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 presents a significant challenge for LLMs as it requires not only understanding the natural language description of a problem but also deducing the necessary sequence of arithmetic operations, performing these calculations accurately, and presenting the final numerical answer in a specific, structured format.

The motivation for selecting GSM8K stems from its focus on genuine multi-step numerical reasoning, a capability that extends beyond typical text generation or information retrieval tasks. Success on this benchmark serves as a valuable indicator of a model's capacity for logical decomposition and arithmetic computation – skills essential for advancing AI towards more complex problem-solving applications. Studying performance on GSM8K allows for a clear analysis of current LLM reasoning limitations and provides a concrete metric for evaluating the effectiveness of various fine-tuning techniques in enhancing these specific cognitive abilities. To excel at this task, models must demonstrate robust natural language understanding, logical planning, knowledge of fundamental mathematical operations, numerical precision, and the ability to strictly adhere to output formatting constraints (specifically, concluding with `#### <number>`).

B. Data Description

The data utilized in this project originates from the `gsm8k` dataset, sourced via the Hugging Face Hub (`dataset.name: gsm8k`). This dataset contains thousands of grade school level math word problems. Each problem is accompanied by a detailed, step-by-step natural language solution that culminates in a final numerical answer, clearly demarcated by the `####` pattern.

Two distinct configurations of this dataset were employed during the Supervised Fine-Tuning (SFT) phase to explore the impact of data format:

- **main (`dataset.config_name: main`):** This configuration provides standard question-answer pairs. The answer string typically includes the intermediate reasoning steps written in a natural, explanatory style.
- **socratic (`dataset.config_name: socratic`):** This variant presents the reasoning process in a more decomposed, step-by-step manner, often resembling a guided dialogue or a Socratic questioning approach, potentially offering more explicit guidance for the model during training.

The standard dataset splits were adopted for the experiments. The `train`

split, containing 7,473 examples, was used for fine-tuning (both SFT and PPO rollouts using the `main` config). The `test` split comprises 1,319 examples; however, for consistent and efficient evaluation across different models and methods, a randomly selected subset of 256 examples from the `test` split was used (`num_eval_samples: 256`). A dedicated validation split was not explicitly utilized for hyperparameter tuning in this project iteration.

Dataset Examples:

To illustrate the stylistic difference between the `main` and `socratic` answer formats, consider 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

An assessment of potential ethical issues related to the GSM8K task and dataset was conducted:

- **Dataset Bias:** The problems are derived from web scrapes, which might introduce biases reflective of online educational materials (e.g., cultural contexts, common names). However, the objective nature of mathematical reasoning likely minimizes the impact of severe societal biases compared to tasks involving subjective judgments or social interactions.
- **Task Risks:** The primary risk associated with an AI proficient in solving math problems relates to potential misuse in academic settings (e.g., cheating). However, the intended applications are generally benign, focusing on educational assistance or as components in larger AI reasoning systems. A secondary consideration is the risk of over-reliance potentially hindering the development of human mathematical skills.
- **Sensitive Information:** The dataset contains fictional word problems and is considered highly unlikely to include any personally identifiable or sensitive information.

Based on this assessment, the ethical risks associated with using the GSM8K dataset for this research project were deemed relatively low.

D. Formulation of Training Data

The core task was formulated as a **generative sequence-to-sequence problem**. The model receives the mathematical word problem as input and is trained to generate the sequence representing the step-by-step reasoning process and the final numerical answer.

- **Supervised Fine-Tuning (SFT) Formulation:**
  - **Input:** The model received input sequences formatted according to the configuration (e.g., `Question: <problem text>\nAnswer:`).
  - **Target:** The target sequence for loss calculation was the ground-truth answer string provided in the dataset (including reasoning and `#### number`), appended with the model's specific end-of-sentence (EOS) token.
- **Reinforcement Learning (PPO) Formulation:**
  - **Input:** The model (policy) received the question prompt.
  - **Action:** The model generated sequences (rollouts) representing potential answers.
  - **Reward:** A reward signal (details in Problem 3.A) was calculated based on the generated sequence, guiding the PPO algorithm to update the model towards generating higher-reward outputs.
- **Preprocessing & Label Masking (SFT):** A crucial step in preparing data for SFT involved label masking to ensure the model learned to *generate* the answer, not just reproduce the prompt.
  1. The `format_prompt` utility function structured the input question text.
  2. The formatted prompt and the ground-truth answer string were concatenated.
  3. This combined text was tokenized into `input_ids` and `attention_mask`, with truncation applied at `max_seq_length`.
  4. Initial `labels` were created by copying `input_ids`.
  5. Tokens corresponding *only* to the input prompt portion were identified, and their corresponding positions in the `labels` tensor were set to `-100`.
  6. During training, the standard cross-entropy loss function automatically ignores these `-100` labels, meaning the loss was computed solely based on the model's predictions for the tokens belonging to the reasoning and final answer part of the sequence.
- **Example SFT 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 SFT 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>`

