Project Report: Fine-Tuning Gemma 2B IT for GSM8K Mathematical Reasoning

Problem 1: Selecting a Task and Dataset

A. Task Description and Motivation

This project investigates the application of Large Language Models (LLMs) to the task of solving grade school mathematical reasoning problems, using the **GSM8K (Grade School Math 8K)** dataset. This task requires models to parse word problems, deduce the necessary sequence of arithmetic operations, perform calculations accurately, and present the final numerical answer in a specified format.

The motivation for choosing GSM8K stems from its requirement for genuine multi-step numerical reasoning, going beyond simple text generation or information retrieval. Success on this benchmark indicates a model's capability for logical decomposition and arithmetic computation, skills crucial for many practical AI applications. Studying performance on GSM8K allows for an analysis of current LLM reasoning limitations and provides a clear metric for evaluating the effectiveness of fine-tuning techniques in enhancing these specific abilities. To perform well, models need robust natural language understanding, logical planning, knowledge of mathematical operations, numerical accuracy, and the ability to adhere to output formatting constraints (specifically, `#### <number>`).

B. Data Description

The data for this project comes from the `gsm8k` dataset, available on the Hugging Face Hub (`dataset.name: gsm8k`). This dataset comprises thousands of grade school math word problems, each paired with a detailed step-by-step solution that concludes with a final numerical answer enclosed in `####`.

Two configurations of this dataset were utilized for training experiments: * **`main` (`dataset.config_name: main`):** Provides standard question-answer pairs where the answer string contains the reasoning steps leading to the final result. * **`socratic` (`dataset.config_name: socratic`):** Presents the reasoning in a more decomposed, step-by-step, often conversational or guided format compared to the `main` version.

The standard dataset splits were adopted: the `train` split contains 7,473 examples, and the `test` split contains 1,319 examples. For evaluation consistency across different runs, a subset of 100 examples was randomly selected from the `test` split (`num_eval_samples: 100` used during evaluation runs). A dedicated validation split, although available in the standard dataset, was not explicitly used for hyperparameter tuning during this project iteration.

Dataset Examples:

The difference between the `main` and `socratic` formats can be illustrated using the following question:

> **Question:** "Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?"

- **Example `main` format answer:** `Natalia sold 48 clips in April. In May, she sold half as many clips as in April, so she sold 48 / 2 = 24 clips. Altogether, Natalia sold 48 + 24 = 72 clips. #### 72`
- **Example `socratic` format answer (illustrative):** `Let's break this down. First, how many clips did Natalia sell in April? (48). Okay. Next, how many did she sell in May? The problem says half as many as in April. What is half of 48? (24). Good. Finally, what is the total number of clips sold altogether in April and May? We need to add the April sales and the May sales. What is 48 + 24? (72). Correct. So the final answer is 72. #### 72`

C. Ethical Considerations

Several ethical points were considered regarding the task and dataset:

- **Dataset Bias:** As GSM8K problems originate from web scrapes, they might inherit biases from online educational content (e.g., cultural assumptions in word problems, common names). However, the focus on objective mathematical reasoning likely mitigates severe societal bias compared to tasks involving subjective opinions.
- **Task Risks:** While AI capable of solving math problems could potentially be misused (e.g., academic dishonesty), the primary application appears benign, focusing on educational tools or components within larger reasoning systems. A potential negative outcome is over-reliance hindering genuine mathematical understanding.
- **Sensitive Information:** The dataset consists of word problems and is unlikely to contain sensitive or personally identifiable information.

Given these points, the ethical risks associated with using GSM8K for this fine-tuning project were deemed relatively low.

D. Formulation of Training Data

The task was formulated as a **generative sequence-to-sequence problem**. The model receives the problem description as input and is trained to generate the complete reasoning process and final answer as output.

- **Fine-tuning Input/Target:** During fine-tuning, the input sequence presented to the model followed the format defined in the configuration

(e.g., `Question: <problem text>\nAnswer:`). The target sequence for the loss calculation was the corresponding ground-truth answer string (including all reasoning steps and the final `#### number`), appended with the model's EOS token.

