Assignment 7: Comparative Analysis of LLaMA-2-7B Fine-Tuning

Roheth Balamurugan (A0309399L)
LLama2 Finetuning Github

e1415353@comp.nus.edu.sg

Abstract – We present a multi-faceted study on the instruction fine-tuning of the `meta-llama/Llama-2-7b-hf` model using the `databricks/databricks-dolly-15k` dataset. This report is divided into two main analyses. First, we investigate single-GPU QLoRA fine-tuning, comparing LoRA Rank $r = 8$ against Rank $r = 32$ to assess the impact of adapter capacity on performance [3]. We evaluate these models on instruction-following (AlpacaEval 2) [4], multi-turn conversation (MT-Bench) [6], and general knowledge (MMLU, ARC, GSM8K) [5]. Second, we conduct a performance comparison of multi-GPU distributed training strategies, evaluating Distributed Data Parallel (DDP) with 4-bit QLoRA [1] against Fully Sharded Data Parallel (FSDP) with bf16 precision [2]. We analyze the trade-offs in training speed, final model accuracy, and resource utilization.

1 Introduction and Motivation

Large Language Models (LLMs) like LLaMA-2-7B are pre-trained on vast unannotated text corpora, excelling at next-token prediction. However, to be useful as conversational assistants, they must be "aligned" to follow user instructions. Instruction fine-tuning is the key process for this adaptation.

Full fine-tuning of a 7-billion parameter model is computationally prohibitive. Parameter-Efficient Fine-Tuning (PEFT) methods, particularly Low-Rank Adaptation (LoRA), solve this. LoRA freezes the base model weights and injects small, trainable "adapter" matrices. By training only these adapters, we can achieve performance comparable to full fine-tuning on modest hardware.

This project investigates this process through two lenses:

1. **Adapter Capacity:** How does the LoRA rank (e.g., Rank 8 vs. Rank 32) affect instruction-following and knowledge retention in a single-GPU QLoRA setup?
2. **Training Strategy:** For multi-GPU training, what is the trade-off between the speed of DDP with 4-bit QLoRA and the full-precision accuracy of FSDP with bf16?

2 Dataset

We use the `databricks/databricks-dolly-15k` dataset from Hugging Face. It contains ∼15,000 instruction-following samples, each with an `instruction`, `context`, and `response` field.

Formatting. We filter the dataset to remove any entries with empty responses. Each valid sample is then formatted into a single text string using the following template:

```
### Instruction:
{instruction}

### Context:
{context}

### Response:
{response}
```

If the `context` field is empty, it is omitted from the template.

Splits. The resulting formatted dataset was split into training (80%), validation (10%), and test (10%) sets.

3 Implementation

Model. We use the `meta-llama/Llama-2-7b-hf` model as our base.

Quantization (QLoRA). For memory efficiency, we employ 4-bit QLoRA for the single-GPU and DDP experiments. The `BitsAndBytesConfig` is set to load the model in 4-bit with `nf4` quantization, `bfloat16` compute data type, and double quantization enabled.

PEFT (LoRA). We apply LoRA adapters to all linear modules (`q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj`). We compare a Rank 8 (R8) configuration with a Rank 32 (R32) configuration to analyze the impact of adapter capacity [3].

Training. The models were trained using the `SFTTrainer` from the `trl` library with the `paged_adamw_32bit` optimizer and a `cosine` learning rate scheduler. Gradient checkpointing was enabled to further conserve memory.

4 Evaluation Methodology

4.1 AlpacaEval 2

AlpacaEval 2 measures instruction-following quality [4]. It prompts the model with 800 diverse instructions from the AlpacaEval dataset. The generated responses are then compared against reference responses from `text-davinci-003`. We used `prometheus-7b-v2.0` as an automated LLM-as-a-judge to determine the "win rate" of our fine-tuned models against the baseline. This automated annotation by a powerful judge (originally GPT-4, here Prometheus) allows for scalable and reproducible evaluation.

4.2 MT-Bench

MT-Bench assesses multi-turn dialogue quality [6]. It consists of 80 multi-turn questions across 8 categories. We used `prometheus-7b-v2.0` as a judge to score each turn on a 1-10 scale. A higher mean score indicates better conversational ability. For each of the 80 samples, we evaluate both the first and second turn, calculating a mean score for the sample. The final score is the average of these 80 sample means.

