Fine-Tuning Pretrained Large Language Models

Training LLMs

Training for an LLM is a long and compute intensive task. The level of current models such as GPT5 with 1.8trillion. This initial training create a generalist model that struggles with specific information and queries. A more specifically trained model is needed to allow for accurate domain specific responses. Recreating the model for each task in this case is highly inefficient and would lead to bloated systems with many AIs. As every task would require its own fully trained model.

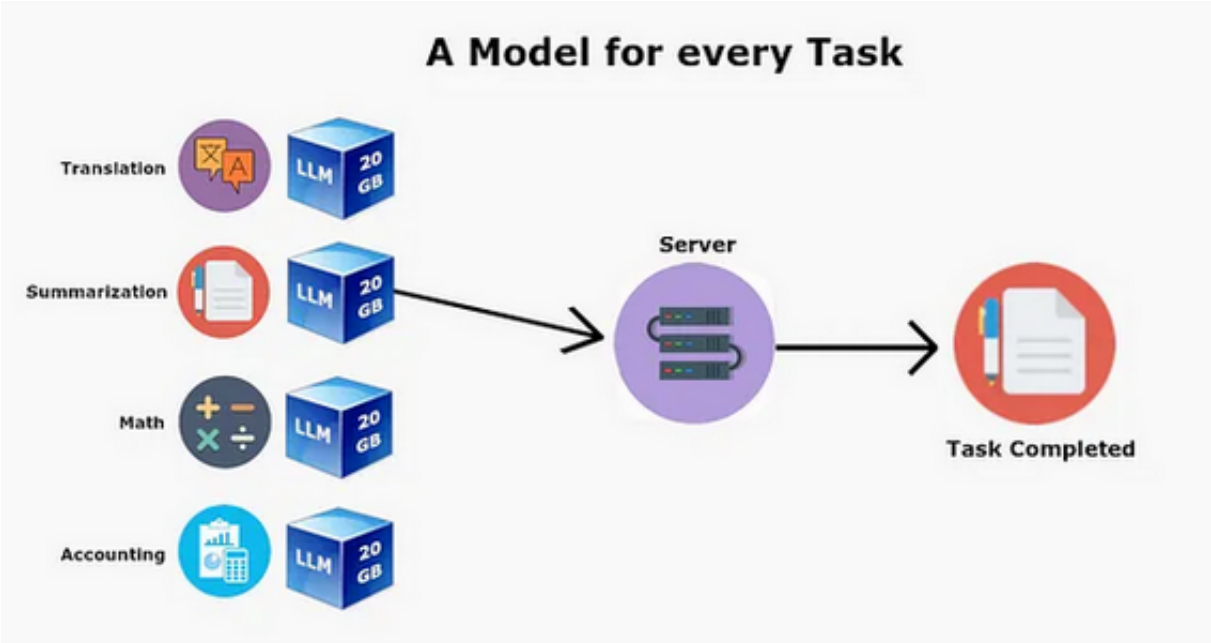

Figure 1: One model for every task

Adapter Architecture

This deficiency in base models opened the door for the adapter architecture to take place. This approach applies an additional layer to the AI adapter. This adapter is a created by freezing the current model weights, then training an additional set of weights to act upon the base model that allows the LLM to have its weights altered only by the adapter letting it be a plug and play solution, The matematical representation of this is at a high level:

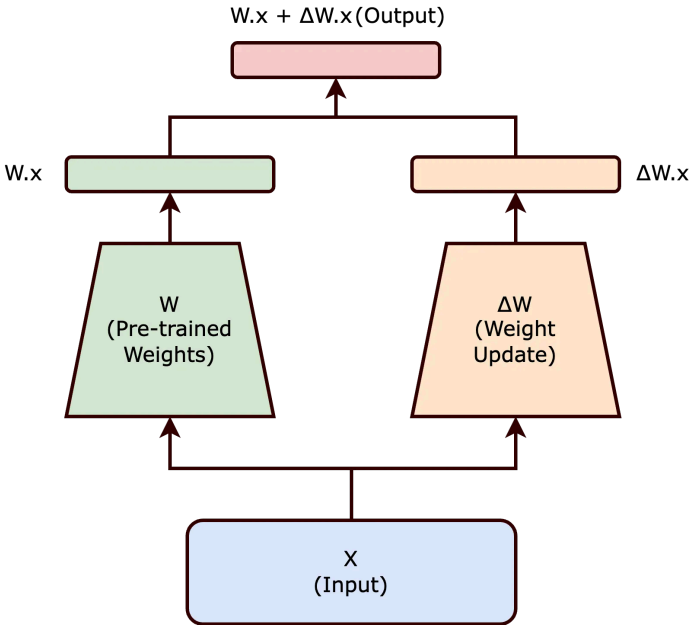

Figure 2: Fine tuning diagram

$$m \in L(D; W.x + \Delta W.x) \quad 1.$$

- L : This represents the loss function used to calculate the gradient that needs to be minimized for the LLM.
- D : This represents the dataset the loss function is being trained on and what is being optimized for.
- Θ : θ_0 represent the weights for base model and $\Delta\theta$ represent the weights from the fine tuning. This is how the adapter is swappable as the weights are not integrated into the base model.

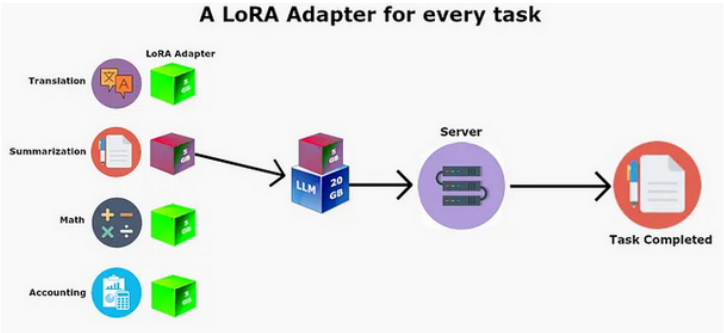

Figure 3: Adapter pattern

Fine Tuning approaches

Full Parameter Fine tuning

Full parameter fine tuning approach was first proposed in 2018 [1] called ULMFiT. This principle has been taken and applied in many forms to models such as DistilBERT[2] and BERT [3]. The base approach is is freezing the original weights then creating a blank matrix of the model weights, then training those to scale each weight individually to bias towards the new target. While efficent it is still a computationally heavy process as every single weight is modified but as a baseline it allows for the adapter architecture to be used.

The formula used to represent this would be

$$m \in L(D; W.x + \Delta W.x) \quad 2.$$

The size of matrix θ_0 and $\Delta\theta$ are both m * n where m and n represent the rows and columns in θ_0. The rest of the definitions are here Equation 1. This method of training allows a small dataset to impact the results of a larger model removing the need to train the model on a huge corpus of data to get tangible results.

Lora Fine Tuning

Vera Fine Tuning

Vera [4] fine tuning is an innovation on LoRa fine tuning created to reduce the memory overhead in LoRa. The method in which it works is based on Random Matrix Adaptation and LoRa. The process begins by generating two low rank matrixes of sizes m * In mathematical terms

$$W.x + \Delta W.x = W.x + \Lambda_b B \Lambda_d A x \quad 3.$$

- A and B: Are randomly generated low rank matrixes of sizes m * r and r * n which multiply to create the W.x matrix.
- Λ_b and Λ_d: Are diagonal matrixes which are used to scale the A and B matrixes. They are of sizes m * m and r * r.

The key innovation to note is the $\Lambda_b B \Lambda_d A x$. These are diagonal scaling matrixes which scale the values of the randomly gernated A and B matrixes which emulate

QLora Fine Tuning

Lora vs Full fine tuning

Lora Vs Vera

Bibliography

[1] J. Howard and S. Ruder, "Universal Language Model Fine-tuning for Text Classification." Accessed: Oct. 22, 2025. [Online]. Available: http://arxiv.org/abs/1801.06146

[2] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." Accessed: Oct. 22, 2025. [Online]. Available: http://arxiv.org/abs/1910.01108

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Accessed: Oct. 22, 2025. [Online]. Available: http://arxiv.org/abs/1810.04805

[4] D. J. Kopiczko, T. Blankevoort, and Y. M. Asano, "VeRA: Vector-based Random Matrix Adaptation." Accessed: Oct. 22, 2025. [Online]. Available: http://arxiv.org/abs/2310.11454