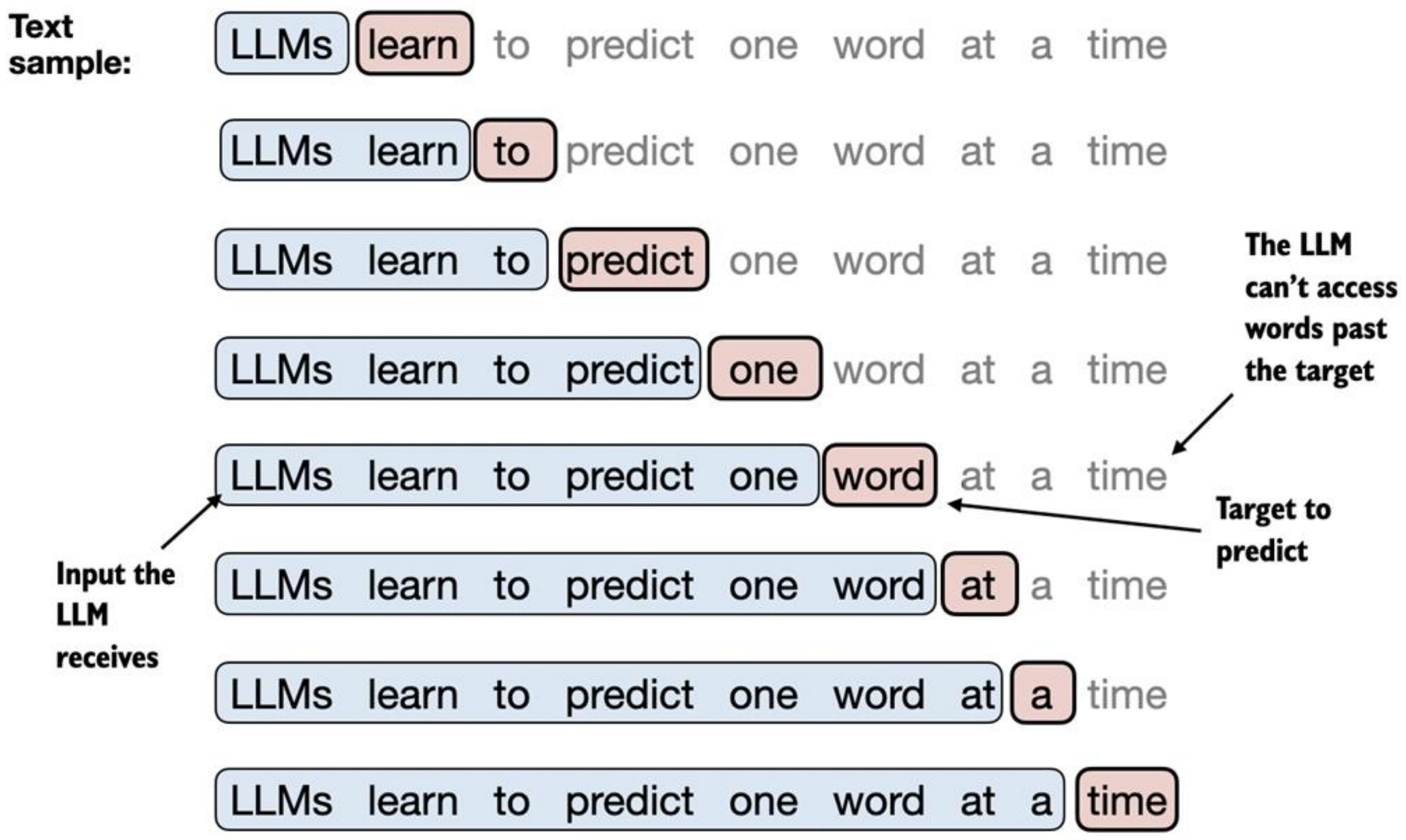
LLM Finetune with

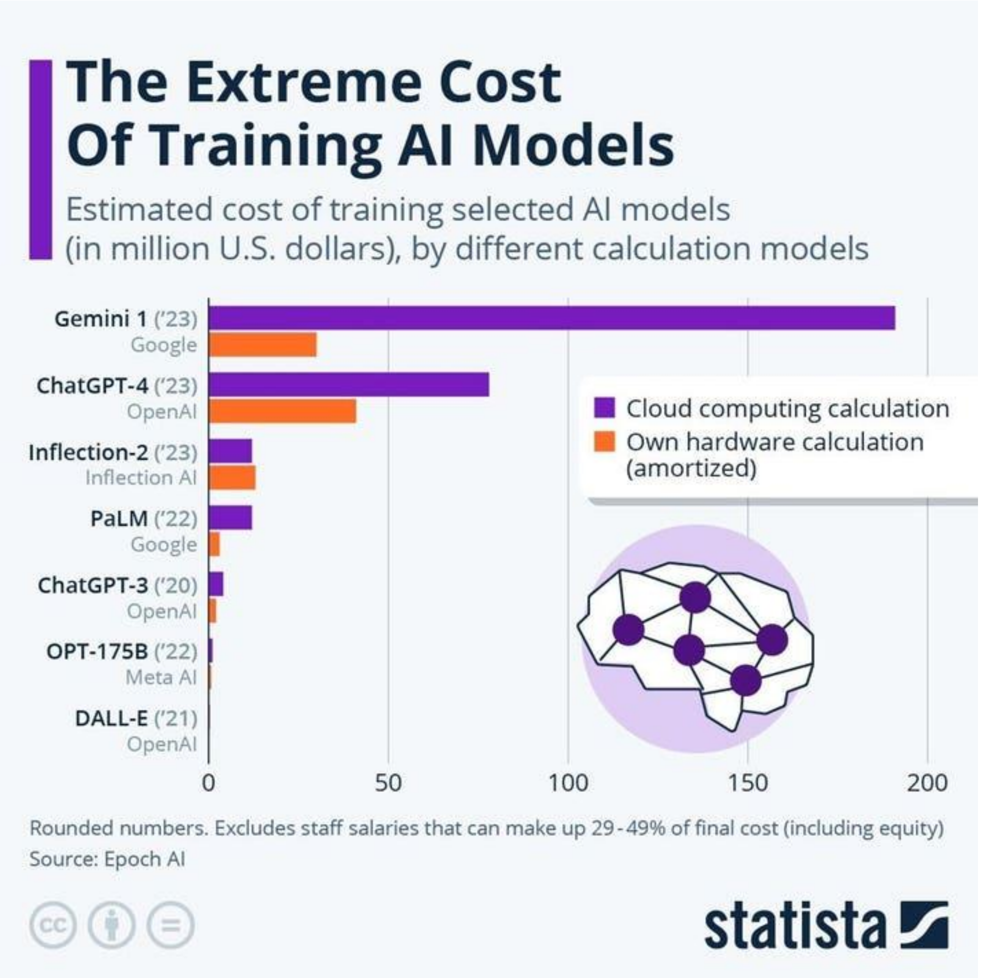
