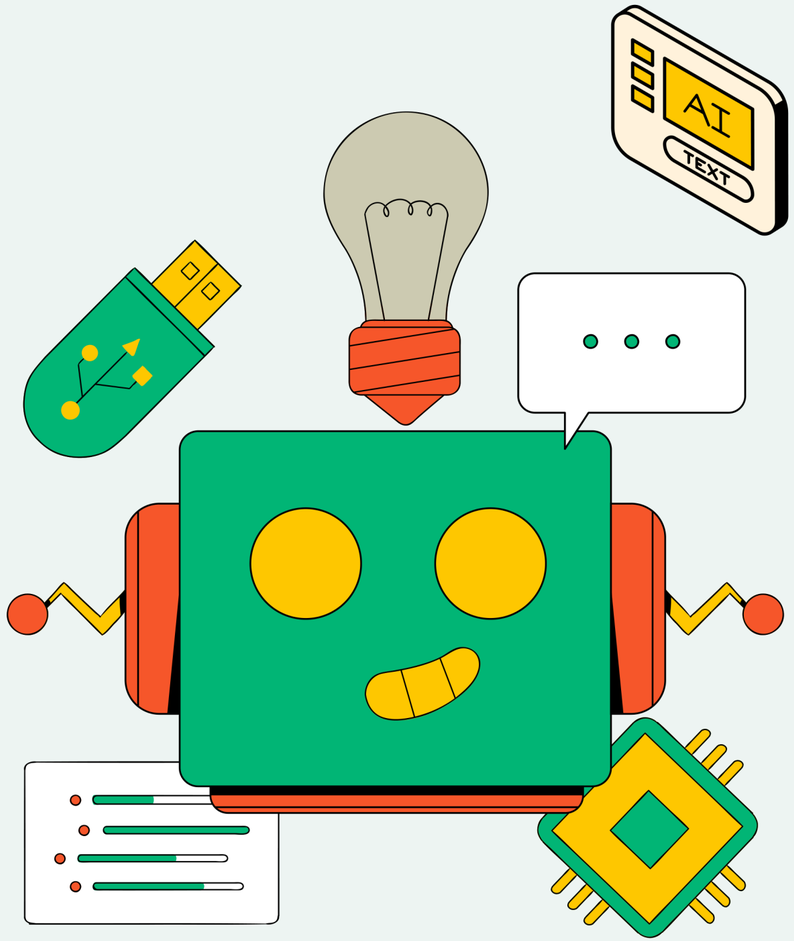
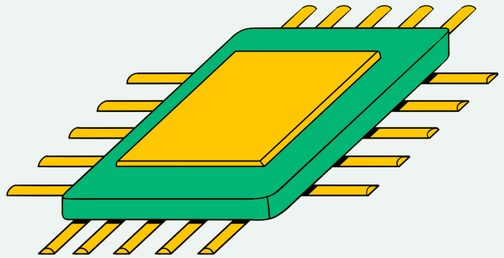


FINE-TUNING TECHNIQUES FOR LARGE LANGUAGE MODELS (LLMS)

PRESENTATION

PRESENTED BY:

ISMAIL BOKRI



PRESENTATION OUTLINE

- Introduction
- fine-tuning goals and benefits
- Fine-tuning techniques
- Conclusion
- Questions and Answers

INTRODUCTION

As technology continues to advance, machine learning models have become more powerful in solving a wide range of tasks. Fine-tuning a model is one such technique that allows us to adapt pre-trained neural network models for specific tasks or datasets.

FINE-TUNING GOALS AND BENEFITS

Fine-tuning is taking a **pre-trained** model and making minor adjustments to its **internal parameters** to specialize it in one specific task. The goal is to **optimize** the model's performance on a new, related task **without starting the training process from scratch**.

Benefits

Improved Performance

Pre-trained large language models (LLMs) have been trained on **vast amounts of data** for general tasks. By fine-tuning a pre-trained model, we can leverage this wealth of knowledge and representations, leading to improved performance on our **specific task**.

Data Efficiency

In many real-world scenarios, obtaining **labeled data** for a specific task can be challenging and **time-consuming**. Fine-tuning offers a solution by allowing us to effectively train models even with **limited labeled data**.

WHY FINE-TUNING?

