

Implementation Assignment 1: Parameter-Efficient Fine-Tuning

CS 394E — Contemporary Topics in LLMs · KAUST · Fall 2026

Released	Tuesday 2026-09-29
Due	Tuesday 2026-10-06, 23:59 AST, on Blackboard
Starter file	<code>finetune.ipynb</code> (on the course website, Assignments page)
Weight	5% of the final grade (graded out of 10 points, plus up to 2 bonus points)

What this assignment is about

You will fine-tune a small LLM (Microsoft’s Phi-2, 2.7B parameters) to classify news articles from the AG News dataset, using **QLoRA**: the base model is loaded in 4-bit precision and only small LoRA adapter matrices are trained. You will then measure two things before and after fine-tuning: **task accuracy**, and **safety behavior**, i.e. how often the model refuses a small set of harmful requests.

The safety half is inspired by *Safety-Tuned LLaMAs: Lessons From Improving the Safety of Large Language Models that Follow Instructions* (Bianchi et al., ICLR 2024, <https://arxiv.org/abs/2309.07875>). The paper finds that instruction-tuned models readily comply with harmful requests, that adding a few hundred safety examples to the fine-tuning data makes them substantially safer, and that too many make them refuse harmless requests as well. The optional bonus lets you try the second finding yourself. The paper came up in the week 4 lecture on post-training; it is not an assigned reading, but it is the place to go if you want to go deeper.

The goal is not only to make the code run, but to understand what happens at each stage of the pipeline.

What you will do

Work through the notebook, complete the seven TODOs, and answer the questions in a short report.

Part	What
0	Setup and GPU sanity checks
1	Load AG News, build train/eval subsets and prompts (TODO 1–2)
2	Baseline accuracy of the base model, with a robust label parser (TODO 3)

Part	What
3	QLoRA fine-tuning: choose LoRA target modules and training arguments (TODO 4–5)
4	Task evaluation after fine-tuning
5	Mini red-team safety evaluation (TODO 6)
6	Experiments and an experiment log (TODO 7)
Bonus	Mix refusal examples into training and retrain (optional)

Experiments (Part 6)

Vary one parameter at a time, so that each comparison changes only one thing:

- the **training-set size**, `subset_train` in {2000, 5000, 8000}, with `lora_r` = 16;
- the **LoRA rank**, `r` in {8, 16}, with `subset_train` = 8000.

The run with `subset_train` = 8000 and `lora_r` = 16 belongs to both, so this is **four runs** in total. Both values are set in the configuration cell. For each run, record task accuracy, training time and GPU, and the safety refusal rate (overall and per category).

Compute

You need **Python 3.10 or newer** (3.12 is what we tested with); Colab and Ibex both provide it. Part 3 needs a **CUDA GPU with at least 16 GB of memory**; bitsandbytes 4-bit quantization does not run on a CPU or on Apple Silicon. Either of these is enough:

- **Google Colab**: the free T4 GPU works. A T4 has no bf16, so use `fp16=True` in your training arguments. Free Colab sessions time out, so save your experiment numbers as you go.
- **KAUST Ibex**: any of its GPU nodes.

Each of the four runs is a separate training job, and the 8000-example runs are the slow ones, so budget **several hours of GPU time** in total on a T4 (less on a newer GPU). **Start early**: a queue or a Colab quota is not a reason for an extension.

What to submit

On Blackboard, submit:

1. The completed notebook `finetune.ipynb` **with all outputs visible** (training logs, metrics, printed results), from your final run.
2. A **PDF report of 1–2 pages** containing:
 - your setup (GPU, model, dataset sizes) and QLoRA configuration,
 - a results table covering your four runs,
 - the key code snippets (especially your TODO implementations),
 - a safety analysis (refusal rate by category, before and after),
 - your answers to the two questions below,
 - limitations of your evaluation.

Questions to answer in the report

1. **Accuracy.** Compare the base model's `none_rate` with the fine-tuned model's. How much of the accuracy gain comes from the model learning the *task*, and how much from it learning the *output format*? What evidence in your results supports your answer?
2. **Safety.** Read the raw outputs for the safety prompts before and after fine-tuning. If the refusal rate changed, is that evidence that the model became less (or more) safe, or of something else? What would you need to measure to tell the two apart?

Grading (10 points + 2 bonus)

Component	Points
TODOs 1–6 implemented correctly; notebook runs end to end	4
All four runs completed and recorded in the experiment log (TODO 7)	2
Report: results, safety analysis, and the two questions	4
Total	10
Bonus (optional): refusal exemplars, retrain, and compare	+2

The bonus is optional and worth up to 2 extra points.

Expected effort

About **one working day** of your own time, plus GPU time that mostly runs unattended. The code itself is short; most of the time goes into setting up the environment and waiting for training runs, which is why starting early matters.

Late policy

10% penalty per 24 hours (or fraction thereof) up to 72 hours late, after which a zero is recorded without an approved extension.

Use of AI assistants

Using AI tools for this assignment is **not forbidden, but discouraged**: the point is to work through the pipeline yourself. If you do use them, **disclosure is mandatory**: at the end of your report, name the tools you used and what you used them for. You remain responsible for everything you submit.