Module 5:

Fine-Tuning LLMs: Unleashing Their Power

Content

- ❖ Introduction to LLM fine-tuning
- ❖ Stages in the Development of Modern LLMs
- ❖ Supervised Fine-tuning, Preference Fine-tuning (RLHF, DPO)
- ❖ Fine-Tuning With Limited Computing Resources (QLORA, Quantization)
- ❖ Hands-on experience

Introduction to fine-tuning LLMs

Fine-tuning: process of turning general-purpose models into specialized models.

joint pain, skin rash, and
sun sensitivity

↓

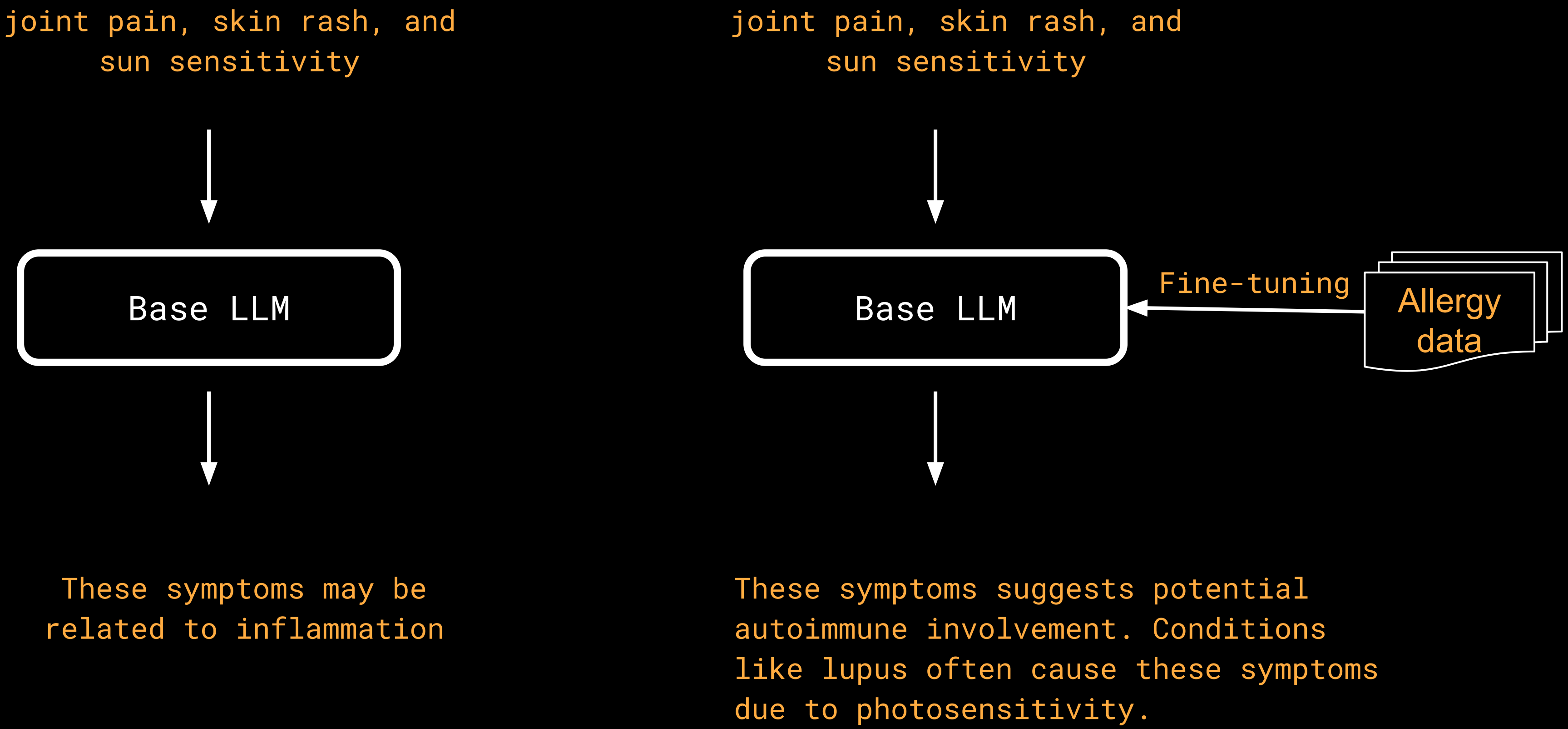
Base LLM

↓

These symptoms may be
related to inflammation

Fine-tuning: process of turning general-purpose models into specialized models.

Fine-tuning: process of turning general-purpose models into specialized models.



Fine-tuning LLMs

Fine-tuning involves adjusting the parameters of a pre-trained LLM using a smaller, task-specific dataset.

The goal is to customize the model for specific language patterns and vocabulary related to a particular task.

