Instruction Fine-Tuning of LLaMA-2-7B with LoRA on Dolly-15K

Nauman (AUTHOR)[1]

[1]Department / Organization, City, Country
email@example.com

Abstract – We fine-tune meta-llama/Llama-2-7b-hf using Parameter-Efficient Fine-Tuning (LoRA) on the databricks/databricks-dolly-15k dataset to adapt the base model for instruction following. We report training/validation loss curves and evaluate with AlpacaEval 2 (GPT-4 as judge over the AlpacaEval dataset) and MT-Bench (FastChat). Due to budget constraints, MT-Bench judgments are computed on a subset; full model answers for all 80 conversations are generated and stored for reproducibility. We document a known limitation: the single-turn Dolly format mismatches MT-Bench's multi-turn conversational format, and the model may append " Response:" artifacts. We provide cleaned, GitHub-renderable notebooks and scripts to reproduce fine-tuning, inference, and evaluation.

1 Background and Motivation

Large language models (LLMs) such as LLaMA-2-7B are powerful but require alignment to follow user instructions reliably. Instruction fine-tuning adapts a pretrained model from generic next-token prediction to helpful, safe, and coherent task completion. Parameter-Efficient Fine-Tuning (PEFT) with LoRA enables this alignment on modest hardware by updating a small set of low-rank adapter weights while keeping base weights frozen.

We target Dolly-15K (instruction, context, response) to train an assistant-style model and evaluate with two community benchmarks expected by the assignment: AlpacaEval 2 (instruction following, GPT-4 judged) and MT-Bench (multi-turn dialogue quality, GPT-4 judged). The goal is to demonstrate loss convergence, measurable gains over the base model, and to document practical limitations and reproducibility.

2 Relevance and related works

LoRA/PEFT has become a standard approach for efficient adaptation of LLMs, enabling fine-tuning on consumer-grade GPUs. Dolly-15K provides a permissive instruction-following corpus. AlpacaEval 2 and MT-Bench (via FastChat) are widely used evaluation protocols; both rely on LLM-as-a-judge (e.g., GPT-4) for scalable scoring. This work follows those best practices, while explicitly noting dataset–evaluation format mismatches (single-turn vs. multi-turn) as an important confounder for MT-Bench.

3 Methods and Approach

Dataset. We use databricks/databricks-dolly-15k. Each record is formatted as:

```
### Instruction:
{instruction}

### Context:
{context}

### Response:
{response}
```

We remove empty responses and split into train/validation/test (80/10/10).

Model

Training. Base: meta-llama/Llama-2-7b-hf. We apply LoRA with PEFT, enabling 4-bit quantization if needed for memory. Key settings include learning rate, epochs, global batch size, max length, gradient checkpointing, and saving LoRA adapters. Training/validation loss is tracked per epoch. Hardware details are recorded (GPU, VRAM, runtime).

Inference. For testing/inference, LoRA adapters are merged into the base model to create a standalone checkpoint for evaluation and answer generation.

Evaluation.

- *AlpacaEval 2*: It is a dataset of instruction-following prompts. We generate model responses and compute win rate using GPT-4 as an automated judge against reference responses.
- *MT-Bench (FastChat)*: We generate model answers for 80 multi-turn conversations for both baseline and fine-tuned models. GPT-4 single-answer grading is run locally on our JSONL outputs (subset if budget-limited), producing overall and per-category scores. We document that the training prompt style may induce formatting artifacts (e.g., appending " Response:"), which negatively impacts MT-Bench.

Reproducibility. All notebooks include explanatory markdown; Colab-specific metadata (e.g., `metadata.widgets`) is stripped to ensure GitHub rendering while preserving cell outputs. A FastChat submodule is added to run MT-Bench lo-

cally; answer files reside under the expected `model_answer/` directory and judgments are saved under `model_judgment/`.

4 Expected Results and Impact

Training dynamics. We expect smooth loss convergence, with validation loss decreasing alongside training loss.

AlpacaEval 2. We expect the fine-tuned model to outperform the base model in GPT-4-judged win rate on the AlpacaEval dataset. We report the computed win rate (and sample size) and include a few qualitative examples.

MT-Bench. We report overall and category scores for the subset judged within budget, clearly noting coverage (e.g., first-n conversations). We interpret results with the caveat that single-turn training may underperform on multi-turn dialogue.

Artifacts and Reuse. The project delivers reproducible notebooks, scripts, and a cleaned repository ready for GitHub rendering, along with documented steps to re-run MT-Bench judging locally.

4.1 Broader Impacts

The work demonstrates a resource-efficient path to align open LLMs for instruction following, lowering the barrier for education and prototyping. Reproducible pipelines and clear caveats can help practitioners avoid evaluation pitfalls (e.g., format mismatches) and adopt rigorous, transparent reporting.

1. What is the potential for the proposed activity to benefit society or advance desired societal outcomes?

2. To what extent do the proposed activities suggest and explore creative, original or potentially transformative concepts?

3. Is the plan for carrying out the proposed activities well-reasoned, well-organized and based on sound rationale? Does the plan incorporate a mechanism to assess success?

4. How well qualified is the individual, team or institution to conduct the proposed activities?

5. Are there adequate resources available to the principal investigator (either at the home institution or through collaborations) to carry out the proposed activities

6. How will the access to the computer resources provided by the consortium help you advance your research if your project is supported?

(1-2 paragraphs)

Table 1: Evaluation summary (example structure; fill with measured values).

Metric	**Base**	**Fine-tuned**
AlpacaEval 2 win rate (%)	–	–
MT-Bench overall (subset)	–	–

5 Data policy

We open-source the code, notebooks, and configuration to reproduce training, inference, and evaluation. Notebooks are cleaned to render on GitHub while preserving outputs. MT-Bench model answers and (partial) GPT-4 judgments are saved in the repository structure expected by FastChat for local inspection. Dataset usage follows respective licenses.