unsloth

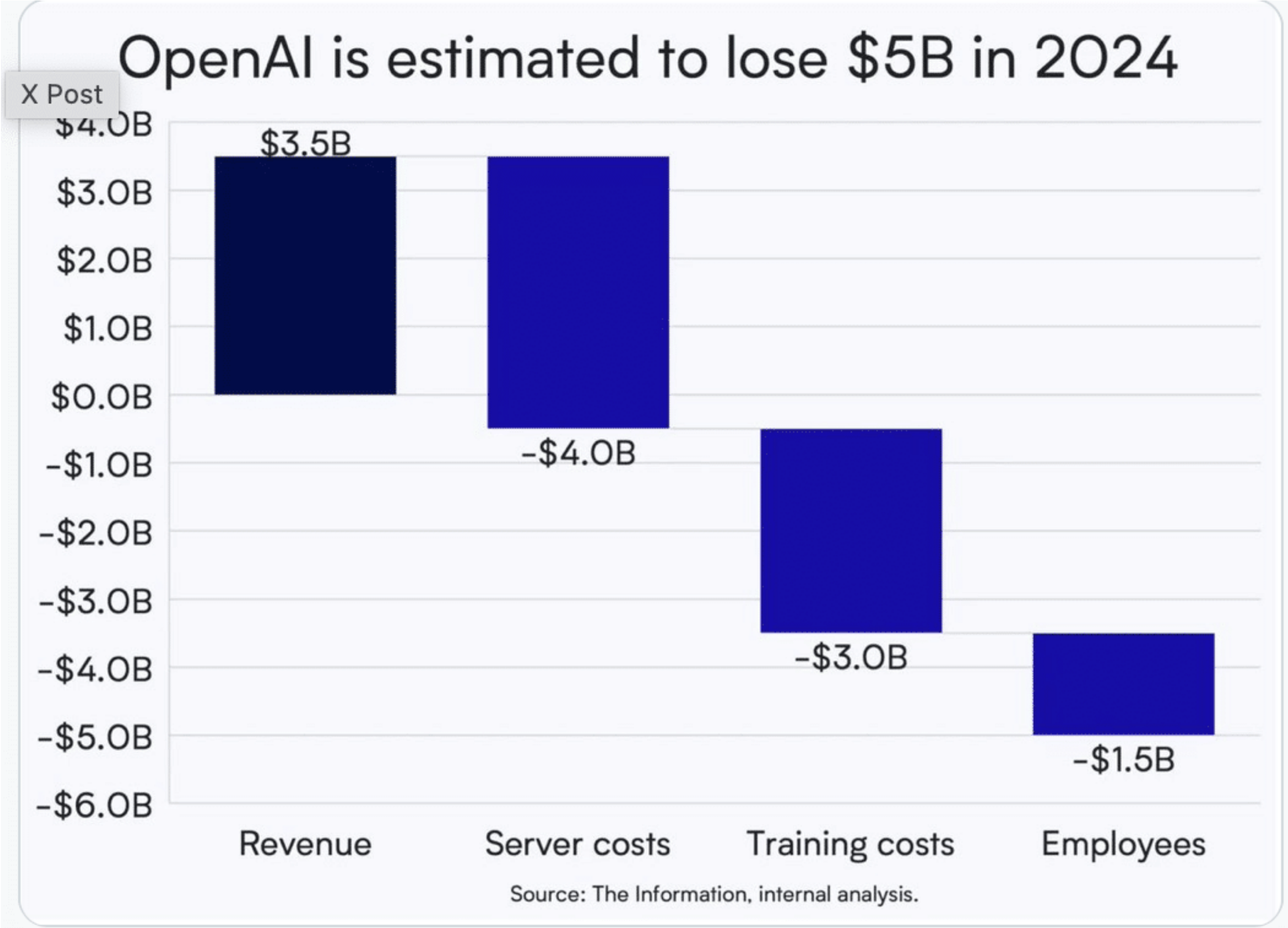
Why need this?

- Make response better