E. Method for Evaluation

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

- **Procedure:** For each sample in the 256-example test set, the model generated a completion sequence. A utility function (`extract_gsm8k_answer`) then parsed both the model's generation and the ground-truth answer string, specifically searching for the numerical value immediately following the `####` marker using regular expressions. A generated answer was deemed "correct" if and only if the extracted number precisely matched the ground-truth number (allowing for minor floating-point tolerance). The overall accuracy was then calculated as the percentage of correct answers across the test set.
- **Model Output Used:** The full generated text sequence produced by the model's `generate()` method.
- **Motivation:** This Exact Match approach, focusing on the final numerical result, is the standard evaluation protocol for the GSM8K benchmark. It provides a strict, objective measure of the model's ability to arrive at the correct answer and facilitates comparison with results reported in other research.

Problem 2: In-Context Learning

While fine-tuning was the primary focus, establishing an initial baseline using In-Context Learning (ICL) with the pre-trained `google/gemma-2b-it` model was performed to understand the starting capabilities before adaptation.

A. Final Prompt (for Base Model Evaluation)

The prompt configuration found to be most effective for the base model evaluation involved a one-shot strategy, combining a system-level instruction, a single demonstration example, and the target 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 specific question text from the evaluation 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. **Zero-shot + Instruction:** The system instruction preceding the test question.
4. **One-shot + Instruction:** The system instruction, followed by the example, followed by the test question.

Informal testing indicated that the **One-shot + Explicit Instruction** approach yielded the most coherent and accurate results compared to the other strategies, particularly in encouraging the model to attempt the desired format. Therefore, this strategy was adopted for the formal baseline evaluation. The other approaches often resulted in incorrect formats, incomplete answers, or failures to follow the reasoning structure.

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 project pivoted to the smaller `gemma-2b-it` model. Its significantly lower baseline performance (6% accuracy with the same prompt) provided a more substantial gap and thus a better opportunity to clearly observe and compare the impact of different fine-tuning techniques like Full SFT and LoRA.

Problem 3: Finetuning an LLM for your Task

A. Method for Finetuning

The project implemented three distinct fine-tuning stages: Full SFT, LoRA SFT, and PPO RL. The setup for each was determined through a combination of standard practices, resource considerations, and insights gained during debugging.

- **Model & Initial Tuning Methods:** The `google/gemma-2b-it` model served as the base. It was fine-tuned using both **Full SFT** (updating all ~2.5B parameters) and **LoRA SFT** (rank `r=16`, `alpha=32`, targeting linear

layers in attention and MLP blocks, resulting in ~20.6M trainable parameters) for direct comparison. Subsequently, the best-performing SFT model (**Full SFT Main**) was used as the starting point for **Reinforcement Learning using PPO**.