There are many reasons why you should choose the fine-tuning approach, but **lower costs** and less infrastructure requirements are always the major ones.

Building a large language model (LLM) from scratch can be **very expensive, time-consuming, and energy-intensive**. A customer who wants their own LLM to answer specific questions often cannot provide these resources. That’s why fine-tuning has gained all this popularity.

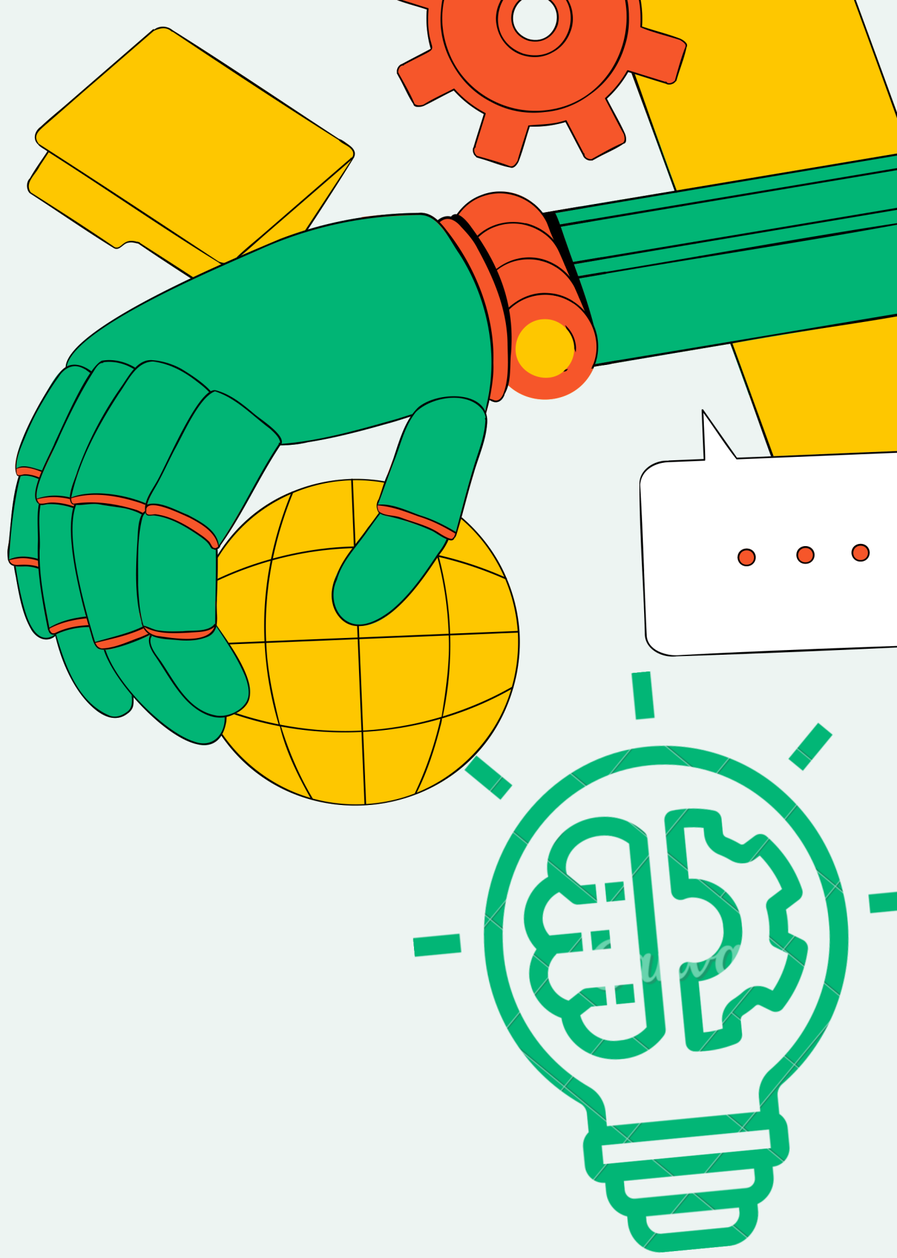
--> There are only a few LLMs in the world, and all other models are built upon them, either through fine-tuning or other techniques.

