Parameter Efficient Fine-Tuning 101

What is Parameter Efficient Fine-Tuning (PEFT)?

It is a method to enhance LLM performance on specific tasks by introducing a small set of trainable parameters. In the fine-tuning stage, only the newly added parameters are adjusted.

PEFT methods can be categorized in various ways. Here we use classification from the paper **"Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning"** where methods are grouped based on whether they introduce new parameters to the model or fine-tune a small subset of existing parameters.

1. Additive Methods

Additive methods involve enhancing a pre-trained model by **adding** extra parameters or layers, with training focused solely on these newly introduced parameters. Examples:

1. **Ladder Side Tuning (LST)** trains a ladder side network, a compact separate network utilizing intermediate activations via shortcut connections from backbone networks for predictions, substantially reducing memory needs by bypassing backpropagation through the backbone and focusing only on the side network and ladder connections.

2. Adapters

Adapters constitute an additive parameter-efficient fine-tuning approach, incorporating small fully-connected networks following Transformer sub-layers. Examples:

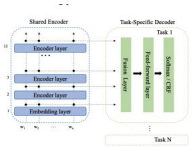
1. **AdaMix** fine-tunes a blend of adaptation modules, specific to the chosen PEFT method, within each Transformer layer while freezing majority of Pre-trained Language Model weights.
2. **Mix-And-Match (MAM)** adapter combines prefix tuning and scaled parallel adapters, inspired by findings on optimal modifications to feedforward networks and head attentions in transformer-based models.

3. Soft Prompts

Soft Prompts refine language model behavior by fine-tuning a portion of the model's input embeddings fine-tuned via gradient descent. Examples:

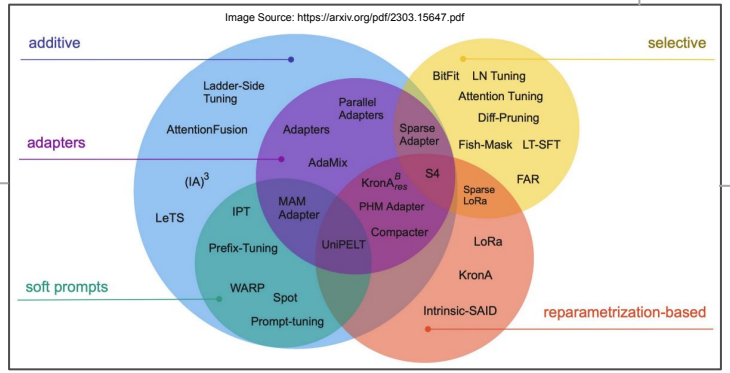
1. **Prompt-Tuning** utilizes a compact trainable model preceding the LLM to encode the text prompt and generate task-specific virtual tokens.
2. **Prefix tuning** adds task-specific vectors to the input's beginning, optimizing only the prefix parameters and integrating them into the hidden states across model layers, enabling virtual token attention.
3. **Instance-wise Prompt Tuning** (IPT) incorporates knowledge from input data instances into prompts enhancing LLMs with more detailed and context-rich information.

2. **AttentionFusion** was designed to mitigate computational overhead through a streamlined, attention-based fusion module. This module efficiently computes task-specific token representations by aggregating intermediate layer representations from a pre-trained network.
3. **IA3** (Infused Adapter by Inhibiting and Amplifying Inner Activations) fine-tunes transformer-based architectures by rescaling inner activations with injected learned vectors in the attention and feedforward modules, with these vectors being the sole trainable parameters during fine-tuning, keeping the original weights frozen.

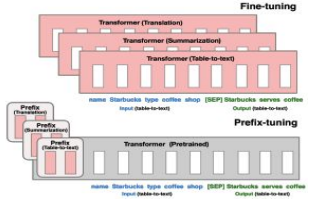


Task specific AttentionFusion model.
Source: https://aclanthology.org/2022.findings-naacl.64.pdf

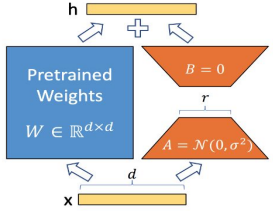
Image Source: https://arxiv.org/pdf/2303.15647.pdf



PEFT Taxonomy



Prefix Tuning Mechanism
Source: https://arxiv.org/pdf/2101.00190.pdf

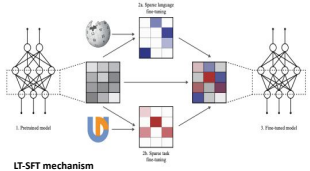


LoRA Mechanism
Source: https://arxiv.org/pdf/2106.09685.pdf

4. Selective Methods

Selective methods in Parameter-Efficient Fine-Tuning (PEFT) involve focusing on specific layers, layer types, or internal structures during the fine-tuning process, as exemplified by strategies like tuning only a few top layers, layer types, biases, or particular rows in modern approaches. Examples:

1. **BitFit** is a sparse fine-tuning method that exclusively modifies the bias terms of the model, or a designated subset, while keeping other parameters unchanged.
2. **Diff Pruning** enables scalable and parameter-efficient transfer learning for pretrained networks by learning task-specific "diff" vectors and adaptively pruning them with a differentiable L0-norm penalty.
3. **LT-SFT** introduces sparse, real-valued masks derived from the Lottery Ticket Hypothesis, achieving task-specific and language-specific adaptations without modifying the original model architecture.



LT-SFT mechanism
Source: https://aclanthology.org/2022.acl-long.125.pdf

5. Reparametrization

Reparametrization utilizez low-rank representations to minimize the number of trainable parameters, drawing from the extensively explored idea that neural networks possess low-dimensional representations in both empirical and theoretical analyses of deep learning. Examples:

1. **LoRa** (Low-Rank Adaptation) proposes a novel approach for large-scale pre-trained language models, utilizing trainable rank decomposition matrices injected into each layer of the Transformer architecture while freezing pre-trained model weights
2. **Sparse LoRa** extends the low-rank adaptation method offering dynamic adjustments to the intrinsic rank during adaptation through a gate unit optimized with the proximal gradient method.
3. **KronA** leverages the Kronecker product as a decomposition method, a powerful alternative to low-rank factorization, demonstrated to outperform it in model compression tasks . This method can be applied to all transformer-based language models
4. **Intrinsic SAID** empirically demonstrates that common pre-trained models exhibit a low intrinsic dimension, allowing effective fine-tuning with a small subset of trainable parameters. They show that larger models tend to have even lower intrinsic dimension after pre-training updates
5. **S4** introduces a novel design paradigm for parameter-efficient fine-tuning, presenting design spaces characterized by layer grouping, trainable parameter allocation, tunable groups, and strategy assignment.