Benefits of fine-tuning

AI TRAINING

- Specificity and Relevance: Fine-tuning ensures LLMs understand industry-specific terms and generate relevant content.
- Improved Accuracy: Domain-specific fine-tuning enhances precision, aligning model outputs with expectations.
- Customized Interactions: Tailoring LLM responses maintains brand consistency and user experience.

Benefits of fine-tuning

- Data Privacy and Security: Fine-tuning controls exposure to sensitive data, preventing inadvertent leaks.

- Addressing Rare Scenarios: Fine-tuning optimally handles unique business challenges.

Stages in the Development of Modern Foundation Models

Development of Modern Foundation Models

Base LLM

The benefits of regular exercise

↓

Pretrained LLM

↓

The benefits of regular exercise are well-documented. Exercise can improve mood, boost energy,

Development of Modern Foundation Models

Base LLM

↓

Instruction-tuned LLM

The benefits of regular exercise

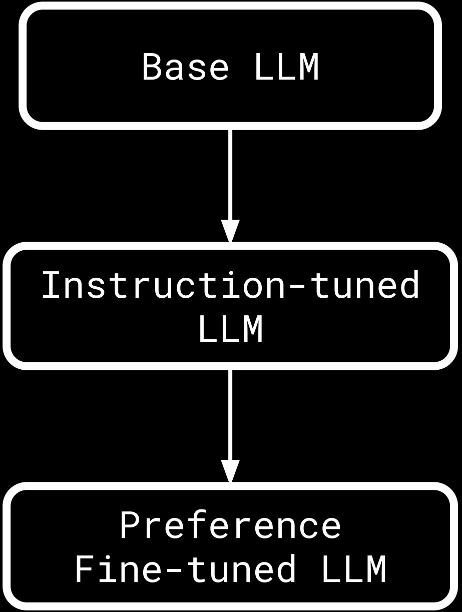
↓

Instruct-tuned LLM

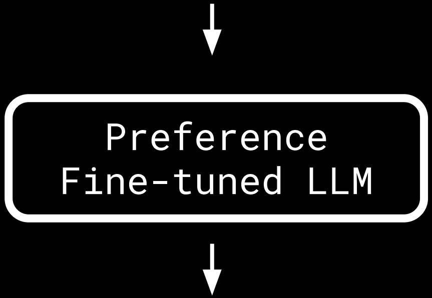
↓

Regular exercise offers numerous benefits for both physical and mental health. Here are some of the main advantages: Weight Management: Regular exercise helps in maintaining a healthy weight by burning calories and boosting metabolism improved Mood: ns

Development of Modern Foundation Models



The benefits of regular exercise



Regular exercise offers a multitude of advantages: Weight management: Burn calories and maintain a healthy weight. Disease prevention: Reduce risks of heart disease, stroke, diabetes, and certain cancers.

Mood enhancement: Boost happiness..

Pretraining LLM

Self-Supervised Setting:

The model predicts the next word in a given context.

Loss Function:

Cross Entropy Loss

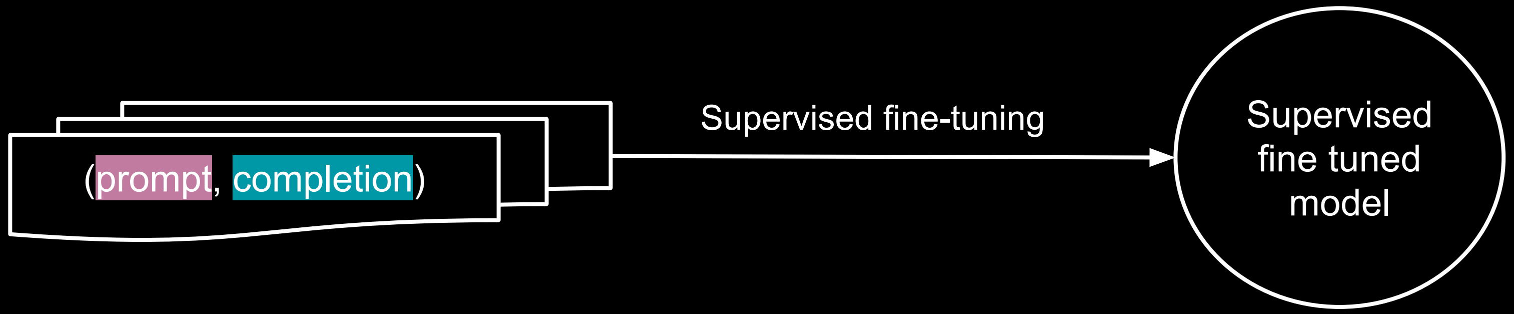
Data:

LLMs are pre-trained on vast corpora from the internet.