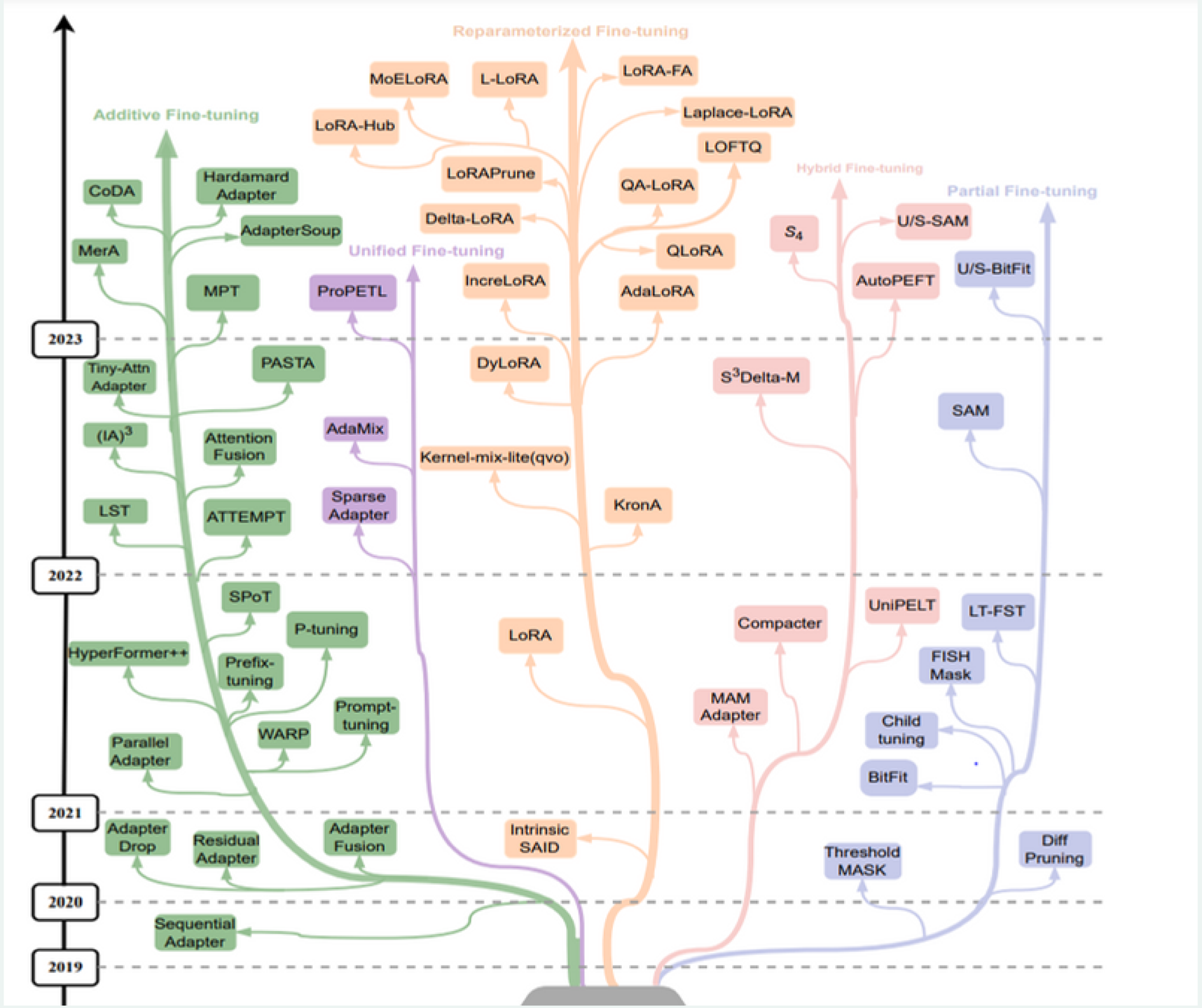
FINE-TUNING TECHNIQUES

- It is true that fine-tuning can save time and resources since we are working with matrices that already exist and just need to modify them. However, fine-tuning can still be a challenging task to achieve.

