lack direct semantic associations with linguistic elements.
- Semantic Clustering Analysis:
  Despite the inherent ambiguity of soft prompt tokens, empirical analysis reveals promising insights into their interpretability. Examination of the nearest neighbour tokens to the soft prompt location demonstrates that these tokens form tight semantic clusters. In essence, words closest to the soft prompt tokens exhibit similarities in meaning, suggesting that the prompts are learning word-like representations.
- The emergence of semantic clusters around soft prompt tokens underscores their potential utility in guiding model behaviour. While these tokens may not directly correspond to conventional vocabulary items, their ability to capture semantic coherence within the embedding space enhances the model's understanding of task-specific contexts. Consequently, prompt tuning facilitates nuanced adjustments to model behaviour without explicit modification of underlying weights.
- Prompt Tuning, alongside methods like LoRA, contributes to the repertoire of parameter-efficient fine-tuning techniques. By introducing trainable tokens while preserving model weights, Prompt Tuning offers a complementary approach to LoRA's rank decomposition strategy. Both methods empower practitioners to enhance model performance across diverse tasks while minimizing computational overhead.