This diverse data source enables the model to capture general language patterns and understanding

Supervised fine-tuning

- Purpose: Enable an LLM to perform specific tasks
- Requirements: Large dataset of prompts and correct answers for the task
- Process: maximise the likelihood of tokens in the answers
- Cross Entropy Loss



Preference fine-tuning

AI TRAINING

- Purpose: adjust the LLM to better reflect the preferences from a comparison dataset
- Relevance:

  useful for critiquing LLM-generated answers

  Faster and easier than manually writing adequate answers.

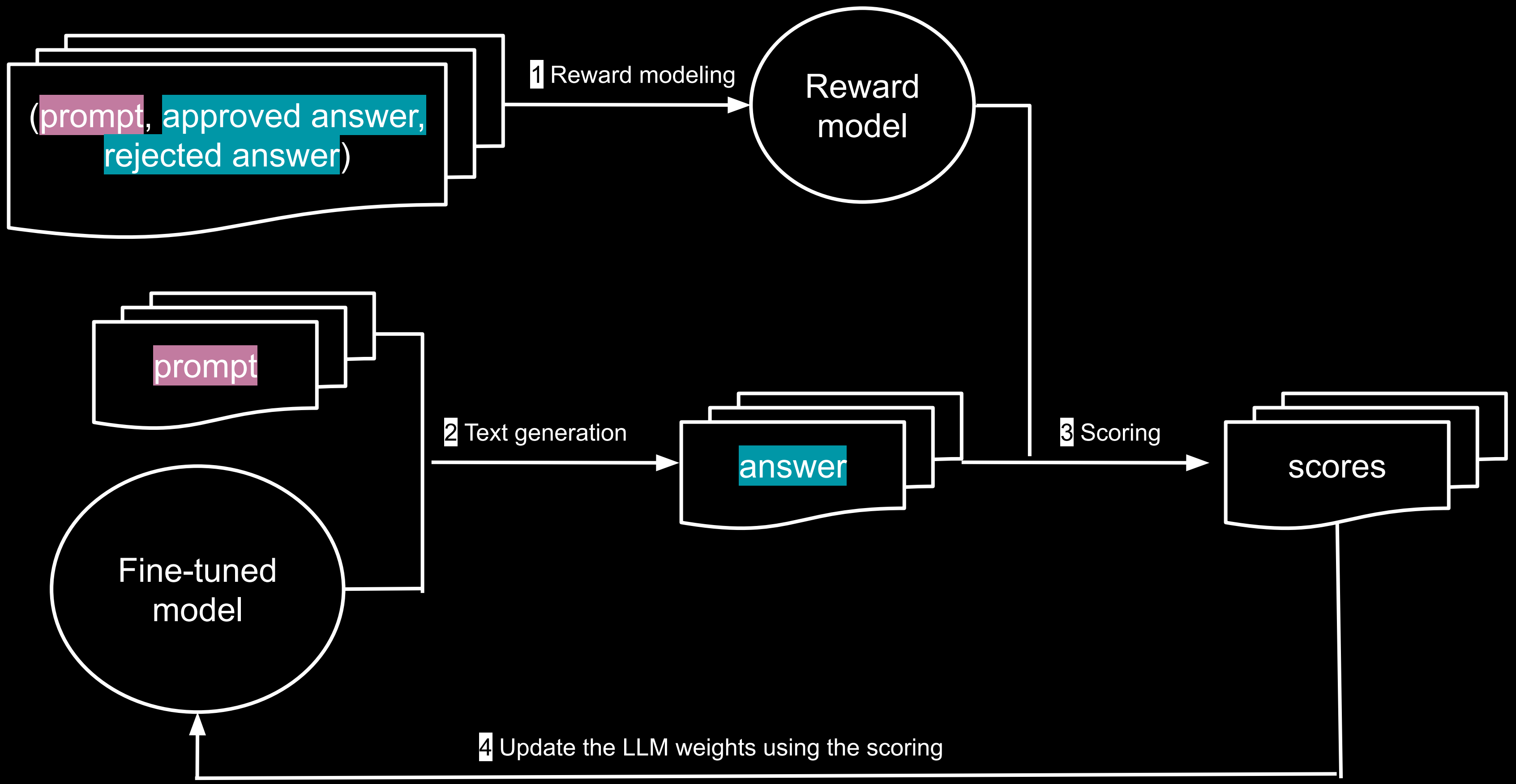
- Comparison dataset: contains prompts, approved answers and rejected answers

- Fine-tuning methods

  Reinforcement Learning with Human Feedback (RLHF)

  Direct Preference Optimization (DPO)