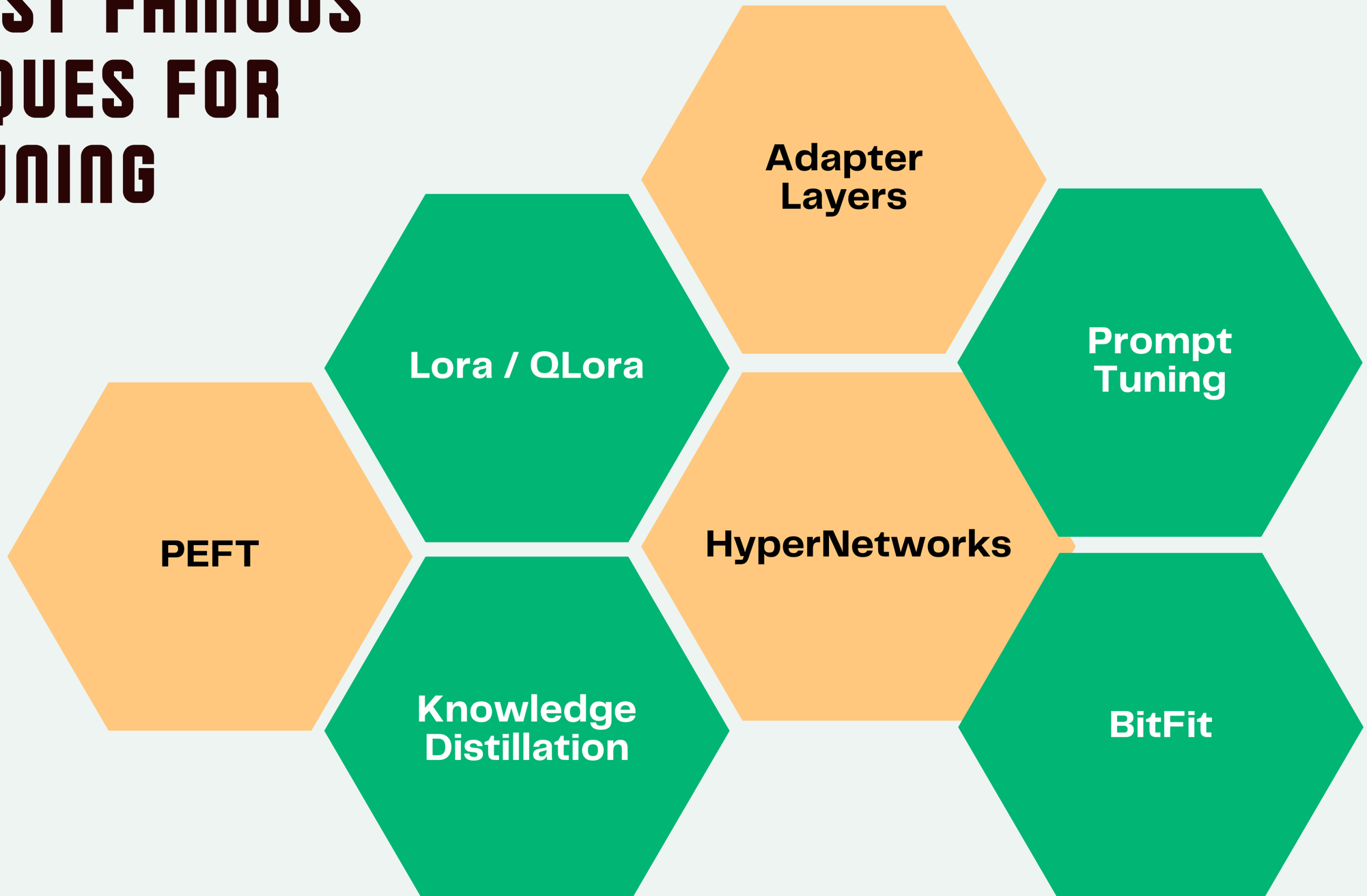
- The weight matrices in these LLM models are very large; for example, a 100-billion-parameter model has 100 billion weights. These weights often need to be updated repeatedly over multiple epochs.

- This is why the right fine-tuning technique is crucial for optimizing our work.