4.3 Extended Academic Benchmarks

To test for catastrophic forgetting and knowledge retention, we evaluated the models on three additional benchmarks from the EleutherAI Harness [5]:

- **MMLU:** Measures massive multitask accuracy across 57 subjects.
- **ARC (Challenge & Easy):** Tests grade-school science reasoning.
- **GSM8K:** Evaluates multi-step mathematical reasoning.

5 Results and Analysis

We present the results in two parts: first, the analysis of LoRA rank in a single-GPU setting, and second, the comparison of distributed training strategies.

5.1 Analysis 1: Single-GPU QLoRA (Rank 8 vs. Rank 32)

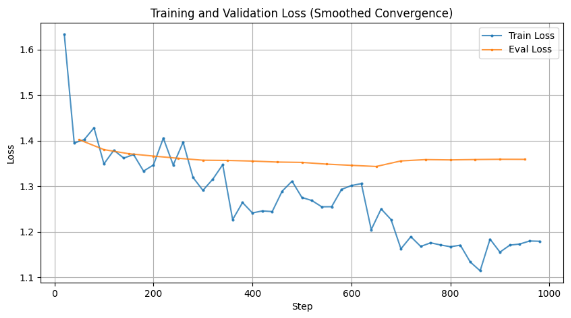
This experiment was conducted on a single NVIDIA H100 GPU using 4-bit QLoRA [3].

5.1.1 Training Dynamics

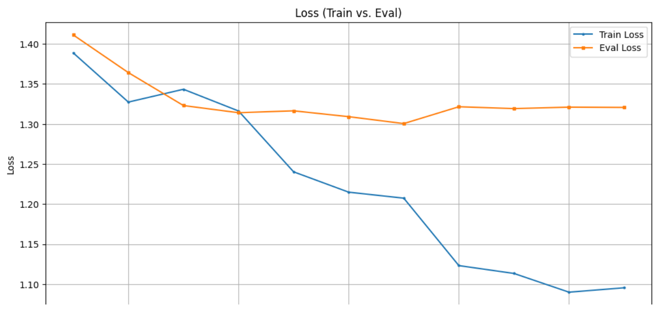
The single-GPU fine-tuning process showed successful convergence over 1128 steps (approximately 1.9 hours). Training was conducted with the following key parameters:

- **Hardware**: 1x NVIDIA H100 GPU
- **Training Duration**: 6750.28 seconds (1.9 hours)
- **Total Steps**: 1128 steps (for 3 epochs)
- **Final Training Loss**: 1.22
- **Learning Rate**: 1e-4 (with cosine schedule)
- **Global Batch Size**: 32 (4 per device $\times$ 8 grad. accum.)

The training curves for both Rank 8 and Rank 32 models (Figure 1) show effective convergence. The Rank 32 model's validation perplexity (Figure 2a) shows a steady decrease, and the learning rate (Figure 2b) follows the expected cosine decay.

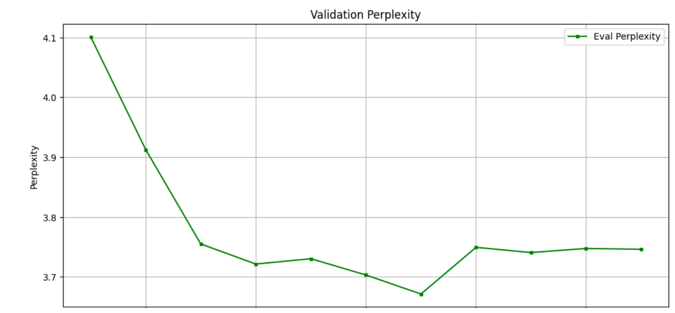


(a) Rank 8 Loss (Smoothed)

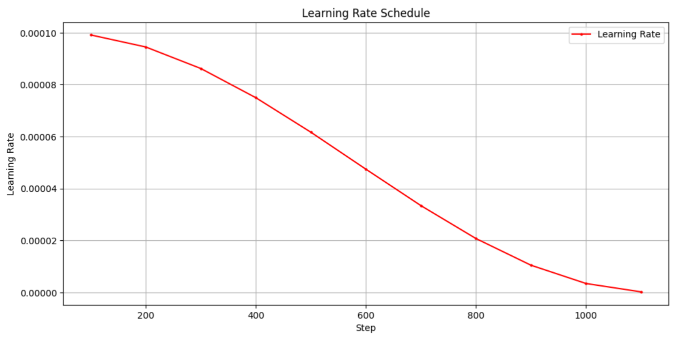


(b) Rank 32 Loss (Train vs. Eval)