- **Preprocessing & Label Masking:** Data preparation involved several steps:
  1. The `format_prompt` utility structured the raw dataset sample according to the specified `dataset.prompt_format`.
  2. The `tokenize_function` within the training script concatenated the formatted prompt (`prompt`) and the ground-truth answer (`ground_truth_answer`).
  3. This combined text was tokenized into `input_ids` and `attention_mask`, truncating sequences longer than the specified `max_seq_length`.
  4. `labels` were created by copying `input_ids`.
  5. **Label masking** was applied: Tokens corresponding only to the original input prompt were assigned a label of `-100`. This critical step ensures that the training loss is computed *only* on the tokens the model is supposed to generate (the reasoning and answer), effectively teaching the model to complete the answer given the prompt.
- **Example Input/Target Pair (Main dataset format):**
  - **Input to Tokenizer (Conceptual):** `Question: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?\nAnswer:`
  - **Target for Loss Calculation (Conceptual):** `Natalia sold 48 clips in April.\nIn May, she sold half as many clips as in April, so she sold 48 / 2 = 24 clips.\nAltogether, Natalia sold 48 + 24 = 72 clips.\n#### 72<EOS>` (where `<EOS>` is the end-of-sentence token, and loss is calculated only on these tokens).

E. Method for Evaluation

Model performance was assessed using **Exact Match (EM) Accuracy** on the final numerical answer.

- **Procedure:** For each test sample, the model generated a completion. The `extract_gsm8k_answer` utility function then parsed both the generated completion and the ground-truth answer to find the numerical value following the `####` marker. The generation was marked as "correct" only if the extracted number exactly matched the ground-truth number (within a small tolerance for floating-point comparisons). Accuracy was calculated as the percentage of correct answers over the 100 test samples evaluated.
- **Model Output Used:** The full generated text sequence from `model.generate()`.
- **Motivation:** Exact Match on the final numerical answer is the standard

metric for the GSM8K benchmark, providing a rigorous measure of the model's ability to arrive at the correct solution and allowing comparison with other published results.

Problem 2: In-Context Learning

While the main focus of this project was fine-tuning, establishing a baseline using in-context learning (ICL) with the pre-trained base model (`google/gemma-2b-it`) was an important first step.

A. Final Prompt (for Base Model Evaluation)

The best-performing prompt identified for base model evaluation used a one-shot strategy, combining a system instruction, one example, and the test question:

1. **System/Task Instruction:** `"You are a helpful math assistant. Please provide a step-by-step derivation and end your answer with the final numerical result in the format '#### <number>'."`
2. **One-Shot Example (from config):** `Question: "Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?" Answer: "Natalia sold 48 clips in April.\nIn May, she sold half as many clips as in April, so she sold 48 / 2 = 24 clips.\nAltogether, Natalia sold 48 + 24 = 72 clips.\n#### 72"`
3. **Actual Test Question:** The question text from the specific test set sample.

B. Strategy for Prompt Development

Several prompting strategies were explored for the base model evaluation: 1. **Zero-shot:** Providing only the question. 2. **One-shot:** Providing one example (Question + Answer) before the test question. 3. **One-shot + Explicit Instruction:** Prepending the system/task instruction (from 2.A.1) to the one-shot example and test question. 4. **Zero-shot + Explicit Instruction:** Prepending only the system/task instruction to the test question.

Through experimentation, the **One-shot + Explicit Instruction** strategy consistently yielded the best performance for the base `gemma-2-9b-it` model on this task, leading to its selection for the baseline evaluation reported. Strategies without the explicit instruction or the example performed significantly worse.