- Add more knowledge



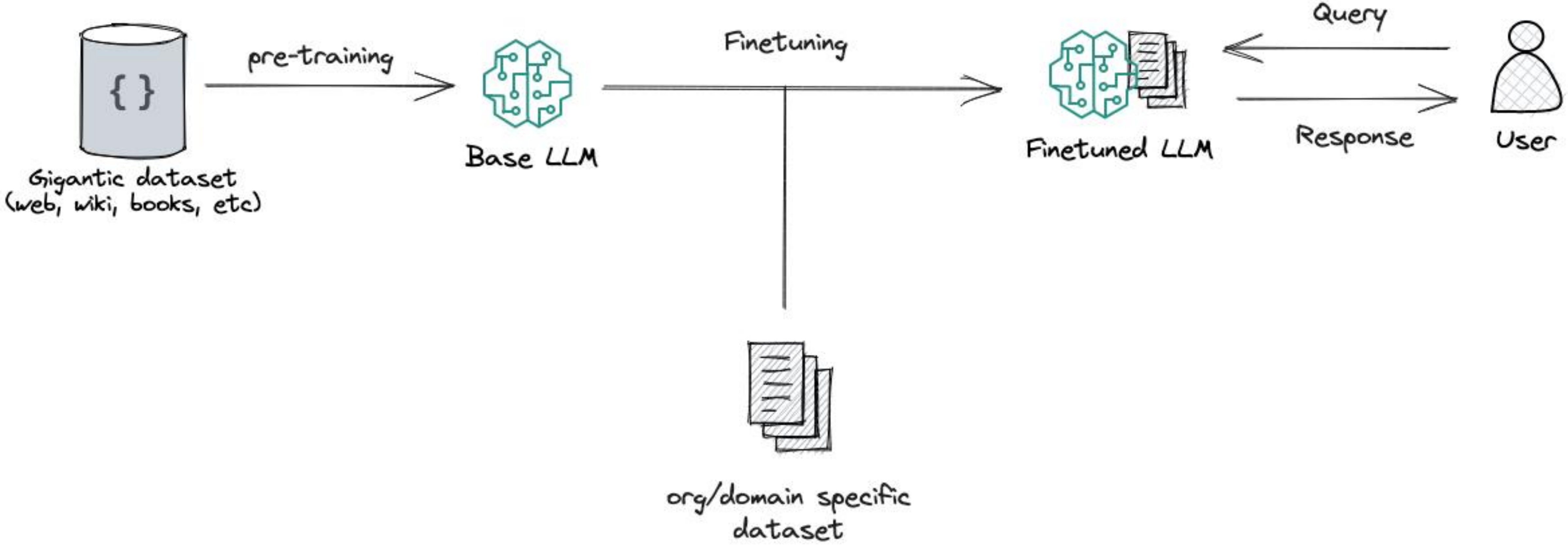
Can we train LLM?