- **Frameworks & Tools:** SFT stages utilized the Hugging Face ecosystem (`transformers`, `peft`, `datasets`, `accelerate`), with `Trainer` managing the training loop. `OmegaConf` handled configuration merging, and `WandB` was employed for logging. The PPO stage was implemented from scratch.
- **SFT Optimization & Precision:** For SFT, the `adamw_bnb_8bit` optimizer was used with `bf16` mixed-precision training to optimize memory usage and training speed.
- **SFT Hyperparameters:** Key settings included a learning rate of `1e-5` (decaying via `cosine_with_min_lr` to `1e-6`), 1 training epoch, and an effective batch size of 16 (using `per_device_train_batch_size=8` and `gradient_accumulation_steps=2`).
- **Reinforcement Learning (PPO) Stage:**
  - **Starting Model:** Checkpoint from the Full SFT Main run.
  - **Algorithm:** PPO (from scratch implementation).
  - **Reward Function:** The reward signal for PPO was based directly on the primary evaluation metric: **Exact Match Accuracy**. After generating a completion (rollout) for a given prompt, the final numerical answer was extracted. If this number exactly matched the ground-truth numerical answer, a positive reward (e.g., +1) was assigned; otherwise, a neutral reward (e.g., 0 or -1) was given. This directly optimizes the model policy towards generating answers that are numerically correct according to the benchmark's standard.
  - **Key PPO Hyperparameters:** Learning Rate `5e-7`, Adam 8-bit optimizer, 2 PPO epochs per batch, 512 rollout samples per PPO step, PPO batch size 32 (mini-batch 2), grad accum 8, KL coeff 0.05, VF coeff 0.1, Entropy coeff 0.01, Clip ratio 0.2, GAE lambda 0.95, Gamma 0.99. Trained for 100 PPO steps.
- **Gradient Checkpointing:** Enabled for SFT runs (`gradient_checkpointing: true`). Debugging necessitated explicitly setting '{'use_reentrant': False}' for compatibility with LoRA. PPO stage also used gradient checkpointing.

The development process required addressing several technical challenges, including data processing errors (column mismatches during mapping), WandB logging issues (metrics not appearing in summaries), and particularly the complex interaction causing gradient errors when combining LoRA and gradient checkpointing.

B. Scaling

Access to greater computational resources would allow for several enhancements to this project:

- **Larger Models:** Utilizing larger base models (e.g., `gemma-9b-it`, Llama

3 variants).
- **More Complex Datasets:** Like MATH, PRM800K.
- **Extended Training:** Training for multiple epochs (SFT and RL), using validation sets for optimal stopping.
- **Increased Batch Sizes:** Increasing SFT and PPO batch sizes.
- **Systematic Hyperparameter Optimization:** Implementing automated HPO for SFT, LoRA, and PPO parameters.
- **Advanced RL:** Exploring more sophisticated reward modeling (e.g., rewarding correct intermediate steps, penalizing incorrect format) or different RL algorithms.

Problem 4: Results

A. Results

Models were evaluated on a 256-sample subset of the GSM8K test set using Exact Match accuracy.

Run Configuration	Dataset	Tuning Method	Exact Match Accuracy (%)
Base Model (One-Shot)	Main	None (Base)	6.0
Full SFT - Main	Main	Full SFT	**31.0**
Full SFT - Socratic	Socratic	Full SFT	23.0
LoRA SFT - Main	Main	LoRA	20.0
LoRA SFT - Socratic	Socratic	LoRA	11.0
PPO (from Full SFT Main)	Main	RL (PPO)	28.0

- **Observations:** Fine-tuning (both SFT and PPO) yielded substantial improvements over the 6% baseline. Full SFT on the Main dataset achieved the highest accuracy at 31.0%. The PPO stage, starting from this best SFT model and optimizing directly for exact match reward, resulted in a slightly lower accuracy of 28.0%. The overfitting pattern observed with the Socratic dataset during SFT (lower training loss but worse test accuracy) highlights the sensitivity to training data distribution. The PPO result not surpassing the SFT result suggests that the RL setup, while optimizing for the target metric, may have encountered challenges like instability or reward hacking, or perhaps required more tuning/steps to converge effectively, potentially leading to the observed qualitative regressions.

B. Comparing Approaches

- **ICL vs. Fine-Tuning:** Fine-tuning (SFT and PPO) proved vastly superior to the one-shot ICL baseline (max 31% vs 6%), confirming the necessity of adaptation for achieving reasonable performance on this complex reasoning task.

- **Full SFT vs. LoRA:** Full SFT consistently resulted in higher accuracy than LoRA across both datasets (Main: 31% vs 20%; Socratic: 23% vs 11%). While LoRA offered efficiency gains (higher throughput), it came at the cost of final performance in this configuration.
- **SFT vs. PPO:** PPO, applied after Full SFT on the Main dataset, did not improve upon the SFT result, instead showing a slight decrease in accuracy (28% vs 31%). This indicates that RL, while powerful, requires careful tuning and reward design to guarantee improvements over a strong SFT baseline, especially given the potential for regressions observed qualitatively. Optimizing directly for the final metric via RL doesn't always lead to better overall generalization compared to SFT.
- **Main vs. Socratic Dataset:** The `main` dataset proved more effective for training models that generalized well to the test set, yielding better results than the `socratic` dataset for both SFT methods.
- **Deployment Choice:** Based purely on accuracy, the **Full SFT Main** model would be the preferred choice. LoRA offers a more resource-efficient alternative if a moderate accuracy drop is acceptable. The PPO model, in this iteration, did not offer an advantage over its SFT starting point and introduced some undesirable behaviors.