Reinforcement Learning with Human Feedback



Reinforcement Learning with Human Feedback

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)} \boxed{\left[r_\phi(x, y) \right]} - \boxed{\beta \mathbb{D}_{\mathrm{KL}} \left[\pi_\theta(y \mid x) \mid\mid \pi_{\mathrm{ref}}(y \mid x) \right]}$$

maximise rewards

use KL-divergence penalty to prevent **reward hacking** (controlled by β)

Reinforcement Learning with Human Feedback

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \boxed{\left[r_\phi(x, y)\right]} - \boxed{\beta\mathbb{D}_{\text{KL}}\left[\pi_\theta(y \mid x) \mid\mid \pi_{\text{ref}}(y \mid x)\right]}$$

maximise rewards

use KL-divergence penalty to prevent **reward hacking** (controlled by β)

- Various challenges

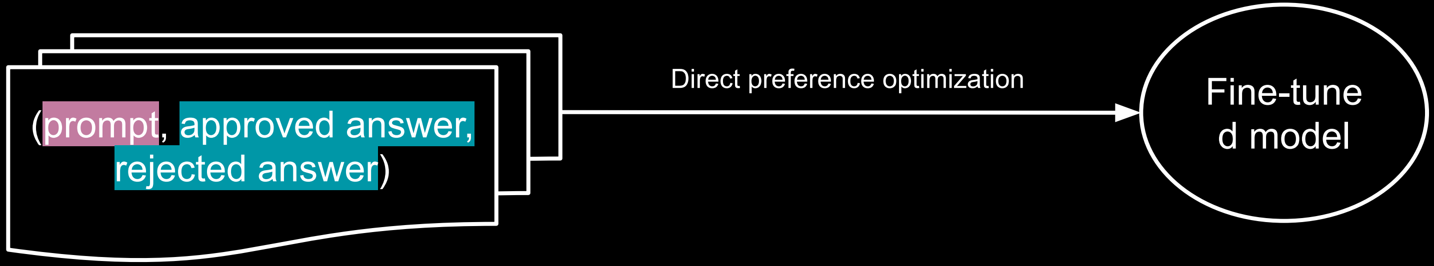
  RL notoriously unstable, many hyperparameters

  Need a separate RM ⇒ 3 LLMs to jungle

Direct Preference Optimization

It solves the same problem by minimizing a training loss directly based on the preference data (without reward modeling or reinforcement learning)

Simpler and more stable alternative



Direct Preference Optimization

$$\max_{\pi} \mathbb{E}_{(x,y_w,y_l)\sim D} \log \sigma\left(\beta \log \frac{\pi(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

Rafailov and al. (2023)

Direct Preference Optimization

$$\max_{\pi} \mathop{\mathbb{E}}_{(x,y_w,y_l)\sim D}$$

Good response

bad response

Direct Preference Optimization

$$\log \frac{\pi(y_w|x)}{\pi_{\text{ref}}(y_w|x)}$$

Model we’re optimizing

Reference model (SFT)

Good response

Direct Preference Optimization

$$\log \frac{\pi(y_l|x)}{\pi_{\text{ref}}(y_l|x)}$$

Model we're optimizing

Reference model (SFT)

Bad response

Direct Preference Optimization

$$\max_{\pi} \mathbb{E}_{(x,y_w,y_l)\sim D} \log \sigma \left(\beta \log \frac{\pi(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

```python
import torch.nn.functional as F

def dpo_loss(pi_logps, ref_logps, yw_idxs, yl_idxs, beta):
    pi_yw_logps,  pi_yl_logps =  pi_logps[yw_idxs],  pi_logps[yl_idxs]
    ref_yw_logps, ref_yl_logps = ref_logps[yw_idxs], ref_logps[yl_idxs]
    pi_logratios  = pi_yw_logps - pi_yl_logps
    ref_logratios = ref_yw_logps - ref_yl_logps
    losses = -F.logsigmoid(beta * (pi_logratios - ref_logratios))
    rewards = beta * (pi_logps - ref_logps).detach()
    return losses, rewards
```

Algorithm
- Sample good/bad response
- Run pairs through 2 models (active and reference)
- Backpropagation

Efficient fine-tuning

Vanilla fine-tuning of LLMs

- It refers to the process of adjusting all parameters of a pre-trained model when further training it on a new dataset.

- Challenges
    - Parameter Count
    - Gradient computation and storing (memory-intensive)
    - Computationally expensive and time-consuming.

Parameters-Efficient Fine-Tuning (PEFT)

AI TRAINING

PEFT Paradigm:

Address the challenges of vanilla fine-tuning.

Target Parameters:

PEFT identifies specific layers or parameters that significantly impact the task performance.

Freezing Others:

The rest of the learned parameters (non-target parameters) are frozen.