Understand Fine-tuning

- Fine-tuning an LLM customizes its behavior, enhances + injects knowledge, and optimizes performance for domains/specific tasks. For example:
  - GPT-4 serves as a base model; however, OpenAI fine-tuned it to better comprehend instructions and prompts, leading to the creation of ChatGPT-4 which everyone uses today.
  - DeepSeek-R1-Distill-Llama-8B is a fine-tuned version of Llama-3.1-8B. DeepSeek utilized data generated by DeepSeek-R1, to fine-tune Llama-3.1-8B. This process, known as distillation (a subcategory of fine-tuning), injects the data into the Llama model to learn reasoning capabilities.


pre-training
Base LLM
Finetuning
Finetuned LLM
Query
Response
User
Gigantic dataset
(web, wiki, books, etc)
org/domain specific
dataset

Base Model

(davinci (GPT-3))

Prompt: Tell me how to fine-tune a model

Completion:
How can I control the complexity of a model?
How do I know when my model is done?
How do I test a model?
How do I define a task for my model?
How do I scale a model?
How do I make a probabilistic model?
How do I start using a model once it's complete?
How do I cope with missing data?
How do I make my modeling efforts sustainable?

Fine-tuned Model

(text-davinci-003)

Prompt: Tell me how to fine-tune a model

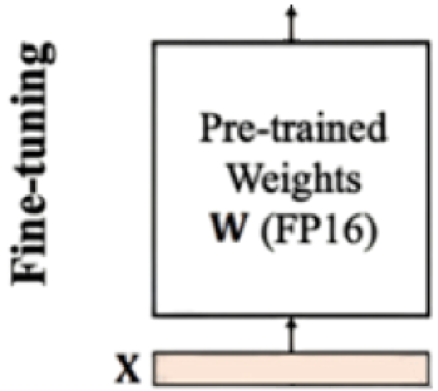
Completion:
Fine-tuning a model involves adjusting the parameters of a pre-trained model in order to make it better suited for a given task. There are generally three steps to fine-tuning a model:

1. Select a base model - Select a pre-trained deep learning model that has been trained on a large dataset.
2. Adjust parameters - Adjust parameters of the pre-trained model to better suit the desired task. This may include changing the number of layers, adjusting learning rate, adding regularization, or tweaking the optimizer.
3. Train the model - Train the new model on the desired dataset. The amount of data and the amount of training required will depend on the task and the model.

Lora & QLora

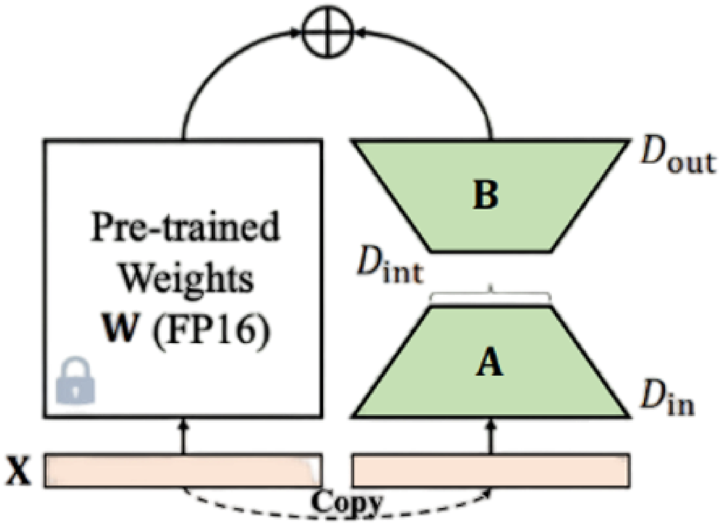
- LoRA: Fine-tunes small, trainable matrices in 16-bit without updating all model weights.
- QLoRA: Combines LoRA with 4-bit quantization to handle very large models with minimal resources.

