[https://www.superannotate.com/blog/llm-fine-tuning#:~:text=Large%20language%20model%20(LLM)%20fine,turning%20them%20into%20specialized%20models](https://www.superannotate.com/blog/llm-fine-tuning" \l ":~:text=Large%20language%20model%20(LLM)%20fine,turning%20them%20into%20specialized%20models).

# Supervised fine-tuning (SFT)

Supervised fine-tuning means updating a pre-trained language model using labeled data to do a specific task. The data used has been checked earlier. This is different from unsupervised methods, where data isn't checked. Usually, the initial training of the language model is unsupervised, but fine-tuning is supervised.

### **How is fine-tuning performed?**

A diagram of a computer

Description automatically generated

Once your instruction data set is ready, as with standard supervised learning, you divide the data set into training validation and test splits. During fine-tuning, you select prompts from your training data set and pass them to the LLM, which then generates completions.

The aim is to adapt the previously learned general knowledge to the nuances and specific patterns present in the new dataset, thereby making the model more specialized and effective for the target task.

During this process, the model is updated with the labeled data. It changes based on the difference between its guesses and the actual answers. This helps the model learn details found in the labeled data. By doing this, the model improves at the task for which it's fine-tuned.

# Fine-tuning methods

### **Instruction fine-tuning**

One strategy used to improve a model's performance on various tasks is instruction fine-tuning. It's about training the machine learning model using examples that demonstrate how the model should respond to the query. The dataset you use for fine-tuning large language models has to serve the purpose of your instruction. For example, suppose you fine-tune your model to improve its summarization skills. In that case, you should build up a dataset of examples that begin with the instruction to summarize, followed by text or a similar phrase. In the case of translation, you should include instructions like “translate this text.” These prompt completion pairs allow your model to "think" in a new niche way and serve the given specific task.

A diagram of a program

Description automatically generated

### **Full fine-tuning**

Instruction fine-tuning, where all of the model's weights are updated, is known as full fine-tuning. The process results in a new version of the model with updated weights. It is important to note that just like pre-training, full fine-tuning requires enough memory and compute budget to store and process all the gradients, optimizers, and other components being updated during training.

### **Parameter-efficient fine-tuning**

Training a language model is a computationally intensive task. For a full LLM fine-tuning, you need memory not only to store the model, but also the parameters that are necessary for the training process. Your computer might be able to handle the model weights, but allocating memory for optimizing states, gradients, and forward activations during the training process is a challenging task. Simple hardware cannot handle this amount of hurdle. This is where PEFT is crucial. While full LLM fine-tuning updates every model's weight during the supervised learning process, **PEFT methods only update a small set of parameters**. This transfer learning technique chooses specific model components and "freezes" the rest of the parameters. The result is logically having a much smaller number of parameters than in the original model (in some cases, just 15-20% of the original weights; LoRA can reduce the number of trainable parameters by 10,000 times). This makes memory requirements much more manageable. Not only that, but PEFT is also dealing with catastrophic forgetting. Since it's not touching the original LLM, the model does not forget the previously learned information. Full fine-tuning results in a new version of the model for every task you train on. Each of these is the same size as the original model, so it can create an expensive storage problem if you're fine-tuning for multiple tasks.

### **Other types of fine-tuning**

Let's learn a few more types of learning:

**Transfer learning:** Transfer learning is about taking the model that had learned on general-purpose, massive datasets and training it on distinct, task-specific data. This dataset may include labeled examples related to that domain. Transfer learning is used when there is not enough data or a lack of time to train data; the main advantage of it is that it offers a higher learning rate and accuracy after training. You can take existing LLMs that are pre-trained on vast amounts of data, like GPT ¾ and BERT, and customize them for your own use case.

**Task-specific fine-tuning:** Task-specific fine-tuning is a method where the pre-trained model is fine-tuned on a specific task or domain using a dataset designed for that domain. This method requires more data and time than transfer learning but can result in higher performance on the specific task.

For example, translation using a dataset of examples for that task. Interestingly, good results can be achieved with relatively few examples (500-1000). Often, just a few hundred or thousand examples can result in good performance compared to the billions of pieces of text that the model saw during its pre-training phase. However, there is a potential downside to fine-tuning on a single task. The process may lead to a phenomenon called **catastrophic forgetting**.

Catastrophic forgetting happens because the full fine-tuning process modifies the weights of the original LLM. While this leads to great performance on a single fine-tuning task, it can degrade performance on other tasks. For example, while fine-tuning can improve the ability of a model to perform certain NLP tasks like [sentiment analysis](https://www.superannotate.com/blog/sentiment-analysis-explained) and result in  quality completion, the model may forget how to do other tasks. This model knew how to carry out named entity recognition before fine-tuning correctly identifying.

**Sequential fine-tuning:** Sequential fine-tuning is about sequentially adapting a pre-trained model on several related tasks. After the initial transfer to a general domain, the LLM might be fine-tuned on a more specific subset. For instance, it can be fine-tuned from general language to medical language and then from medical language to pediatric cardiology.

Note that there are other fine-tuning examples – multi-task, adaptive, behavioral, and instruction fine-tuning for large language models. These cover some important specific cases for training language models.