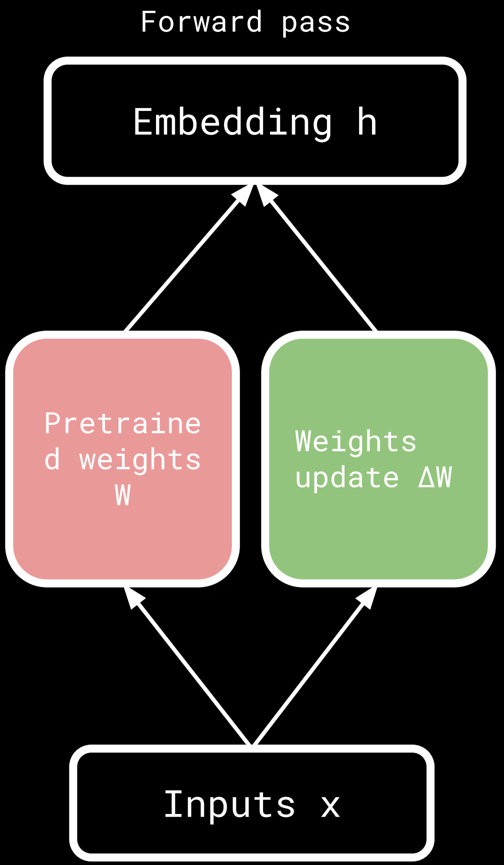
Low Rank Adaptation : LoRA

Fine-tuning: leverages general knowledge

From a pretrained model

- 405 B parameters for Llama 3.1

Fine-tuning ⇒ memory intensive



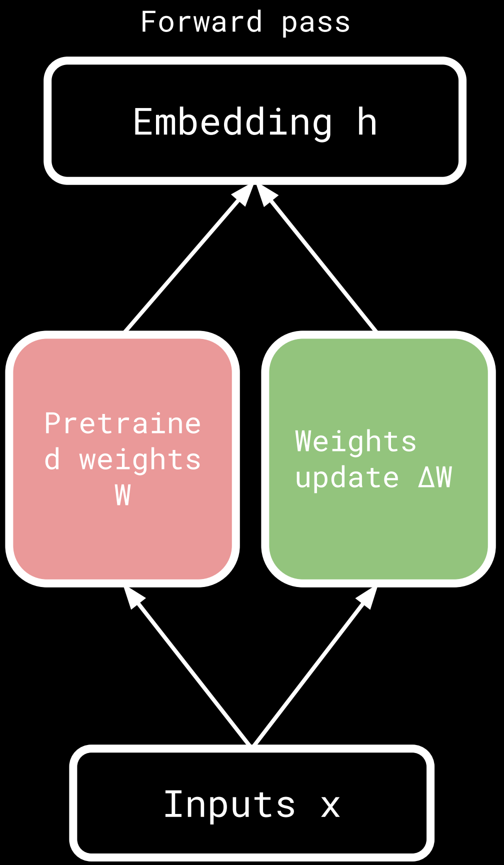
Low Rank Adaptation : LoRA

Fine-tuning: leverages general knowledge

From a pretrained model

- 405 B parameters for Llama 3.1

Fine-tuning ⇒ memory intensive