Full Fine-Tuning
16-bit precision



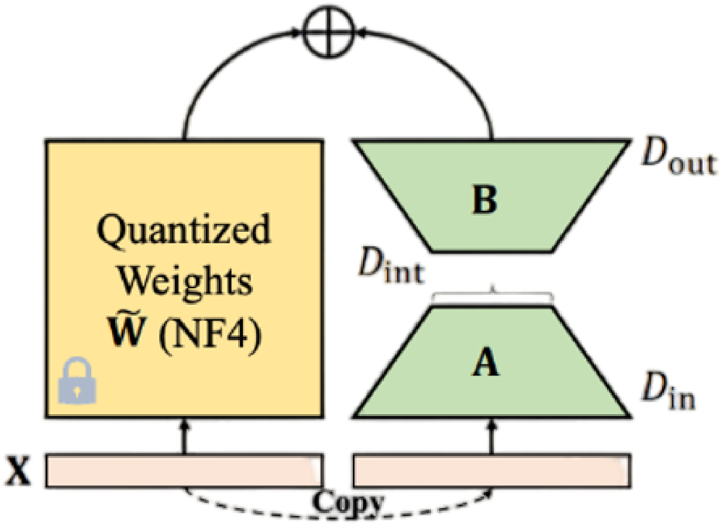
✅ Best performance
❌ Very high VRAM usage

LoRA
16-bit precision



✅ Quick training
❌ Still costly

QLoRA
4-bit precision



✅ Low VRAM usage
❌ Degrades performance

What is unsloth?

Easily **finetune & train LLMs**
Get *faster* **with unsloth**

Unsloth is not just a library; it's a technological symphony orchestrated for the fine-tuning and training of large language models (LLMs).

Pros of Unsloth

Speed Redefined : Unsloth boasts a staggering 30x increase in training speed. Alpaca, a benchmark task, now takes merely 3 hours instead of the conventional 85. This acceleration is a testament to Unsloth's commitment to efficiency and productivity.

Memory Efficiency: A game-changer in the memory domain, Unsloth promises a 60% reduction in memory usage. This not only enables the handling of larger batches but also ensures a seamless fine-tuning process without compromising on performance.

Accuracy Amplified: The authors proudly declare a 0% loss in accuracy, with an additional option for a +20% increase in accuracy using their MAX offering. This commitment to maintaining and elevating accuracy levels sets Unsloth apart in the competitive landscape.

Hardware Compatibility: Unsloth extends its reach by supporting NVIDIA, Intel, and AMD GPUs. This inclusivity ensures accessibility to a wide array of hardware configurations, making it a versatile choice for developers across different platforms.

Step to Finetune

1. Choose the Right Model + Method
2. Prepare Your Dataset: You will need to create a dataset usually with 2 columns - question and answer. The quality and amount will largely reflect the end result of your fine-tune so it's imperative to get this part right.
3. Select training parameters
4. Train, save weight.
5. Test the finetuned model.