# Fine-tuning best practices

**Clearly define your task:**

Defining your task is a foundational step in the process of fine-tuning large language models. A clearly defined task offers focus and direction. It ensures that the model's vast capabilities are channeled towards achieving a specific goal, setting clear benchmarks for performance measurement.

**Choose and use the right pre-trained model:**

Using pre-trained models for fine-tuning large language models is crucial because it leverages knowledge acquired from vast amounts of data, ensuring that the model doesn't start learning from scratch. This approach is both computationally efficient and time-saving. Additionally, pre-training captures general language understanding, allowing fine-tuning to focus on domain-specific nuances, often resulting in better model performance in specialized tasks.

While leveraging pre-trained models provides a robust starting point, the choice of model architecture — including advanced strategies like [Mixture of Experts (MoE) and Mixture of Tokens (MoT)](https://www.superannotate.com/blog/mixture-of-experts-vs-mixture-of-tokens) — is crucial in tailoring your model more effectively. These strategies can significantly influence how the model handles specialized tasks and processes language data.

**Set hyperparameters:**

Hyperparameters are tunable variables that play a key role in the model training process. Learning rate, batch size, number of epochs, weight decay, and other parameters are the key hyperparameters to adjust that find the optimal configuration for your task.

**Evaluate model performance:**

Once fine-tuning is complete, the model's performance is assessed on the test set. This provides an unbiased evaluation of how well the model is expected to perform on unseen data. Consider also iteratively refining the model if it still has potential for improvement.

## A video source

<https://www.youtube.com/watch?v=eC6Hd1hFvos&t=15s>

A screenshot of a computer

Description automatically generated

A screen shot of a graph

Description automatically generated

A screenshot of a computer

Description automatically generated

Transfer Learning: Freeze most of layers, unfreeze couple of last ones, and train unfreeze layers.

A screenshot of a computer

Description automatically generated

## LoRA

A screenshot of a computer

Description automatically generated

**PEFT -> LoRA:** freeze all layers of base model. Add additional trainable parameters for some layers(which layers I don’t know now!!!). This helps use to keep previously learned knowledge intact, but we also add more trainable layers with less number of parameters ((d\*r + r\*k) many of them) to fine-tune base model for our task. If we fully train a layer of d\*k (1000\*1000), we have million parameters to train, if we use d\*r + r\*k, we have 1000\*2 + 2\*1000 = 2000 parameters. So if we do it for many layers, we get rid of millions of trainable parameters.

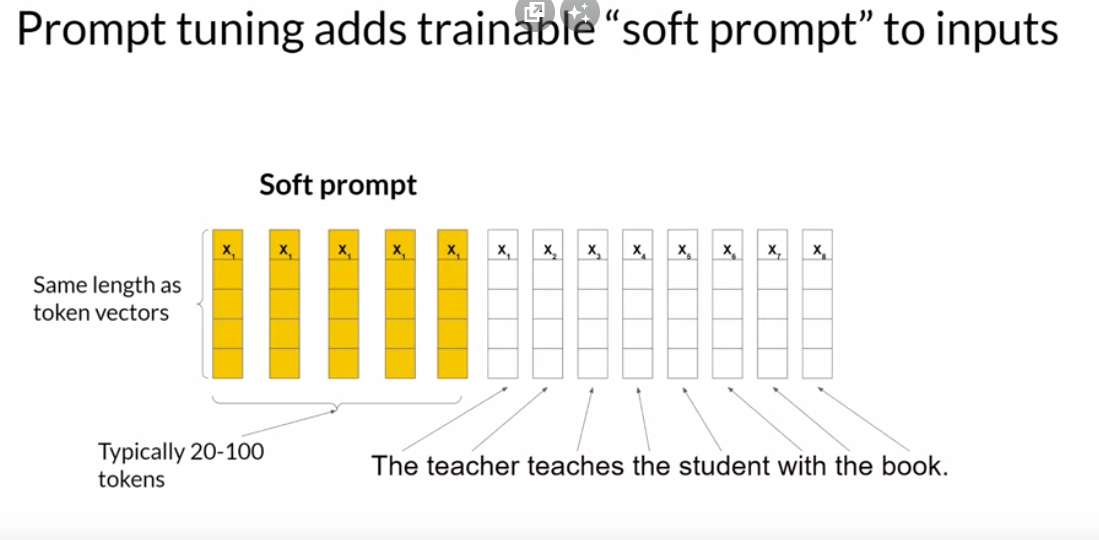
**Choice of LoRa rank for different metrics**

A screenshot of a graph

Description automatically generated

Check also QLora

## SOFT PROMPT



A screenshot of a graph

Description automatically generated

A graph with colorful lines and numbers

Description automatically generated

# OTHER INFO

The easy way to reach fine tuning models:

A screenshot of a video

Description automatically generated

## Hugging-Face Transformers

<https://www.youtube.com/watch?v=jan07gloaRg&list=PLz-ep5RbHosU2hnz5ejezwaYpdMutMVB0&index=3>

## Coursera Course

https://www.coursera.org/learn/generative-ai-with-llms/lecture/NZOVw/peft-techniques-1-lora