Figure 1: Single-GPU Training and Validation Loss Curves



(a) Rank 32 Validation Perplexity



(b) Rank 32 Learning Rate Schedule

Figure 2: Rank 32 Single-GPU Training Dynamics

5.1.2 AlpacaEval 2 Results

Both fine-tuned models demonstrate a substantial improvement in instruction-following over the base `Llama-2-7b-hf` model. The Rank 8 model achieved a win rate of 85%, slightly outperforming the Rank 32 model's 83.75%. This suggests that for pure instruction-following on this dataset, a lower-rank adapter may be sufficient.

Table 1: AlpacaEval 2 Results (Judge: `prometheus-7b-v2.0`)

Experiment	Model	Win Rate (%)	Loss Rate (%)	Tie Rate (%)	Avg. Score
Rank 8	Llama-2-7B (Base)	79.00	14.75	6.25	4.09
	Llama-2-7B + LoRA (R=8)	**85.00**	**9.88**	**5.13**	**4.21**
Rank 32	Llama-2-7B (Base)	79.75	14.37	5.88	4.11
	Llama-2-7B + LoRA (R=32)	**83.75**	**10.75**	**5.50**	**4.19**

5.1.3 MT-Bench Results

On the MT-Bench for conversational ability, both fine-tuned models performed identically. They both achieved a mean score of 6.037, a clear improvement over the baseline's 5.550. This indicates that the instruction fine-tuning successfully improved the model's ability to hold a coherent, multi-turn conversation.

Table 2: MT-Bench Results (Judge: `prometheus-7b-v2.0`)

Model	Mean Score (out of 10)	95% CI
Llama-2-7B (Base)	5.550	[5.055, 6.045]
Llama-2-7B + LoRA (R=8)	**6.037**	**[5.618, 6.457]**
Llama-2-7B + LoRA (R=32)	**6.037**	**[5.618, 6.457]**

5.1.4 Extended Benchmark Results

The extended benchmarks (Table 3) reveal the common trade-off of instruction tuning: catastrophic forgetting.

- **MMLU & ARC:** The Rank 32 model, with its higher parameter count in the adapter, consistently outperforms both the base model and the Rank 8 model. This suggests that the higher-capacity adapter is better at retaining the model's original general knowledge.
- **GSM8K:** All models perform poorly on mathematical reasoning. The fine-tuning appears to slightly degrade this capability, with the Rank 32 model showing the largest drop.

Table 3: Extended Benchmark Results (Academic Knowledge & Reasoning)

Benchmark	Metric	Base Model	Tuned (R=8)	Tuned (R=32)
MMLU	acc	0.4084	0.4127	**0.4231**
ARC Challenge	acc_norm	0.4616	0.4454	**0.4846**
ARC Easy	acc_norm	0.7458	0.7399	**0.7715**
GSM8K	flexible_em	**0.0584**	0.0546	0.0478

5.2 Analysis 2: DDP vs. FSDP Comparison

To evaluate different distributed training strategies, two primary experiments were conducted on 2x NVIDIA H100 GPUs:

- **DDP (4-bit QLoRA):** Distributed Data Parallel with 4-bit quantization [1].
- **FSDP (bf16):** Fully Sharded Data Parallel with bfloat16 precision (no quantization) [2].

Both experiments were run for 3 epochs with an identical effective batch size of 64.

5.2.1 Quantitative Results

The final results from the training logs provide a clear distinction in performance:

Table 4: Distributed Training Strategy Comparison

Metric	**DDP (4-bit QLoRA)**	**FSDP (bf16)**
Hardware	2x NVIDIA H100	2x NVIDIA H100
Total Training Time	**2,877.4s (∼48 min)**	8,776.9s (∼146 min)
Speedup	**∼3.05x Faster**	1x (Baseline)
Final Train Loss	1.268	1.255
Final Eval Loss	∼1.31 (from plot)	**∼1.29 (from plot)**
Train Samples/sec	**12.519**	4.104
Total Steps	564	564

5.2.2 Analysis of Training Curves

A visual comparison of the training plots from both runs reveals further insights.