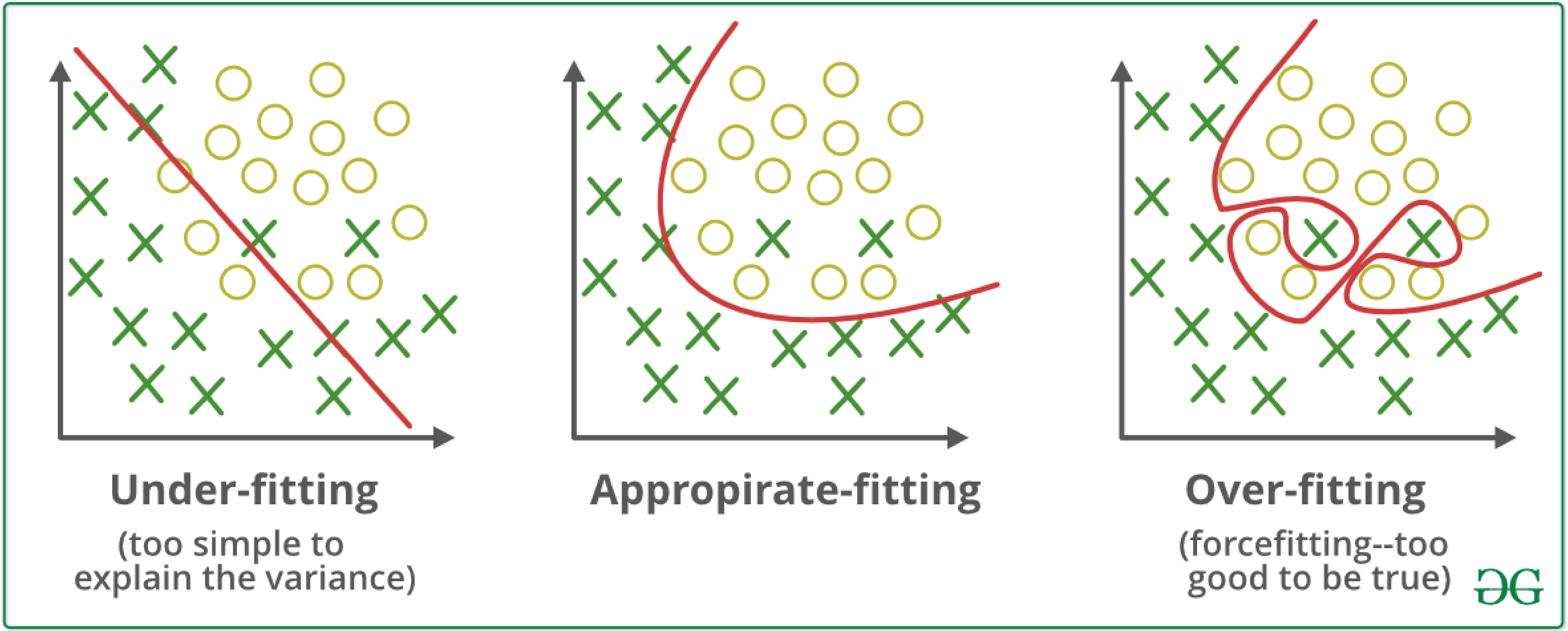
Step to Finetune

```
model = FastLanguageModel.get_peft_model(
    model,
    r = 16, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none",    # Supports any, but = "none" is optimized
    # [NEW] "unsloth" uses 30% less VRAM, fits 2x larger batch sizes!
    use_gradient_checkpointing = "unsloth", # True or "unsloth" for very long context
    random_state = 3407,
    use_rslora = False,  # We support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)
```

Understand Model Parameters:

- Leave almost alone
- Understand some of them:
    - Learning rate:
        - Higher Learning Rates: Faster training, reduces overfitting just make sure to not make it too high as it will overfit
        - Lower Learning Rates: More stable training, may require more epochs.
        - Typical Range: 1e-4 (0.0001) to 5e-5 (0.00005).
    - Epochs: 1-3 to avoid overfitting.

Overfitting vs Underfitting



Train

- **per_device_train_batch_size = 2** – Increase for better GPU utilization but beware of slower training due to padding. Instead, increase gradient_accumulation_steps for smoother training.

- **gradient_accumulation_steps = 4** – Simulates a larger batch size without increasing memory usage.

- **max_steps = 60** – Speeds up training. For full runs, replace with num_train_epochs = 1 (1–3 epochs recommended to avoid overfitting).

- **learning_rate = 2e-4** – Lower for slower but more precise fine-tuning. Try values like 1e-4, 5e-5, or 2e-5.

Test

- In order to evaluate, you could do manually evaluation by just chatting with the model and see if it's to your liking.
- You can also enable evaluation for Unsloth, but keep in mind it can be time-consuming depending on the dataset size.
- To speed up evaluation you can: reduce the evaluation dataset size or set **evaluation_steps = 100**.

Handson

- Neet GPU >= 24GB VRAM
- Linux: Ubuntu