Fine-Tuning Qwen2.5-0.5B for Mathematical Reasoning
A Bootstrap-Based Evaluation

Anonymous CVPR submission

Paper ID *****

Abstract

We fine-tune Qwen2.5-0.5B on Math220k to improve its mathematical reasoning and multiple-choice accuracy. Due to the dataset's lack of a test split and noisy solutions, we use a custom train/test partition and strict answer extraction. Evaluation is conducted via bootstrap sampling across Math220k, Math500, and GPQA. The fine-tuned model improves GPQA clean accuracy from 0.089 to 0.386, indicating better answer formatting and reasoning. Math500 accuracy remains stable, suggesting generalization is preserved. Our setup ensures consistent evaluation and reveals the limitations of relying on noisy auto-generated datasets.

1. Supervised Fine-Tuning (SFT) Setup

1.1. Dataset and Split

We used the **math220k** dataset as the training corpus. Since it does not contain a predefined test set, we manually split the `default` subset into two parts: 1. 83,000 samples for training, and 2. 10,000 samples as the test set. An additional 10,000 samples were held out from the training set to serve as a validation set (referred to as the "on-hold" subset). The `default` subset was chosen exclusively to avoid unverified or augmented data, ensuring more stable supervision.

1.2. Training Configuration

The base model used is Qwen2.5-0.5B. We trained it using a learning rate of 2.5×10^{-6} with a cosine decay scheduler and no warm-up steps. Model checkpoints were saved every 500 steps for downstream evaluation.

1.3. Prompt Formatting and Supervision

Each sample was constructed by concatenating three components: 1. a problem statement from the `problem` field; 2. an instruction prompting the model to solve the question step-by-step and produce the final answer in a structured format (either `\boxed{X}` or `Answer: X`); and 3. the `solution` field appended as a reference output for supervised learning. This prompt construction was handled via a custom formatter implementation located in `dexin_src/utils/formatter.py`, ensuring uniform formatting and compatibility with the answer extraction routines used during evaluation.

2. Model Evaluation

2.1. Evaluation Strategy

Due to limitations of the official Open-R1 environment, which restricts the use of tools such as `Trainer.log_metrics`, `evaluate()`, or `lighteval`, we developed a custom evaluation script `eval_checkpoints_auto.py`. This script evaluates each model checkpoint independently by comparing model outputs against reference answers using exact-match accuracy. All evaluation was conducted on fixed-size bootstrapped datasets of 1000 examples per split to ensure stability and comparability across checkpoints.

2.2. Bootstrapping Rationale

Bootstrap sampling (size = 1000) was adopted for three primary reasons: 1. it reduces accuracy variance, particularly for small datasets such as GPQA; 2. it allows quick evaluation cycles on a local machine without sacrificing statistical significance; and 3. it guarantees fairness, as every checkpoint is evaluated on datasets of identical size and distribution. This makes it easier to observe the effect of fine-tuning without confounding it with data fluctuation.