Reparameterized Fine-tuning
Additive Fine-tuning
Unified Fine-tuning
Hybrid Fine-tuning
Partial Fine-tuning
MoELoRA
L-LoRA
LoRA-FA
LoRA-Hub
Laplace-LoRA
LOFTQ
LoRAPrune
QA-LoRA
Delta-LoRA
QLoRA
IncreLoRA
AdaLoRA
CoDA
Hardamard Adapter
AdapterSoup
MerA
MPT
ProPETL
S_4
U/S-SAM
AutoPEFT
U/S-BitFit
2023
PASTA
DyLoRA
S^3Delta-M
Tiny-Attn Adapter
SAM
$(IA)^3$
Attention Fusion
AdaMix
Kernel-mix-lite(qvo)
Sparse Adapter
KronA
LST
ATTEMPT
2022
SPoT
P-tuning
LoRA
Compacter
UniPELT
LT-FST
HyperFormer++
Prefix-tuning
FISH Mask
MAM Adapter
Prompt-tuning
WARP
Child tuning
Parallel Adapter
BitFit
2021
Adapter Drop
Residual Adapter
Adapter Fusion
Intrinsic SAID
Threshold MASK
Diff Pruning
2020
Sequential Adapter
2019

THE MOST FAMOUS TECHNIQUES FOR FINE-TUNING



PARAMETER-EFFICIENT FINE-TUNING (PEFT)

- PEFT is a fine-tuning approach that only fine-tunes a **small number** of model parameters while **freezing** most parameters of the pretrained LLMs.

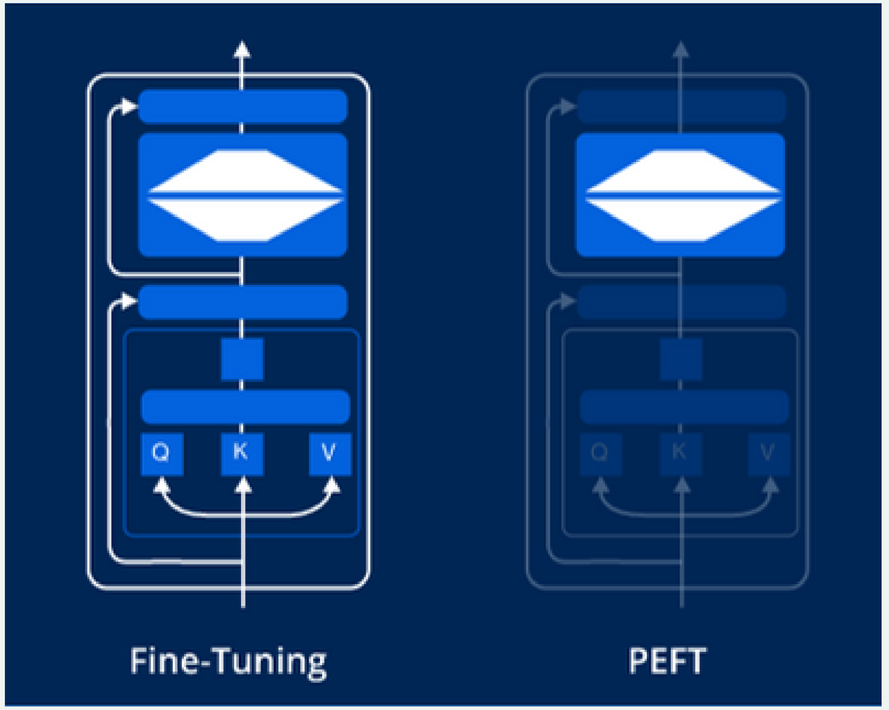
- PEFT overcomes the issues of **catastrophic forgetting**, a behavior observed during the full fine-tuning of LLMs. This occurs when fine-tuning changes most of the weights, causing the model to lose its initial performance.

- PEFT approaches have been shown to be better than fine-tuning in **low-data regimes** and generalize better to out-of-domain scenarios.

- PEFT approaches enable you to get performance comparable to full fine-tuning while only having a small number of trainable parameters .

- PEFT achieves this efficiency by freezing some of the layers of the pre-trained model and o**nly fine-tuning the last few layers** that are specific to the downstream task.

- The model can be adapted to new tasks with less computational overhead and fewer labeled examples.