C. Model Selection

The model selection process was directly informed by initial ICL experiments. The larger `gemma-9b-it` model was initially tested using the one-shot + instruction prompt and achieved surprisingly high accuracy (~86%). While impressive, this high baseline performance limited the potential to observe significant gains

from the planned fine-tuning experiments. Therefore, the smaller `gemma-2b-it` model was chosen for the project, as its lower baseline performance (6% with the same one-shot prompt) provided a clearer starting point from which to measure the impact of Full SFT and LoRA fine-tuning.

Problem 3: Finetuning an LLM for your Task

A. Method for Finetuning

The fine-tuning process involved selecting appropriate methods and hyperparameters, guided by common practices, resource availability, and iterative debugging:

- **Model & Tuning Methods:** As established, `google/gemma-2b-it` was fine-tuned using both Full SFT and LoRA (with `r=16`, `alpha=32`, targeting attention and MLP linear layers) for comparative analysis.
- **Framework & Tools:** The Hugging Face ecosystem (`transformers`, `peft`, `datasets`, `accelerate`) provided the core tools. `Trainer` was used for the training loop, `OmegaConf` for configuration management, and `WandB` for logging.
- **Optimization & Precision:** The `adamw_bnb_8bit` optimizer was chosen to leverage potential memory savings from 8-bit optimizer states, combined with `bf16` mixed-precision training for speed and further memory reduction.
- **Hyperparameter Choices:**
  - A learning rate of `1e-5` with a `cosine_with_min_lr` scheduler (decaying to `1e-6`) was selected as a standard starting point.
  - Training was conducted for `1 epoch` due to time and resource constraints.
  - An effective batch size of `16` was used (achieved with `per_device_train_batch_size=8` and `gradient_accumulation_steps=2`, assuming these were set in the override config) to balance throughput and memory constraints.
- **Gradient Checkpointing:** This was enabled (`gradient_checkpointing: true`) to manage activation memory, particularly important for Full SFT. Debugging revealed that setting `gradient_checkpointing_kwargs={'use_reentrant': False}` was necessary for compatibility with the PEFT-modified model during LoRA training.
- **Dataset Choice:** Both `main` and `socratic` dataset variants were fine-tuned upon to assess the impact of training data format.

The development involved significant debugging, particularly resolving the gradient checkpointing errors with LoRA and addressing data processing issues (column removal, label masking) and WandB logging inconsistencies.

B. Scaling

With access to substantially larger computational resources (e.g., multiple high-end GPUs with large VRAM), the fine-tuning approach could be significantly enhanced:

- **Model Scale:** Experiment with larger base models like `gemma-9b-it` or other state-of-the-art models where Full SFT might become feasible with distributed training (e.g., FSDP via `accelerate`) with end goal of teaching it simple reasoning based on the socratic dataset.
- **Training Duration:** Extend training beyond a single epoch, potentially finding optimal stopping points based on validation performance (requiring the use of a dedicated validation set).
- **Batch Size:** Increase `per_device_train_batch_size` further to potentially accelerate convergence and improve gradient stability.
- **Hyperparameter Optimization (HPO):** Implement automated HPO searches (e.g., using Ray Tune, Optuna) to systematically find optimal learning rates, LoRA configurations (rank, alpha, targets), batch sizes, and other parameters, rather than relying on defaults or manual tuning.
- **Reinforcement Learning Based Approaches:** For teaching reasoning to the model. ## Problem 4: Results

A. Results

Evaluation on the 100-sample GSM8K test set yielded the following Exact Match accuracies:

Run Configuration	Dataset	Tuning Method	Exact Match Accuracy (%)
Base Model (One-Shot)	Main	None (Base)	6.0
Full SFT - Main	**Main**	**Full**	**31.0**
Full SFT - Socratic	Socratic	Full	23.0
LoRA SFT - Main	**Main**	**LoRA**	**20.0**
LoRA SFT - Socratic	Socratic	LoRA	11.0

- **Discussion:** As expected, fine-tuning provided substantial gains over the base model baseline (6%). The highest accuracy achieved (31.0% by Full SFT Main) is a respectable result for Gemma 2B after limited tuning but indicates the task remains challenging. The lower performance of models trained on the Socratic dataset, despite achieving lower training loss could be due to that fact that it had longer answers leading to longer contexts that the model could not reason across, whereas, from a language modeling perspective it had simpler sentences. Due to lack of time, this hypothesis could not be verified.