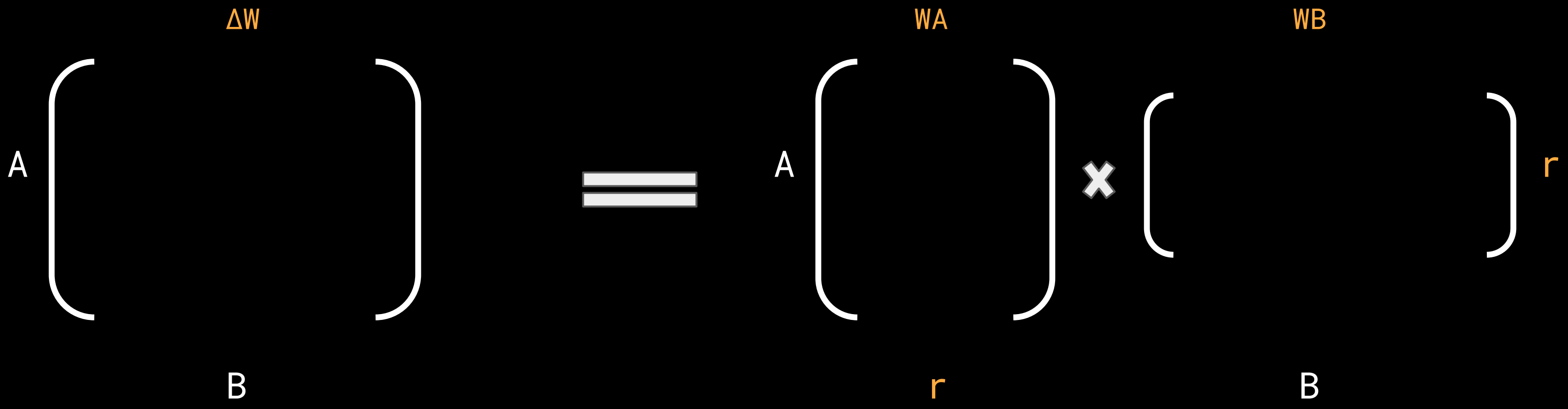


$h = (w + \Delta w)x$

Low Rank Adaptation : LORA

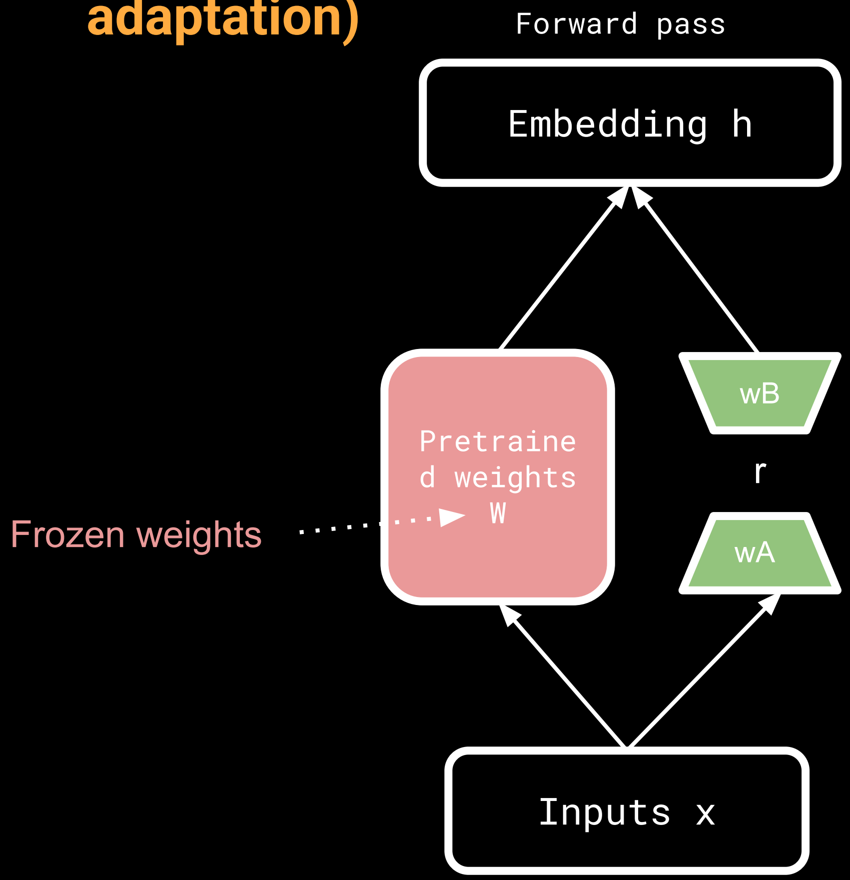
Hu et al. (2021)

- During fine-tuning, ΔW has a low rank and can be decomposed as



- r is hyper-parameter that we need to tune

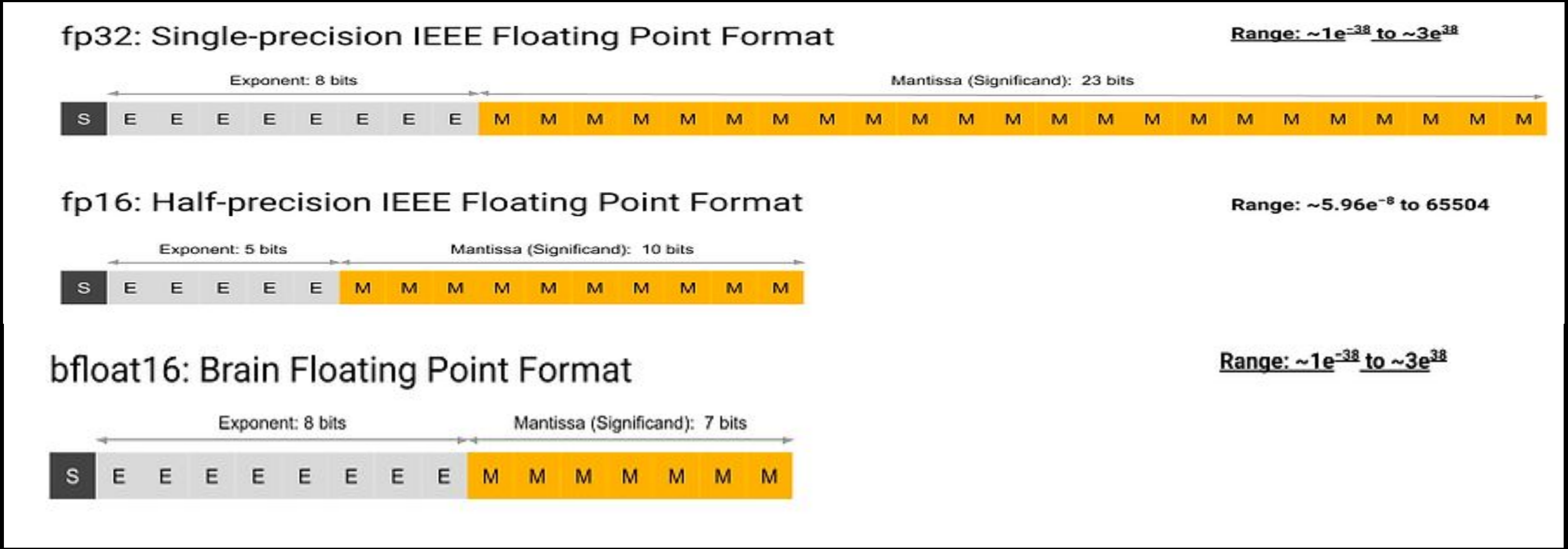
Fine-Tuning With Limited Computing Resources: LORA (Low rank adaptation)