LOW-RANK ADAPTATION (LORA) :

- Lora is a thechnique designed to fine-tune large pre-trained models efficiently by focusing on low-rank updates. This means it adds low-rank matrices to the weights of the original model rather than updating all the weights.

- Instead of working with the original large matrices of the pre-trained LLM, the LoRA (Low-Rank Adaptation) technique divides the original model into two separate, smaller matrices that get multiplied together to form a matrix the same size as the model's weight matrix.

- Only these smaller matrices are adjusted during training, thus reducing the number of parameters to be optimized and saving computational resources.

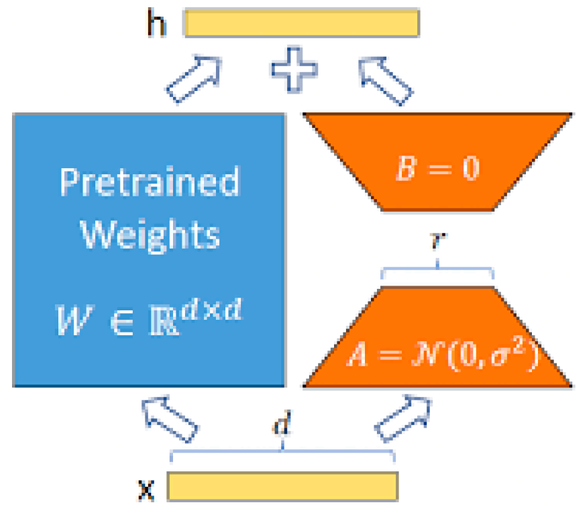
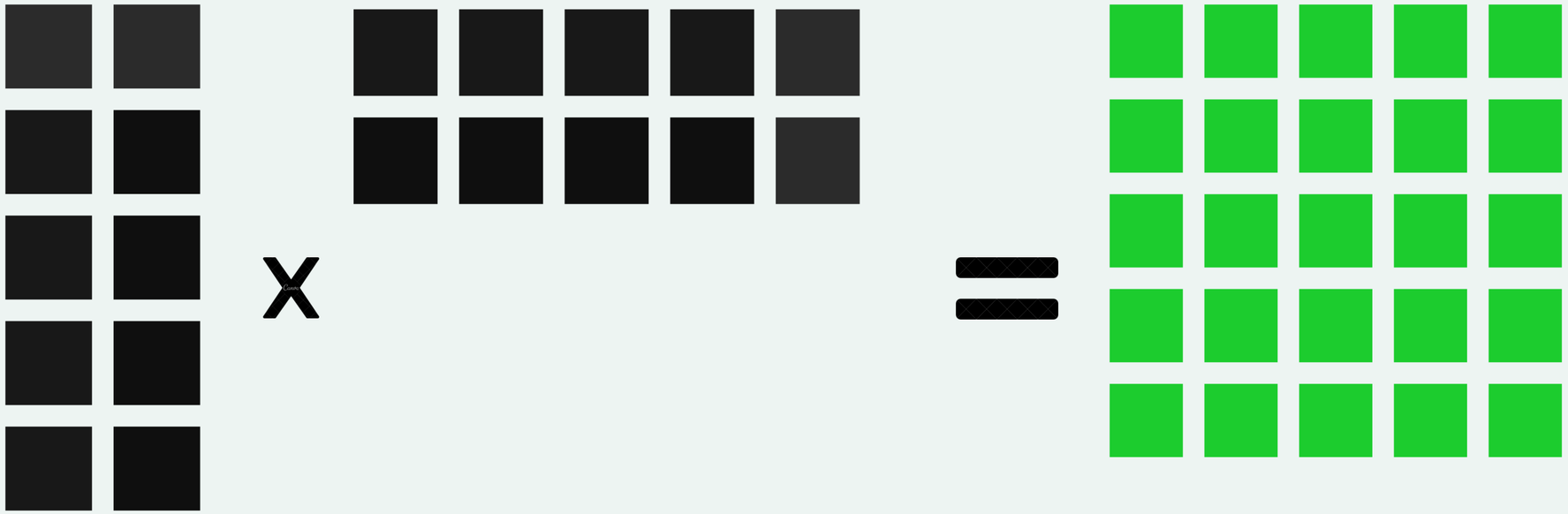


Figure 1: Our reparametrization. We only train A and B.