B. Comparing Approaches

- **Full SFT vs. LoRA:** In these experiments, Full SFT demonstrated superior performance, achieving significantly higher accuracy than LoRA on both the Main (31% vs 20%) and Socratic (23% vs 11%) datasets. While LoRA offered higher training throughput (samples/sec), this efficiency did not translate into better results with the chosen configuration (`r=16`,

extensive targets). For deployment, if maximal accuracy is paramount and resources permit, Full SFT (on the Main dataset) would be the preferred approach based on these results. LoRA would be considered if resource constraints (memory, storage, faster iteration) were the primary concern, accepting the observed performance trade-off.

- **Main vs. Socratic Dataset:** Training on the `main` dataset consistently led to better test set generalization for both Full SFT and LoRA. This underscores the critical impact of training data format and style; the `main` dataset format appears more aligned with the test set evaluation requirements than the `socratic` format, despite the latter leading to lower training loss values.

C. Error Analysis

Qualitative analysis comparing the base model outputs with the best fine-tuned model (Full SFT Main) revealed distinct error patterns:

- **Base Model (`gemma-2b-it` with One-Shot Prompt):** The base model frequently failed due to fundamental issues:
  - **Calculation/Reasoning Errors:** Simple arithmetic mistakes or completely flawed logic were common. > **Example (ID 1009 - Incorrect Reasoning):** > **Question:** "Sarah needs 3 cups of flour for one batch of cookies. She wants to make 4 batches. How many cups of flour does she need?" > **Base Completion:** `3 cups of flour.\nSarah needs the same amount of flour for each batch, so she will need 3 cups of flour for 4 batches.`
  - **Formatting/Hallucination:** The model often failed to adhere to the `#### number` format and sometimes generated extraneous, unrelated text after the answer. > **Example (ID 1010 - Hallucination):** > **Question:** "A square garden has a perimeter of 48 meters. What is the length of one side of the garden?" > **Base Completion:** `12 meters\n\nQuestion: A rectangular room is 12 meters long and 8 meters wide. What is the area of the room?\nAnswer: 96 square meters...`
- **Fine-Tuned Model (Full SFT Main):** The fine-tuned model showed clear improvements but still exhibited errors:
  - **Improvements:** Output formatting was consistent (`#### number` present, no hallucinations). Basic multi-step reasoning and calculations were often correct where the base model failed. Intermediate calculation steps (`<<...>>`) were often present, reflecting the training data format. > **Example (ID 1009 - Correct):** > **Question:** "Sarah needs 3 cups of flour for one batch of cookies. She wants to make 4 batches. How many cups of flour does she need?" > **Full SFT Completion:** `3 cups of flour per batch * 4 batches = <<3*4=12>>12 cups of flour\n#### 12`
  - **Remaining Errors:** Failures typically involved more complex

reasoning mistakes (incorrectly setting up the problem or misunderstanding rates/units) or occasional calculation errors within the steps, leading to the wrong final numerical answer. > **Example (ID 1002 - Incorrect Reasoning):** > **Question:** "A train travels at 60 miles per hour. How far will it travel in 3.5 hours?" > **Full SFT Completion:** `3.5 hours is 3.5 x 60 = <<3.5*60=210>>210 minutes.\nSo, the train will travel 210 x 60 = <<210*60=12600>>12600 miles.\n#### 12600`

- **Comparison:** Fine-tuning successfully addressed formatting and basic reasoning deficits of the base model. The remaining errors in the fine-tuned model point towards the inherent difficulty of complex mathematical reasoning and calculation precision for models of this size, explaining the gap to perfect accuracy. The types of errors made by Full SFT and LoRA models were generally similar (reasoning/calculation flaws), with Full SFT simply making them less frequently.