Finetuning

GPT → 3.5 turbo ← OpenAI API

Llama, Phi ← microsoft

⇒ modular coding

Some theory points on LoRA, PEFT, Quanti-
-zation

⇒ Reward learning (RLHF, Preference optimization) ← Unsloth

Ready made ⇒ Llama factory, unsloth

RAG, Agent ✓ the most imp ⇒ Cloud ↓ ⇒ OPS

App, project

↓

Langchain, vector DB OLLAMA

Langgraph, Groq

CrewAI, Streamlit

(MARI APP)

1 Pretraining 2 SFT 3 RLHF

↓

Supervised data

IIP OIP

LLM ⇒ Challenging

Parameter

PEFT ⇒ LoRA, Quantization + LoRA ⇒ QLoRA

1 PEFT, LoRA ⇒ without modify all parameter we train the model

1) Reduce computational cost
2) less memory
3) speed up the process
4) maintain the model performance

Llama → transformer ⇒ Self Attention, Neural network

↓ weights

W → huge numbers

entire weight

frozen (freeze)

Freeze

$W' = W + \Delta W$

$\Delta W = (A * B)$

A

B

A (r × d)

B (d × r)

↳ A

d ⇒ dimensions

r ⇒ low rank factor

Interview question

QLORA ≈ 32 GB RAM

QLORA = 12-24 GB RAM

Transformer ⇒ self Attention + NN

huge number weight

↑

LORA subset

(weight)

older

+ ΔW → (A × B)

Freeze

(learning)

BP epochs

(Randomly)

A → (r × d)

B → (d × r)

↑

Rank ⇒ (SVD)

Self attention ⇒ weight matrix

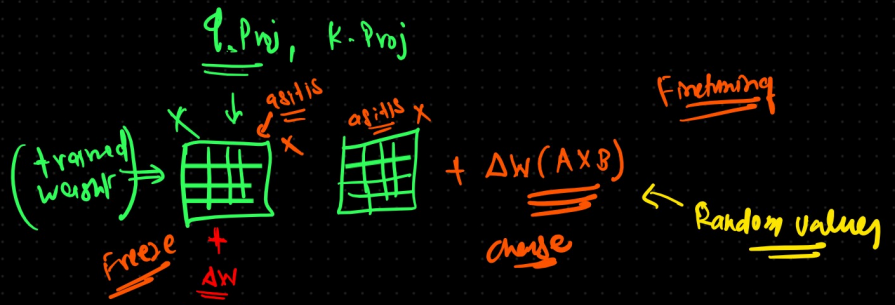
hyperparameter

- W_q ⇒ query ⇒ q-Proj
- W_K ⇒ key ⇒ K. Proj
- W_V ⇒ value ⇒ V-Proj

$W_q K V$

out-Proj

X



⇒ 1 BIT ⇒ 0 or 1

System ⇒ 0 & 1

1 BYTE ⇒ 8 BIT

1 KB ⇒ 1024 BYTE

1 MB ⇒ 1024 KB

1 GB ⇒ 1024 MB

Datatype ⇒ float 32 = 32 bit = 4 byte
float 16 = 16 bit = 2 byte
float 8 = 8 bit = 1 byte
Int 8 = 8 bit = 1 byte

Datatype

weight ⇒ 2.87 ⇒ float32 ⇒ 4 byte

GPT3.5 ⇒ 175B weight = 175B × 4 byte

=

Quantization → reduce Precision FP32 → Int 8

4 Byte → 1 Byte

LoRA ⇒ 32 RAM, 64 RAM

Q + LoRA ⇒ 16 RAM, 24GB RAM

⇒ Quantization ⇒ Reduce the Precision

Loss of information

① 4 Byte → 2 byte ②

4 Byte → 1 byte ③

4 byte → 0.5 byte ④

loss↑ ④ loss↓ ①

lower Precision (loss)

(PTQ)

1 Post-training quantization

1. First trained in full precision (FP32)
2. After training the w & b are converted to lower precision

{FP16, Int8}

(QAT)

2 Quantization aware training

1. While training use quantization effects
2. instead of training on full precision the model will train with quantization effects

today's → LoRA & QLoRA ⇒ (PTQ) QAT

(PQT) higher precision (FP32) ⇒ ① Pretraining ✗ ⇒ loded the Pretrained model

② SFT ← Quantization

PQT ← Post Quantization technique

① unsupervised ← FP32 → full precision

② SFT ← quantization

```
bnb_config=BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.float16
)
```

FP32 → normal float 4 bit (NF4 bit) ←

Loading + Computation

Running the model with FP16

NF4 | FP16

Storing in memory (VRAM)

Accuracy ↓ ← (loss↑↑) (NF4) (loss↑) (FP16) Loss ↓ FP32 → Computation

DPO, PPO