h = (w+wA*wB)x

Quantization

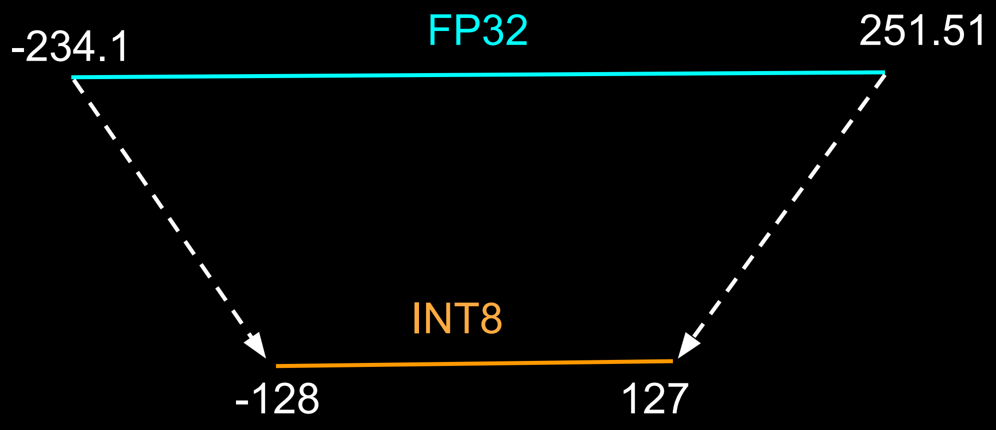
Floating-Point Representation



Floating point formats| Google

Quantization

Quantization refers to the process of mapping input values from a large set (often continuous) to output values in a smaller set, often with a finite number of elements.



Quantization

Zero point quantization

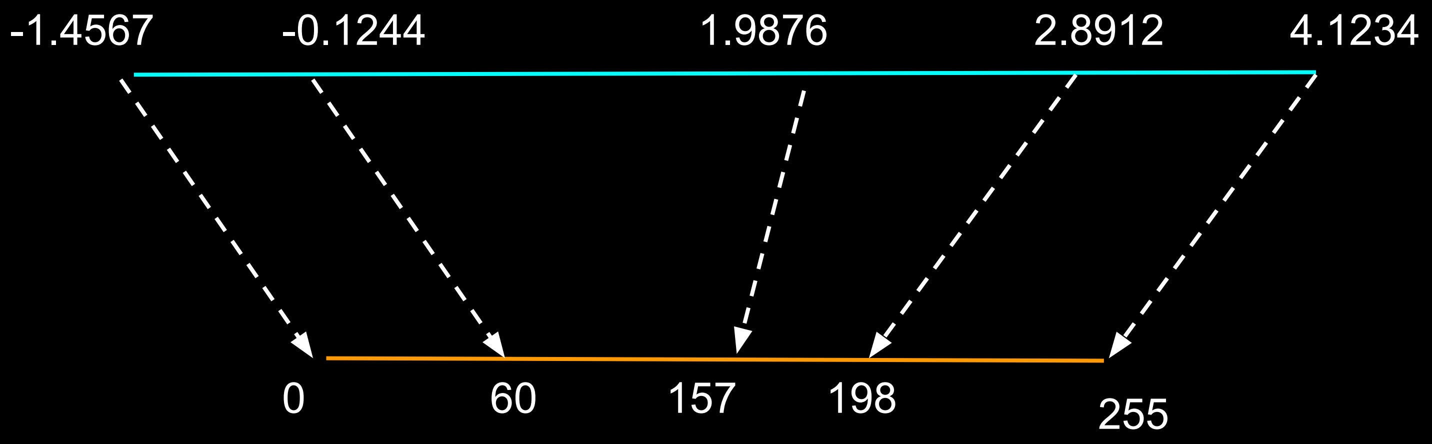
Let consider	2.8912	-0.1244	4.1234	1.9876	-1.4567