EXAMPLE



(5,2) (2,5) (5,5)

Instead of having 25 parameters for a 5x5 matrix, we can reduce the number of parameters to 20 by multiplying two matrices of sizes 5x2 and 2x5. The larger the original matrix, the more significant the optimization.

HYPERPARAMETERS

Working with LoRA, choosing the right hyperparameters is as important as any other step in the process. These are the most important ones:

Rank (r):

The rank parameter determines the rank of the low-rank decomposition matrices used in the LoRA approach. A higher rank increases the model capacity and precision but also increases the number of parameters.

This table show the percent of tatale parameter with diffirent ranks

RANK	7B	13B	70B	180B
1	0.00%	0.00%	0.00%	0.00%
2	0.01%	0.00%	0.00%	0.00%
8	0.02%	0.01%	0.01%	0.00%
16	0.04%	0.03%	0.01%	0.01%
512	1.22%	0.90%	0.39%	0.24%
1,024	2.45%	1.80%	0.77%	0.48%
8,192	19.58%	14.37%	6.19%	3.86%

Learning Rate (lr):

The learning rate controls how much the model weights are adjusted during each training step.

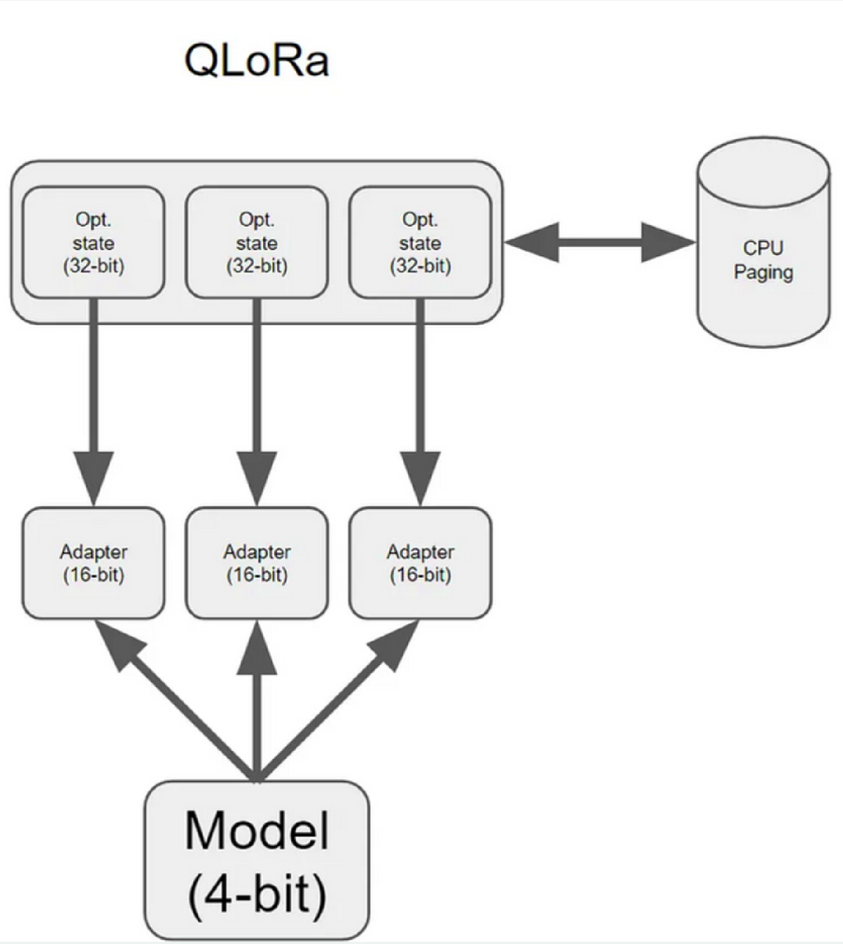
Alpha (α):

The alpha parameter in LoRA is a scaling factor applied to the low-rank matrices. It can control the contribution of the low-rank adaptation to the overall model. The alpha parameter defines how much the original model will be impacted by the new matrices.

