Fine Tuning Large Language Model (LLM)

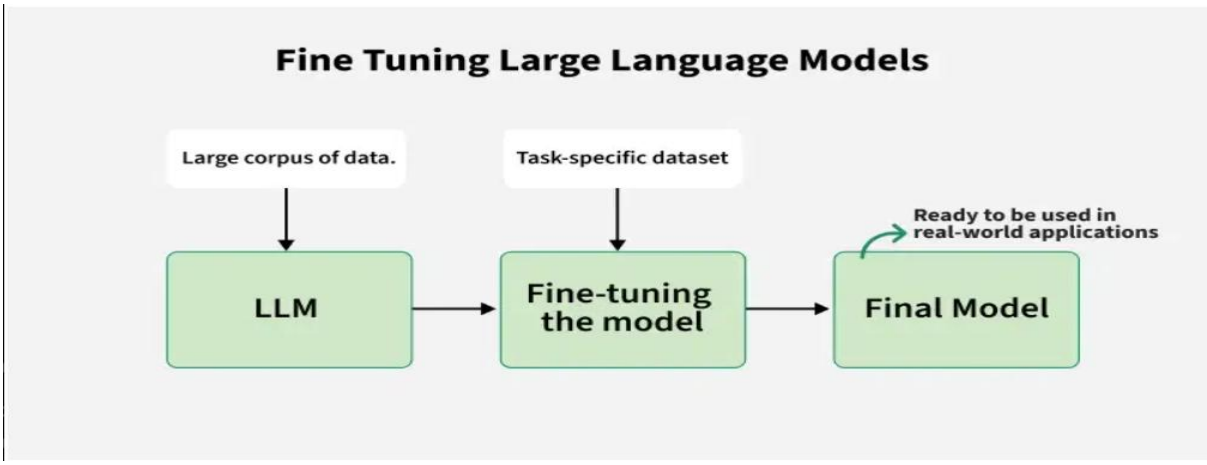
Large Language Models (LLMs) have dramatically transformed natural language processing (NLP), excelling in tasks like text generation, translation, summarization, and question-answering. However, these models may not always be ideal for specific domains or tasks.

To address this, fine-tuning is performed. **Fine-tuning** customizes pre-trained LLMs to better suit specialized applications by refining the model on smaller, task-specific datasets. This allows the model to enhance its performance while retaining its broad language proficiency.

Fine-Tuning in Large Language Models (LLMs)

Fine-tuning refers to the process of taking a pre-trained model and adapting it to a specific task by training it further on a smaller, domain-specific dataset. Fine tuning is a form of transfer learning that refines the model's capabilities, improving its accuracy in specialized tasks without needing a massive dataset or expensive computational resources.
Fine-tuning allows us to:

- **Steer the model** towards performing optimally on particular tasks.
- **Ensure model outputs** align with expected results for real-world applications.
- **Reduce model hallucinations** and improve output relevance and honesty.

