WEEK 8 — LLM FINE-TUNING, QUANTISATION & OPTIMISED INFERENCE

(Colab-Friendly | LoRA + QLoRA + GGUF + vLLM + llama.cpp)

WEEK 8 OBJECTIVES

Interns will learn:
LLM internals & transformer architecture rationale
Parameter-efficient fine-tuning (LoRA / QLoRA)
Dataset preparation for instruction tuning
Quantisation (8-bit, 4-bit, GGUF)
Inference optimization techniques
Memory profiling & speed benchmarking
LLM deployment for production inference

This week produces engineers who can **train and deploy LLMs on minimal resources.**

MODELS (COLAB-FRIENDLY — MANDATORY)

Use ONE of these:

- `Phi-2 / Phi-3 (2.7B – 3.8B)`
- `Mistral 7B Instruct`
- `TinyLlama 1.1B`
- `Qwen 1.5B–4B`

✅ Use:
`transformers + peft + trl + accelerate + bitsandbytes + llama.cpp`

DAY 1 — LLM ARCHITECTURE + DATA PREP FOR FINE-TUNING

🔹 **Learning Outcomes**
LLM anatomy (layers, attention, FFN)
Tokenization & vocab strategy
Instruction tuning vs pretraining
LoRA & PEFT fundamentals

🔹 **Topics to Learn**

- Transformer blocks
- Parameter count vs performance
- What does Fine-tuning actually change?
- Prompt-completion vs chat format
- Instruction dataset design

🔹 **Exercise**

Build your **instruction tuning dataset**:

You must include 3 types:

1. QA
2. Reasoning
3. Extraction

Format (JSONL):

```
{"instruction":"...", "input":"...", "output":"..."}
```

At least:
✅ 1,000 samples
✅ Clean + curated
✅ Domain-based (your choice: Finance, Healthcare, Coding, HR)

Run:

- Token length analysis
- Distribution graphs
- Remove outliers

🔹 **Deliverables**

```
/data/train.jsonl
/data/val.jsonl
/utils/data_cleaner.py
DATASET-ANALYSIS.md
```

DAY 2 — PARAMETER-EFFICIENT FINE-TUNING (LoRA / QLoRA)

🔹 **Learning Outcomes**
Fine-tune LLM on Colab
Use LoRA / QLoRA
Memory-saving tricks
Train with 4-bit / 8-bit

🔹 **Topics**

- PEFT (Parameter Efficient Fine Tuning)
- Rank / Alpha / Dropout
- BitsAndBytes
- Gradient checkpointing
- Mixed precision

🔹 **Exercise**

Fine-tune model using **QLoRA** with:

```
r = 16
lr = 2e-4
batch = 4
epochs = 3
4-bit loading
```

Output:
✔ Trainable params only ~1%
✔ Loss optimizing
✔ Adapter weights saved

🔹 **Deliverables**

```
/notebooks/lora_train.ipynb
/adapters/adapter_model.bin
/TRAINING-REPORT.md
```

DAY 3 — QUANTISATION (8-bit → 4-bit → GGUF)

🔹 **Learning Outcomes**
Why quantise LLMs
Memory vs accuracy trade-off
GGUF & llama.cpp support

🔹 **Topics**

- Post-training quantisation
- Static vs dynamic
- FP16 vs INT8 vs INT4
- llama.cpp conversion

🔹 **Exercise**

Convert model to:

✅ 8-bit
✅ 4-bit
✅ GGUF (q4_0 or q8_0)

Measure:

Format	Size	Speed	Quality
FP16			
INT8			
INT4			
GGUF			

🔹 **Deliverables**

```
/quantized/model-int8
/quantized/model-int4
/quantized/model.gguf
QUANTISATION-REPORT.md
```

DAY 4 — INFERENCE OPTIMISATION + BENCHMARKING

🔹 **Learning Outcomes**
Speed up LLM inference
Batching
Token streaming
CPU vs GPU inference
Context window optimization

🔹 **Topics**

- KV caching
- vLLM
- llama.cpp

- Speculative decoding
- Prompt compression

🔹 **Exercise**

Test inference using:

1. Base model
2. Fine-tuned model
3. Quantised model (gguf + llama.cpp)

Measure:

✅ Tokens/sec
✅ VRAM usage
✅ Latency
✅ Accuracy

Add:

- Streaming output mode
- Batch inference
- Multi-prompt test

🔹 **Deliverables**

```
/benchmarks/results.csv
/inference/test_inference.py
BENCHMARK-REPORT.md
```

DAY 5 — CAPSTONE: BUILD & DEPLOY LOCAL LLM API

🔹 **Learning Outcomes**
LLM as local microservice

Optimised and deployable
Ready for RAG & agents

🔹 **Topics**

- FastAPI / Flask inference server
- Streamed generations
- Prompt templates
- Model caching
- Production thinking

🔹 **Exercise (Capstone)**

Build:

```
POST /generate
POST /chat
```

Features:

✔ Uses quantised model
✔ Infinite chat mode
✔ System + user prompts
✔ Top-k, top-p, temp controls
✔ Logs + request id
✔ Ready for RAG / Agents

Folder:

```
/deploy
  app.py
  model_loader.py
  config.py
```

Optional:

- Dockerfile

- CLI mode
- Streamlit UI

🔹 **Deliverables**

```
/deploy/app.py
/README.md
/DOCKERFILE
FINAL-REPORT.md
```

WEEK 8 COMPLETION CHECKLIST

Skill	Requirement
Dataset	Custom + cleaned
Fine-tuning	LoRA / QLoRA
Quantisation	8-bit + 4-bit + GGUF
Benchmarking	Speed + Memory
Inferences	Optimised
Deployment	API running
Documentation	Full reports

EXPECTED OUTCOME

After this week, engineers can:

✔ Fine-tune any LLM on Colab
✔ Quantise for 4x smaller size
✔ Run models on laptop/CPU
✔ Achieve 2–5x faster inference

✔ Deploy a production-ready LLM
✔ Integrate into RAG or agents