QLoRA

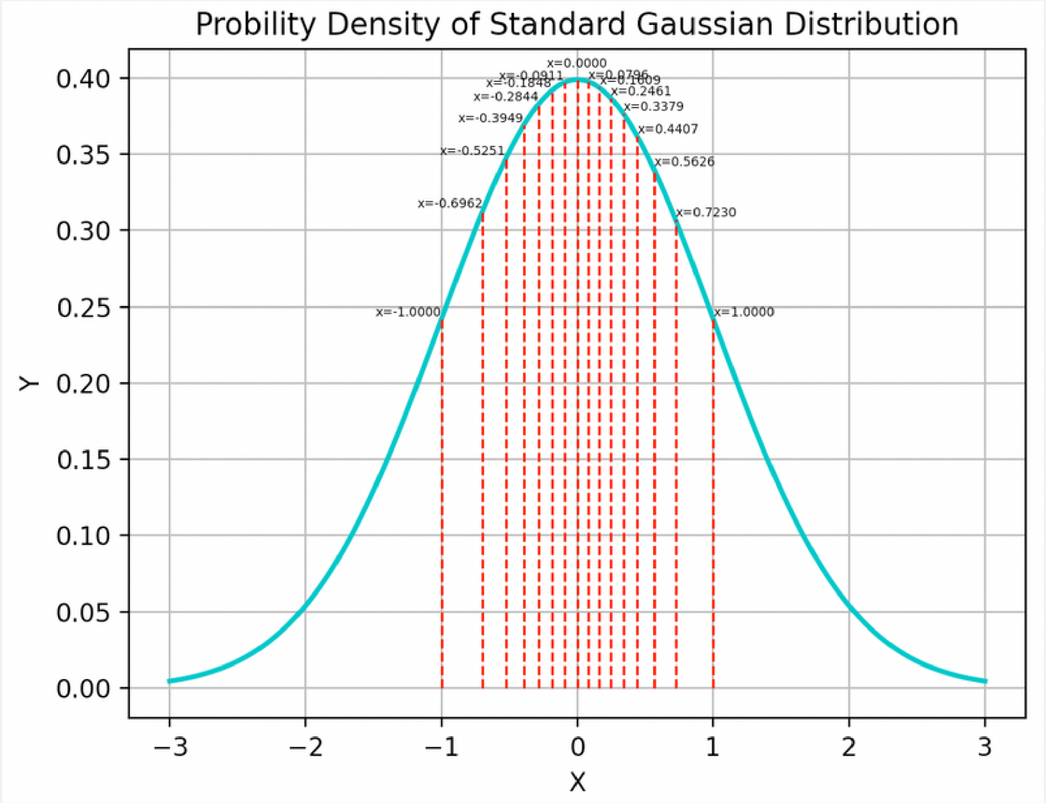
While both LoRA and QLoRA aim to make the fine-tuning of large models more efficient, QLoRA , Some refer to it as LoRA 2.0, takes it a step further by incorporating quantization to reduce resource requirements even more significantly.

Quantization involves reducing the precision of the model weights (e.g., from 32-bit to 8-bit or from 8-bit to 4-bit).



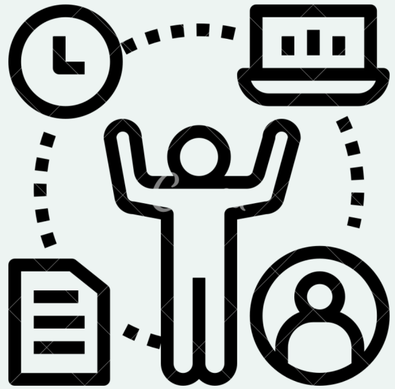
The question to ask is: Will I not lose model precision by doing this quantization step? I mean, if the weights are divided to half or more, will that make the model less efficient?

The answer is no. The way QLoRA is implemented allows you to recover the original model precision. Essentially, the process involves **compressing the parameters** during fine-tuning and then **decompressing** them. This is possible because the parameters generally fall within a **normal distribution**, so the reduced-precision weights can be **mapped back** to their original values using techniques that account for their distribution.



CONCLUSION

The use of LLM models has proven to be highly efficient, and fine-tuning has emerged as the best solution to leverage these models. Choosing the right technique and the appropriate hyperparameters is crucial and should be customized to the specific use case, data availability, resources available, and the level of investment you are willing to make in your model.



THANK YOU FOR YOUR ATTENTION