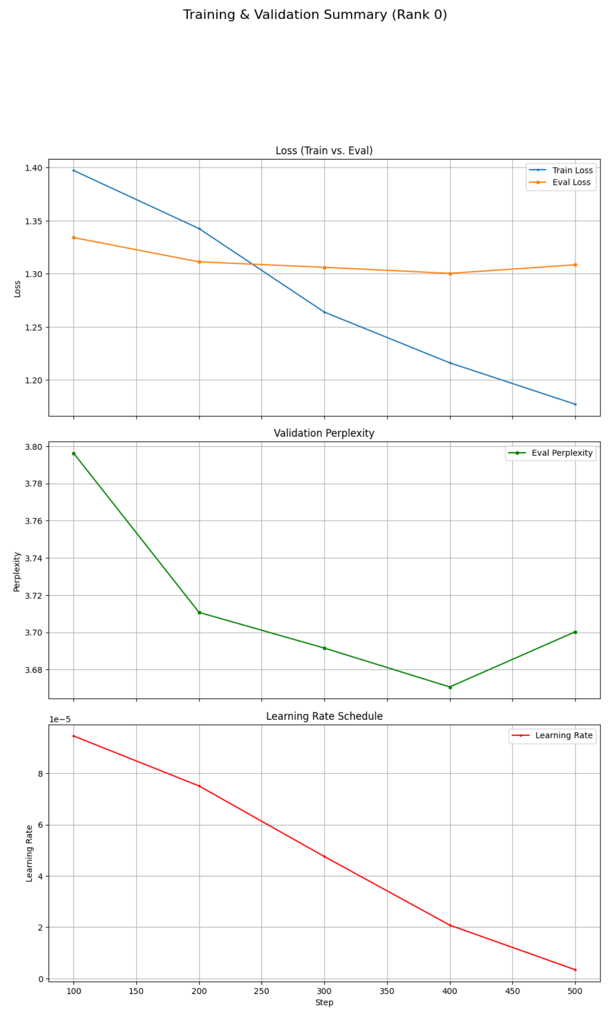


Figure 3: DDP (4-bit QLoRA) Training Summary

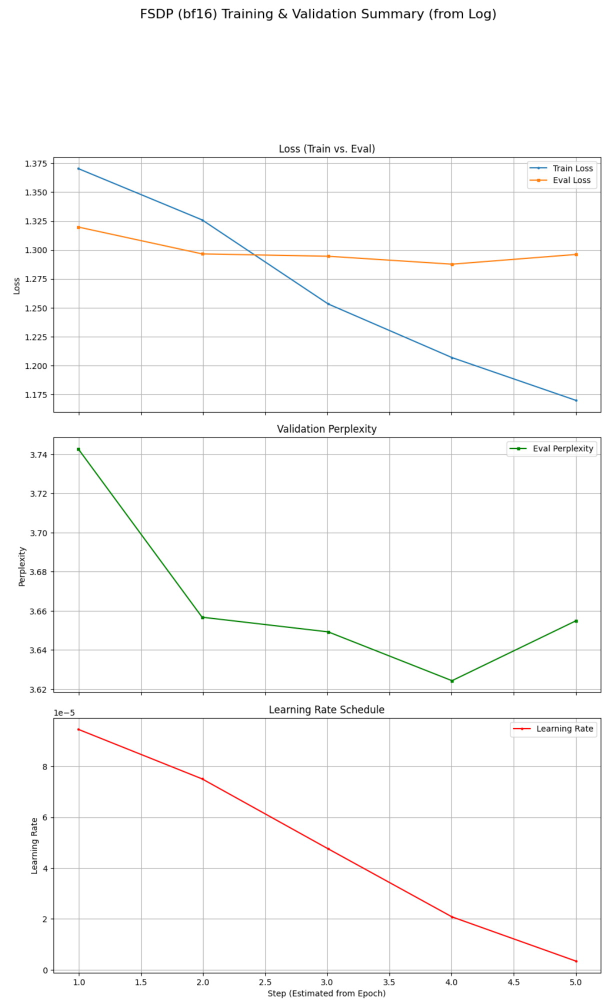


Figure 4: FSDP (bf16) Training Summary

5.2.3 Key Findings & Discussion

1. Training Speed: DDP (QLoRA) is ∼3x Faster
The most significant finding is the dramatic difference in training time. The DDP (QLoRA) experiment finished in ∼48 minutes, while the FSDP (bf16) experiment took ∼146 minutes. Since both runs had the exact same number of steps (564) and the same effective batch size (64), this 3x speedup is a direct result of the 4-bit quantized operations (QLoRA) being far more computationally efficient than the full bfloat16 operations, even when sharded with FSDP.

