NLP-LORA-FINETUNING AND BENCHMARK RESULTS

1. Project Overview

The objective of this project is to enhance the code reasoning and generation capabilities of a pre-trained Large Language Model (LLM) using Low-Rank Adaptation (LoRA). By fine-tuning the base model on two distinct datasets, I aim to compare the effects of “Deep Reasoning" traces versus “Diverse Problem Types” on the model’s final performance in competitive programming tasks.

- Base Model: Qwen/Qwen2.5-Coder-1.5B-Instruct
- Technique: Parameter-Efficient Fine-Tuning (PEFT) via LoRA.

2. Dataset Analysis

Two seperate datasets were utilized for comparative training:

1. DEEP Dataset (CodeGen-Deep-5K): Focused on complex reasoning steps and logical consistency.
2. DIVERSE Dataset (CodeGen-Diverse-5K): Focused on a wide variety of coding problems and edge cases.

Note: For both models, training was focused on the solution field (clean Python code) to optimize for direct output quality.

3. Training Setup & Hyperparameters

The following technical configurations were applied to ensure efficient training while preventing overfitting.

3.1. LoRA Configuration

Deep Dataset

Parameter	Value	Description
Rank (r)	16	Rank of the update matrices
Alpha	32	LoRA scaling factor
Target Modules	q_proj, v_proj, k_proj, o_proj, gate_proj, up_proj, down_proj	Layers where LoRA is applied

Dropout	0.1	Prevention of co-adaptation of weights
Bias	None	No bias terms were train

Diverse Dataset

Parameter	**Value**	**Description**
Rank (r)	32	Rank of the update matrices
Alpha	64	LoRA scaling factor
Target Modules	q_proj, v_proj, k_proj, o_proj, gate_proj, up_proj, down_proj	Layers where LoRA is applied
Dropout	0.05	Prevention of co-adaptation of weights
Bias	None	No bias terms were train

3.2 Technical Configuration

Deep Dataset

Optimizer	AdamW (8-bit)
Learning Rate	2e-5 (with Cosine Decay)
Epochs	3
Context Length	1024
Effective Batch Size	16
System Prompt	"You are an expert Python programmer. Please read the problem carefully before writing any Python code."

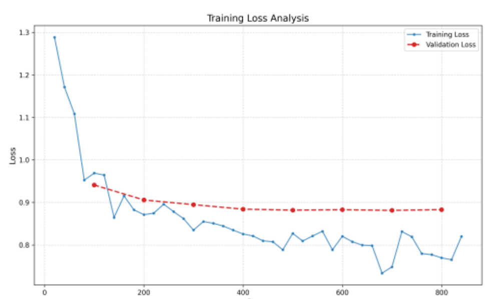
Diverse Dataset

Optimizer	AdamW (8-bit)
Learning Rate	1e-4 (with Cosine Decay)
Epochs	3
Context Length	1024
Effective Batch Size	16
System Prompt	"You are an expert Python programmer. Please read the problem carefully before writing any Python code."

4. Training Results

4.1. Loss Curves

The following charts illustrate the convergence of the training and validation loss over the steps.



Deep Model Loss Graph

Diverse Model Loss Graph

4.2. Checkpoint Selection

Checkpoints were evaluated every 100 steps. The best-performing checkpoints selected for final evaluation were:

- DEEP Model: checkpoint-step-846-epoch-3
- DIVERSE Model: checkpoint-step-800-epoch-3

5. Final Evaluation & Bencmarks

The models were evaluated against a benchmark dataset to measure their accuracy in generating valid, functional Python code.

Model	**Best Checkpoint**	**Pass@1 (%)**	**Solved Questions**
Base Model (Qwen2.5-Coder-1.5B)		21.9	9/41
Fine-Tuned (DEEP Model)	checkpoint-step-846-epoch-3	[Value]	10/41
Fine-Tuned (DIVERSE Model)	checkpoint-step-800-epoch-3	26.8	11/41

6. Conclusion

In this project, I successfully implemented LoRA fine-tuning on the Qwen2.5-Coder-1.5B-Instruct model. My finding suggests that ... This experimentation proves that parameter-efficient fine-tuning can significantly specialize a small-scale model for high-level technical tasks.

7. Resources & Links

- GitHub Repository: https://github.com/abdurrahman-gulmez/NLP-LLM-FINETUNING
- HuggingFace Dataset (DEEP): https://huggingface.co/datasets/Naholav/CodeGen-Deep-5K
- HuggingFace Dataset (DIVERSE):https://huggingface.co/datasets/Naholav/CodeGen-Diverse-5K
- HuggingFace Model (Qwen): https://huggingface.co/Qwen/Qwen2.5-Coder-1.5B-Instruct