Quantization

Zero point quantization

Let consider 2.8912 -0.1244 4.1234 1.9876 -1.4567

Max min



198 60 255 157 0

Quantized LoRA: QLoRA

Dettmers et al (2023)

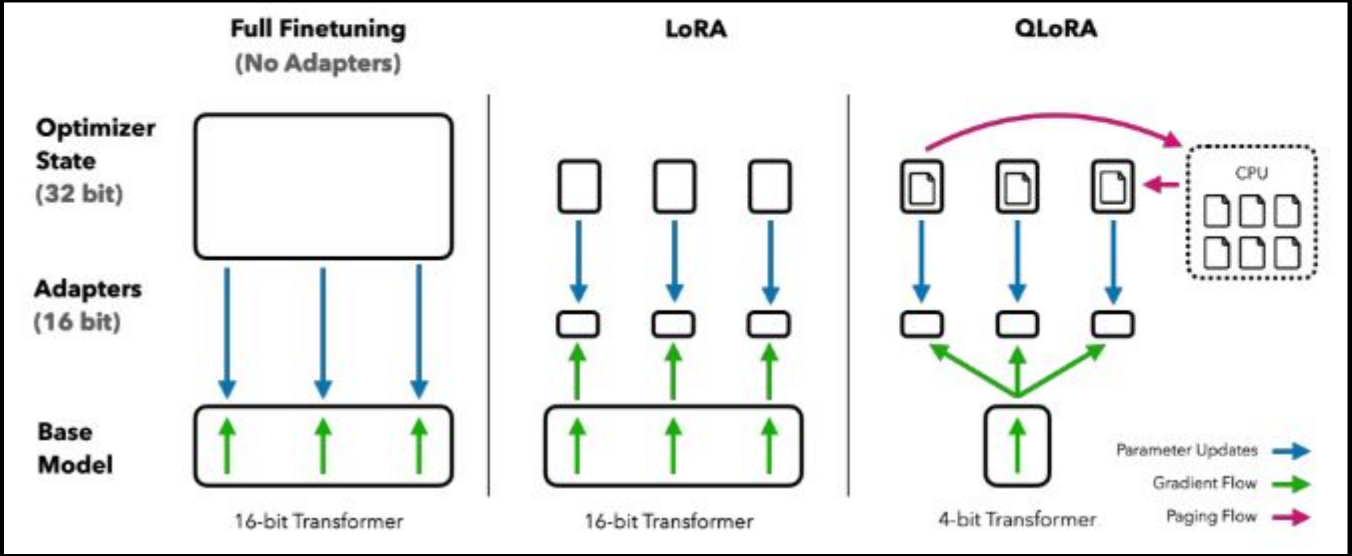
QLoRA: efficient fine-tuning approach designed to reduce memory usage during the fine-tuning of LLMs.

It enables fine-tuning of a 65B parameter model on a single 48GB GPU while maintaining full 16-bit fine-tuning task performance.

The approach involves back-propagating gradients through a frozen, 4-bit quantized pretrained language model into Low Rank Adapters (LoRA).

Fine-Tuning With Limited Computing Resources: QLORA (Quantized LORA)

Dettmers et al (2023)



Fine-Tuning Best Practices

AI TRAINING

Fine-Tuning Best Practices

- Clearly Define Your Task

    Foundational Step: Begin by defining your specific task.

    Focus and Direction: Clear task definition channels the model’s capabilities toward a specific goal.

    Performance Benchmarks: Set measurable benchmarks for evaluating model performance.

- Leveraging Pre-Trained Models

    Efficiency and Understanding: Pre-training captures general language understanding.

    Model Architecture Matters: Choose the right architecture (e.g., MoE, MoT) for effective fine-tuning.

Fine-Tuning Best Practices

Set Hyperparameters

Tunable Variables: Hyperparameters (e.g., learning rate, batch size, weight decay) impact model training.

Optimal Configuration: Experiment to find the best hyperparameter values for your specific task.

Iterative Refinement: Continuously evaluate and adjust hyperparameters during fine-tuning.

Evaluate Model Performance

Unbiased Assessment: Evaluate the fine-tuned model on a separate test set.

Generalization: Assess how well the model performs on unseen data.

Refinement Opportunity: If performance can be improved, consider further iterations.