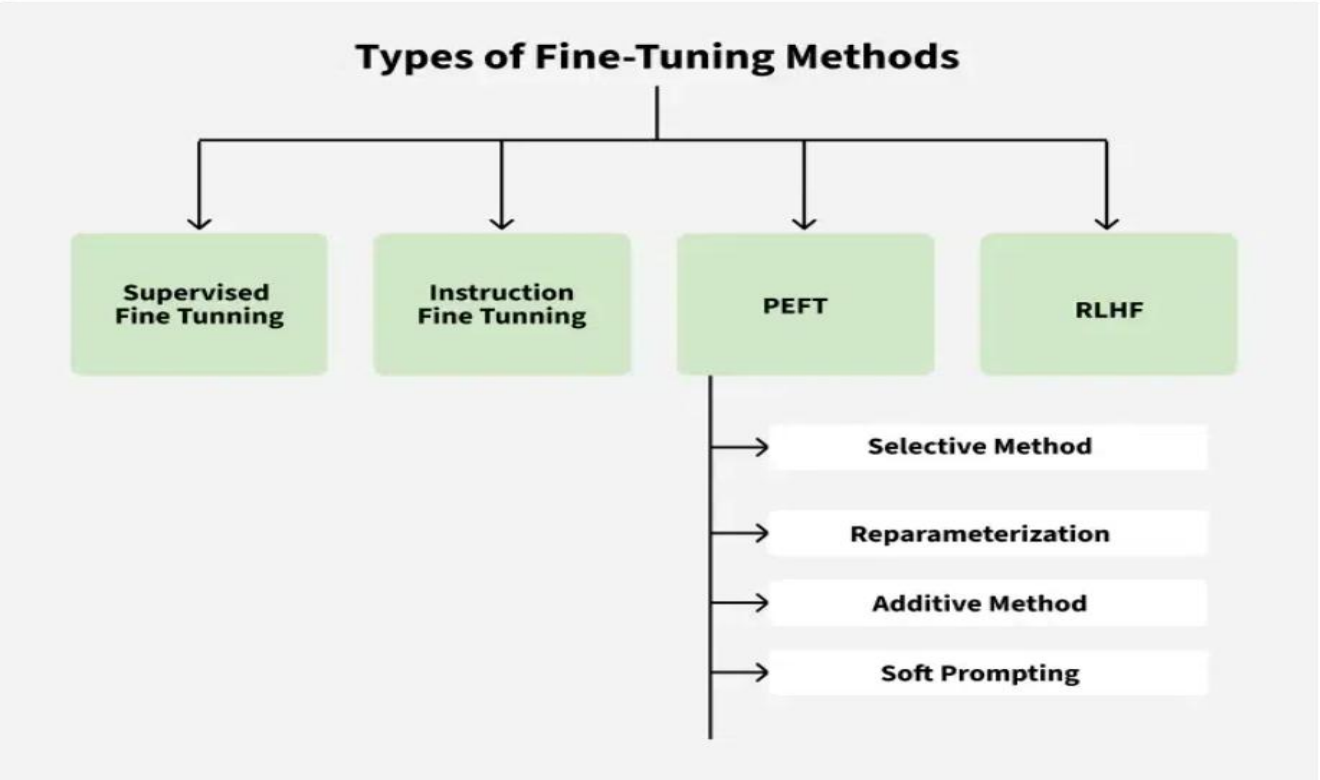


How is fine-tuning performed?

The general fine-tuning process can be broken down into the following steps:

1. **Select Base Model:** Choose a pre-trained model based on your task and compute budget.
2. **Choose Fine-Tuning Method:** Select the most appropriate method (e.g., supervised, instruction-based, PEFT) based on the task and dataset.
3. **Prepare Dataset:** Structure your data for task-specific training, ensuring the format matches the model's requirements.
4. **Training:** Use frameworks like TensorFlow, PyTorch, or high-level libraries like **Transformers** to fine-tune the model.
5. **Evaluate and Iterate:** Test the model, refine it as necessary, and re-train to improve performance.

Types of Fine Tuning Methods



Supervised Fine-Tuning

Supervised fine-tuning involves further training a pre-trained model using a task-specific dataset with labeled input-output pairs. This process allows the model to learn how to map inputs to outputs based on the given dataset.

Process:

1. Use a pre-trained model.
2. Prepare a dataset with input-output pairs as expected by the model.
3. Adjust the pre-trained weights during fine-tuning to adapt the model to the new task.
4.

Supervised fine-tuning is ideal for tasks such as **sentiment analysis, text classification, and named entity recognition** where labeled datasets are available.

Instruction Fine-Tuning

Instruction fine-tuning augments input-output examples with detailed instructions in the prompt template. This allows the model to generalize better to new tasks, especially those involving natural language instructions.

Process:

- Use a pre-trained model.
- Prepare a dataset in the form of instruction-response pairs.
- Train the model with the instruction fine-tuning process, similar to neural network training.

Instruction fine-tuning is commonly used in building **chatbots, question answering systems,** and other tasks that require **natural language interaction**.

Parameter-Efficient Fine-Tuning (PEFT)

Training a full model is resource-intensive. PEFT methods enable the efficient use of memory and computation by modifying only a subset of the model's parameters, significantly reducing the required memory for training.

PEFT Methods:

1. **Selective Method**: Freeze most layers of the model and only fine-tune specific layers.
2. **Reparameterization Method** (LoRA): Use low-rank matrices to reparametrize model weights, freezing the original weights and adding small, trainable parameters.
   Example: If a model has a dimension of 512x64, full fine-tuning would require 32,768 parameters. With LoRA, the number of parameters can be reduced to 4,608.
3. **Additive Method**: Add new layers to the encoder or decoder side of the model and train these for the specific task.
4. **Soft Prompting**: Train only the new tokens added to the model prompt, keeping other tokens and weights frozen.

PEFT is useful when working with large models that exceed memory limits, reducing both training costs and resource requirements

Reinforcement Learning with Human Feedback (RLHF)

RLHF aligns a fine-tuned model's output to human preferences using reinforcement learning. This method refines model behaviour after the initial fine-tuning phase.

Process:

1. **Prepare Dataset**: Generate prompt-completion pairs and rank them based on human evaluators' alignment criteria.
2. **Train Reward Model**: Build a reward model that scores completions based on human feedback.
3. **Update Model**: Use reinforcement learning, typically the **PPO algorithm**, to update the model weights based on the reward model.

RLHF is ideal for tasks where human-like outputs are necessary, such as generating text that aligns with user expectations or ethical guidelines.

Benefits of Fine Tuning LLMs

Fine-tuning offers several advantages:

- **Increased Performance:** Fine-tuned models adapt to new data, leading to more accurate and reliable outputs.
- **Efficiency:** Fine-tuning saves computational costs by adapting pre-trained models rather than training a model from scratch.
- **Domain Adaptation:** LLMs can be tailored to specific industries like medical, legal, or financial domains by focusing on relevant terminology and structures.
- **Better Generalization:** Models fine-tuned on task-specific data generalize better to the unique patterns and structures of the task.

When to use fine-tuning?

When we build an LLM application the first step is to select an appropriate pre-trained or foundation model suitable for our use case. Once the base model is selected we should try prompt engineering to quickly see whether the model fits our use case realistically or not and evaluate the performance of the base model on our use case.

In case with prompt engineering we are not able to achieve a reasonable level of performance we should proceed with fine-tuning. Fine-tuning should be done when we want the model to specialize for a particular task or set of tasks and have a labelled unbiased diverse dataset available. It is also advisable to do fine-tuning for domain-specific adoption like learning medical law or finance language.