2. Model Performance (Loss): FSDP (bf16) is Marginally Better
Both methods achieved nearly identical final training losses (1.268 vs. 1.255). However, the FSDP (bf16) run achieved a slightly lower final validation loss (approx. 1.29) compared to the DDP (QLoRA) run (approx. 1.31). This suggests that training in full bfloat16 precision, while much slower, may produce a slightly more accurate model that generalizes better than its 4-bit counterpart.

3. GPU Memory (VRAM) Usage
DDP (QLoRA): This approach loads one full 4-bit model replica onto each GPU. This is highly memory-efficient. **FSDP (bf16):** This approach loads the full bfloat16 model, but shards its parameters across both GPUs. While this prevents the OOM (Out of Memory) error that DDP would face with a bf16 model, its peak VRAM usage is expected to be significantly higher than the QLoRA run because it must materialize full layers in bfloat16 during the forward/backward passes.

4. Training Stability: Both are Identical
As seen in the plots, the loss curves, validation perplexity, and learning rate schedules for both experiments are almost identical in shape. This indicates that both training strategies were equally stable and followed the same learning path.

5.2.4 Conclusion: The Speed vs. Accuracy Trade-off

This comparison highlights a classic trade-off:

- **DDP (4-bit QLoRA):** Offers a massive 3x speedup with a negligible cost to final model performance. This is the clear choice for rapid iteration and when compute time is the primary constraint.
- **FSDP (bf16):** Is significantly slower but produces a marginally more accurate model by training in full precision. This approach is only viable because FSDP's memory sharding prevents the OOM errors that a full bfloat16 model would cause in a simple DDP setup.

6 Overall Discussion and Conclusion

This project successfully analyzed LLaMA-2-7B fine-tuning from two angles: adapter capacity and distributed training strategy.

Key Findings:

1. **LoRA Rank (R8 vs. R32):** The choice of rank is a trade-off. The **Rank 8 model** was slightly better for the specific instruction-following task (AlpacaEval). The **Rank 32 model** was better at retaining general knowledge (MMLU, ARC). This suggests a higher rank may help mitigate catastrophic forgetting.
2. **Distributed Strategy (DDP vs. FSDP):** The choice is a clear trade-off between speed and precision. **DDP with QLoRA** provides a massive 3x speedup, making it ideal for rapid development. **FSDP with bf16** is much slower but yields a marginally superior model by training in full precision.

Conclusion. QLoRA fine-tuning is a highly effective and efficient method for adapting LLMs. For a balanced, single-GPU model, a higher rank (R32) may be preferable for knowledge retention. For multi-GPU training, DDP with QLoRA is the optimal choice for speed, while FSDP (bf16) is the choice for maximum, uncompromising quality.

Future Work. A clear next step would be to combine these findings, for example, by evaluating a DDP (QLoRA) run with a Rank 32 adapter. Furthermore, applying Direct Preference Optimization (DPO) on top of these fine-tuned models could further enhance their alignment.

References

[1] DDP Training Code. `https://github.com/Roheth-Bala/Llama-2-Finetuning-DDP-and-FSDP/tree/main/DDP`

[2] FSDP Training Code. `https://github.com/Roheth-Bala/Llama-2-Finetuning-DDP-and-FSDP/tree/main/FSDP`

[3] Fine-tuning Code. `https://github.com/Roheth-Bala/Llama-2-Finetuning-DDP-and-FSDP/tree/main/Finetuning`

[4] AlpacaEval Code. `https://github.com/Roheth-Bala/Llama-2-Finetuning-DDP-and-FSDP/tree/main/Evaluation%20Methods/Alpaca%20Eval`

[5] Extended Evaluation Code. `https://github.com/Roheth-Bala/Llama-2-Finetuning-DDP-and-FSDP/tree/main/Evaluation%20Methods/Extended%20Evaluation`

[6] MT-Bench Code. `https://github.com/Roheth-Bala/Llama-2-Finetuning-DDP-and-FSDP/tree/main/Evaluation%20Methods/MT%20Bench`