C. Error Analysis

Qualitative analysis comparing the base model, the best SFT model (Full SFT Main), and the PPO model reveals the evolution of model capabilities and remaining weaknesses:

- **Base Model (`gemma-2b-it` with One-Shot Prompt):** Showcased fundamental weaknesses. Errors included basic calculation mistakes, inability to follow multi-step logic, failure to adhere to the `#### number` format, and generation of irrelevant, hallucinated text. > **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.` > **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 - 31% Accuracy):** Demonstrated significant progress. Format adherence was excellent (correct `#### number` usage, no hallucinations). Basic reasoning and calculations improved, often showing intermediate steps (`<<...>>`). However, errors persisted in setting up more complex problems or performing specific calculations accurately. > **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` > **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`

- **Reinforcement Learning Model (PPO from Full SFT Main - 28% Accuracy):** Showed mixed results compared to its SFT starting point. It maintained good formatting and corrected the reasoning error on the train speed problem (ID 1002). However, it introduced *new* errors on problems the SFT model solved correctly (e.g., rectangle area ID 1004, square perimeter ID 1010) and exhibited significantly degraded, looping behavior on another (book reading ID 1005). This suggests the PPO training, possibly due to the simple exact match reward signal or hyperparameter tuning, optimized for certain behaviors while causing regressions elsewhere, ultimately leading to slightly lower overall accuracy. > **Example (ID 1002 - Correct):** > **Question:** "A train travels at 60 miles per hour. How far will it travel in 3.5 hours?" > **PPO Completion:** `3.5 hours is 3.5 x 60 = <<3.5*60=210>>210 miles.\nSo, the train will travel 210 miles in 3.5 hours.\n#### 210` > **Example (ID 1004 - Regression):** > **Question:** "A rectangle has a length of 15 cm and a width of 8 cm. What is its area?" > **PPO Completion:** `15 cm is equal to 15/100 = <<15/100=0.15>>0.15 m.\n8 cm is equal to 8/100 = <<8/100=0.08>>0.08 m.\nThe area of the rectangle is 0.15 x 0.08 = <<0.15*0.08=0.012>>0.012 m2.\n#### 0.012`
- **Error Comparison:** SFT primarily fixed formatting and basic reasoning flaws of the base model. PPO showed potential to fix specific SFT errors but also introduced regressions, indicating its optimization based on exact match reward was not globally beneficial for accuracy in this instance. The errors remaining after all fine-tuning stages point to the inherent difficulty of complex mathematical reasoning and calculation precision for models of this scale.

D. PPO Training Observations

The PPO training stage, while ultimately not improving test set accuracy, exhibited typical RL training dynamics as observed in the WandB logs.

- **Rewards:** The reward metric `rollout/reward_mean` (total reward) did not show much of a trend and oscillated just below the SFT performance levels.
- **Value Function:** The value function loss (`loss/value`) decreased and converged to a lower value, showing the value network was learning to predict expected returns.
- **KL Divergence:** Metrics like `loss/approx_kl` converged to a relatively low value (0.085 -> 0.06), controlled by the KL coefficient (`kl_coeff: 0.05`), preventing the PPO policy from diverging too drastically from the initial SFT policy.
- **Other Metrics:** Policy entropy loss (`loss/entropy`) increased and then stabilized, indicating exploration followed by increasing confidence. Clipping fractions (`params/value_clip_frac`, `params/policy_clip_frac`) remained low and stable. Gradient norm (`params/grad_norm`) stayed between 3.5 and 4.0 .

While the training metrics suggest the PPO algorithm was functioning and optimizing for the exact match reward signal, the lack of improvement (and slight decrease) in final test accuracy highlights the challenge of designing a reward function or tuning PPO effectively to capture and encourage the desired complex reasoning capabilities needed for robust performance on GSM8K. The qualitative errors introduced by PPO further suggest potential misalignment.