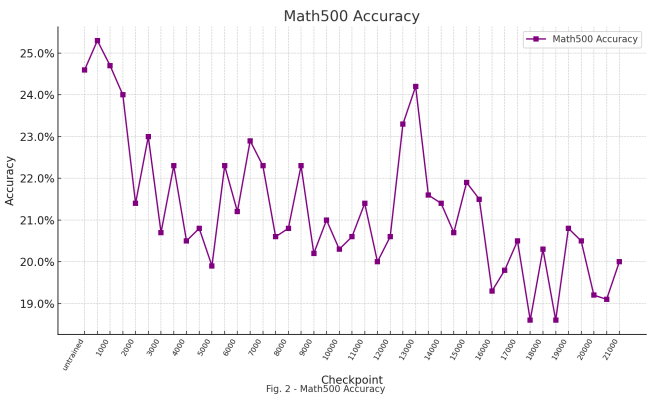

Figure 1. Math500 Accuracy over Checkpoints

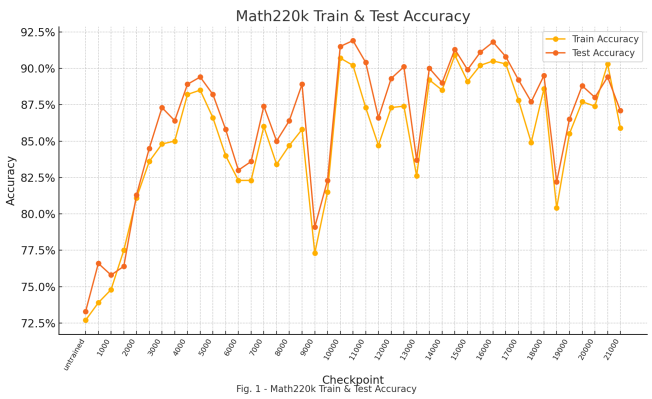

Figure 3. Train and Test Accuracy on Math220k

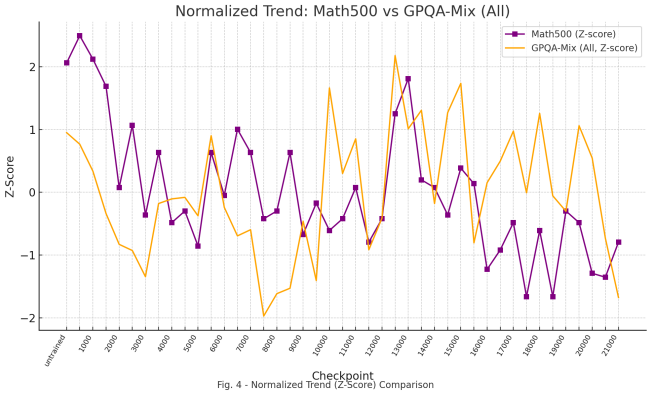

Figure 2. Normalized GPQA-All vs Math500 Accuracy Trends

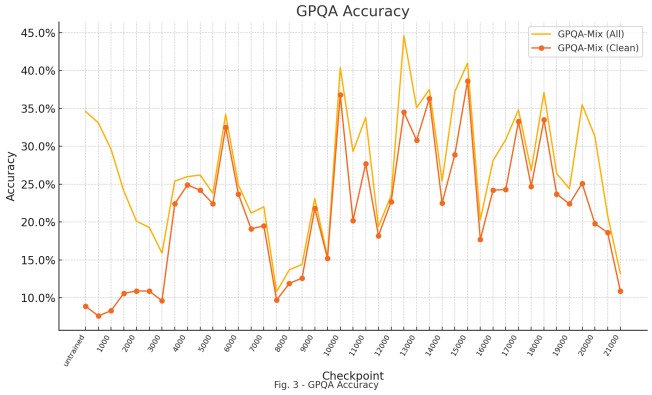

Figure 4. GPQA Accuracy (Clean vs. All)

3. Best Checkpoint Selection and Training Termination

3.1. Answer Extraction Strategy

We defined a two-tiered extraction approach. First, strict extraction identifies answers formatted exactly as instructed during training—`\boxed{X}` or `Answer: X`—which is used for all math datasets and for clean evaluation on GPQA. Second, a soft extraction phase (used only for GPQA) attempts to extract reasonable answers from free-form completions using regular expression heuristics, e.g., matching phrases such as "the correct answer is X" or "option X is correct." These strategies yield two evaluation metrics: 1. *clean accuracy*, based only on strict-format responses; and 2. *all accuracy*, which includes both clean and soft-extracted answers.

3.2. Exclusion of the `solution` Field During Evaluation

Although the math220k dataset includes both a `solution` and `answer` field, we chose to evaluate only using the `answer`. The `solution` field is often noisy, inconsistently formatted, and in some cases incorrect or misleading, as it is generated automatically and unverified. The `answer` field, though sometimes ambiguous in format, tends to be shorter, more stable, and is used as the filtering criterion in the dataset's default split. This decision aligns with best practices in evaluation protocols of datasets such as MATH and GSM8k, where exact answers are prioritized for grading, and the solution is treated as reasoning context.

4. Evaluation Observations

Test and train accuracy on math220k increased rapidly from approximately 72% to 88% within the first few checkpoints. From checkpoint 3500 onward, both train and test accuracy oscillated within a narrow range between 82% and 91%, indicating saturation. On the other hand, math500 accuracy gradually declined from 0.246 (untrained) to values around 0.19–0.21, with fluctuations. This suggests that while the model became more specialized to math220k, it began to lose generalization capabilities for unseen mathematical reasoning tasks.

GPQA accuracy reveals the most compelling trend. In the untrained model, clean and all accuracies diverged by 25.7%, showing poor formatting discipline. However, after checkpoint 3500, this gap shrank substantially. The average difference across checkpoints beyond 3500 is just 3.41% (std: 2.98%), indicating a clear learning of structured answer formatting. Furthermore, maximum improvement in clean accuracy over the base model is 29.7%, while for all accuracy it is 10%. These patterns demonstrate that fine-tuning significantly improved answer formatting without heavily sacrificing correctness.

5. Best Checkpoint Selection and Training Termination

Although both training and test accuracy rose rapidly in early stages, they plateaued after checkpoint-10000, suggesting that the model had saturated its learning capacity on math220k. From that point onward, no further accuracy gains were observed; instead, accuracy remained stable within a tight interval, showing oscillations typical of overfitting onset.

At the same time, accuracy on math500—a general reasoning benchmark—declined gradually. This indicates a loss of generalization and rising task-specific bias. Compounding this issue is the noisy nature of math220k's `solution` field, which may introduce unreliable reasoning patterns when used in SFT. Continued training on such supervision risks amplifying spurious correlations or overfitting to flawed logic.

From GPQA evaluations, it is evident that the model's formatting ability was largely acquired early in training. After checkpoint-3500, the difference between clean and all accuracy stabilized to 2–4%. This plateau suggests no further gains in instruction-following precision. In contrast, answer correctness (all accuracy) plateaued around 0.40, with clean accuracy peaking at 0.386—both exceeding the base model significantly.

Given these trends—accuracy saturation on math220k, slight degradation on math500, diminishing improvement on GPQA, and growing risk of solution contamination—we select **checkpoint-15000** as the optimal trade-off. This checkpoint balances performance across all axes: high in-domain accuracy, competitive general reasoning ability, and faithful adherence to formatting instructions.

6. Resources:

Source code is available at: https://github.com/DexinRen/open-r1_DR_test

Model checkpoint: https://huggingface.co/DexinR/qwen2.5-math220k-ckpt15000

Evaluation result is available at: https://github.com/DexinRen/open-r1_DR_test/tree/master/dexin_src/eval_output