Quantization Meets Reasoning: Exploring and Mitigating Degradation of Low-Bit LLMs in Mathematical Reasoning

Anonymous ACL submission

Abstract

Post-training quantization (PTQ) is a practical route to deploy reasoning-capable LLMs under tight memory and latency budgets, yet it can markedly impair mathematical reasoning, with accuracy drops up to 69.81% for smaller models. We address two deployment-critical questions: **Where** along a step-structured solution does degradation first arise, and **how** can it be mitigated while staying in the low-bit regime? Across representative PTQ methods, model families, and math benchmarks from grade-school to competition level, we perform format-aligned chain-of-thought evaluation with step-aligned error attribution and uncover two robust regularities: (i) PTQ disproportionately elevates method and execution errors rather than high-level conceptual mistakes; and (ii) failures emerge early, with the first vulnerable step cascading to corrupt the final answer. This suggests a localized intervention principle: restore token-level margins exactly at the earliest failure frontier. We instantiate it as a lightweight measure→locate→restore loop operating directly on the quantized model: detecting the first faulty step, constructing targeted **"Silver Bullet"** preference pairs, and applying small-scale supervised tuning. With as few as 332 curated examples and 3–5 minutes of single-GPU compute, it recovers quantized-model math reasoning toward the full-precision baseline while preserving PTQ efficiency, turning low-bit degradation from a global accuracy problem into a local, reproducible process intervention.

1 Introduction

Transformer-based large language models (LLMs) such as LLaMA (Grattafiori et al., 2024), GPT (Achiam et al., 2023), and Qwen (Yang et al., 2024) have demonstrated strong performance on complex reasoning tasks, including mathematical competitions (ChuGyouk, 2025), code generation (Chen et al., 2021), and logical inference (Pan et al., 2023). Yet attaining reliable accuracy on such tasks typically requires large parameter counts. The resulting inference latency and memory footprint make deploying full-precision, ultra-large models impractical in many resource-constrained scenarios. To balance resource use and accuracy, model compression has been extensively studied, including quantization (Yang et al., 2019; Rokh et al., 2023), knowledge distillation (Hinton et al., 2015; Gou et al., 2021), and pruning (Han et al., 2015). Among these, post-training quantization (PTQ) (Banner et al., 2019) lowers precision to reduce memory and improve throughput, especially on edge hardware. However, recent evidence indicates that low-bit operation can substantially degrade mathematical reasoning (Feng et al., 2024; Liu et al., 2025). This raises two practical questions for deployment: **Where does degradation emerge in the reasoning process?** and **How can it be mitigated while remaining in the low-bit regime?**

We study these questions through a systematic evaluation of representative PTQ methods on multiple open-source model families across math benchmarks (Cobbe et al., 2021; Hendrycks et al., 2021; ChuGyouk, 2025) of varying difficulty. By examining each model's step-by-step solution trajectory and performing fine-grained error attribution, we uncover two consistent patterns: (i) PTQ disproportionately increases *method* and *execution* errors—such as misapplied algorithms and arithmetic slips—while leaving high-level conceptual understanding relatively intact; and (ii) these failures tend to emerge *early* in the solution, with the first erroneous step cascading to corrupt all subsequent reasoning.

These patterns convert a diffuse accuracy problem into a localized intervention target. We propose a lightweight *exploring* and *mitigating* loop: identify the first faulty step in the quantized model's trace, construct contrastive "Silver Bullet" prefer-

ence pairs that correct exactly that step, and apply small-scale DPO tuning. With as few as 332 curated pairs and 3-5 minutes on a single GPU, this procedure recovers 4-bit models' math reasoning toward their full-precision baselines while preserving PTQ's efficiency benefits.

As shown on Figure 1, we frame the study in the regime most relevant to practice: PTQ rather than quantization-aware training, so as to preserve the efficiency budget and expose unmodified low-bit failure modes. To cover current practice, We evaluate three representative methods spanning weight-only and weight-activation designs. Experiments use two model familes at 0.5-14B scales commonly deployed under constraints scenario and edge devices (Xie et al., 2025). While our evidence is drawn from these specific choices, the diagnostic and recovery procedures are designed to be architecture- and quantizer-agnostic. Specifically, our primary contributions are as follows:

- We build a step-aligned measurement suite and hierarchical error taxonomy that expose a robust PTQ-induced shift toward *method* and *execution* errors, with earlier first-step flips, consistent across the most popular models, bit-widths, and benchmarks.
- We develop an automated error-attribution pipeline (judge ensemble and light human audit) that attains 97.2% labeling accuracy on 9,908 failure cases, enabling fine-grained, reproducible attribution by error type and first faulty step.
- We introduce a compact capability restoration loop that tunes the quantized model with targeted "Silver Bullet" pairs, recovering W4A16 mathematical reasoning to near full precision with just 332 curated examples and 3-5 minutes on a single GPU without access to pretraining data.

2 Related Works

2.1 Post-Training Quantization and Its Effect on Reasoning

Quantization reduces LLM inference cost by mapping parameters and/or activations to low-bit representations, with a well-documented efficiency-accuracy trade-off (Yang et al., 2019; Rokh et al., 2023). A key design axis is weight-only quantization (e.g., AWQ (Lin et al., 2024), GPTQ (Frantar et al., 2022)), which compresses weights while retaining high-precision activations, versus weight-activation quantization (e.g., SmoothQuant (Xiao et al., 2023)), which quantizes both for stronger end-to-end efficiency. Recent work has begun to examine how these methods affect reasoning specifically. Feng et al. (2024) demonstrate that reduced numerical precision degrades mathematical problem-solving, showing a correlation between bit-width and accuracy on arithmetic-heavy tasks. Concurrently, Liu et al. (2025) provide a broader empirical study of quantized reasoning models, confirming substantial accuracy drops on multiple benchmarks. However, both studies measure degradation at the level of final-answer accuracy; neither localizes where in the step-by-step reasoning chain quantization-induced failures originate, nor proposes a targeted repair mechanism. Our work addresses exactly this gap: we perform process-level error attribution and use the resulting diagnosis to guide efficient recovery.

2.2 Recovering Capabilities in Compressed Models

When quantization or other compression methods degrade model quality, a natural remedy is lightweight post-hoc fine-tuning. LoRA (Hu et al., 2021) and its quantization-aware variant QLoRA (Dettmers et al., 2024) enable parameter-efficient adaptation with minimal memory overhead, making them practical for tuning already-quantized models. In the reasoning domain, preference optimization like DPO (Rafailov et al., 2024) has been shown effective for aligning model outputs with desired reasoning behaviors. However, existing applications of DPO to reasoning improvement operate on full-precision models and use generic preference data without targeting specific failure modes (Wei et al., 2021; Taori et al., 2023; Rafailov et al., 2024). Our recovery procedure differs in two respects: it operates directly on quantized models, and it constructs preference pairs that are anchored at the first diagnosed failure step, providing a targeted learning signal rather than generic supervision.

3 Methodology

3.1 Quantization-Induced Degradation: Measurement and Attribution

Our framework addresses the first deployment question: where does quantization degrade reasoning? Through a three-stage measurement protocol: quantize, standardize, and attribute. We first apply representative PTQ methods to obtain quantized models (§3.1.1). Because raw model outputs of-

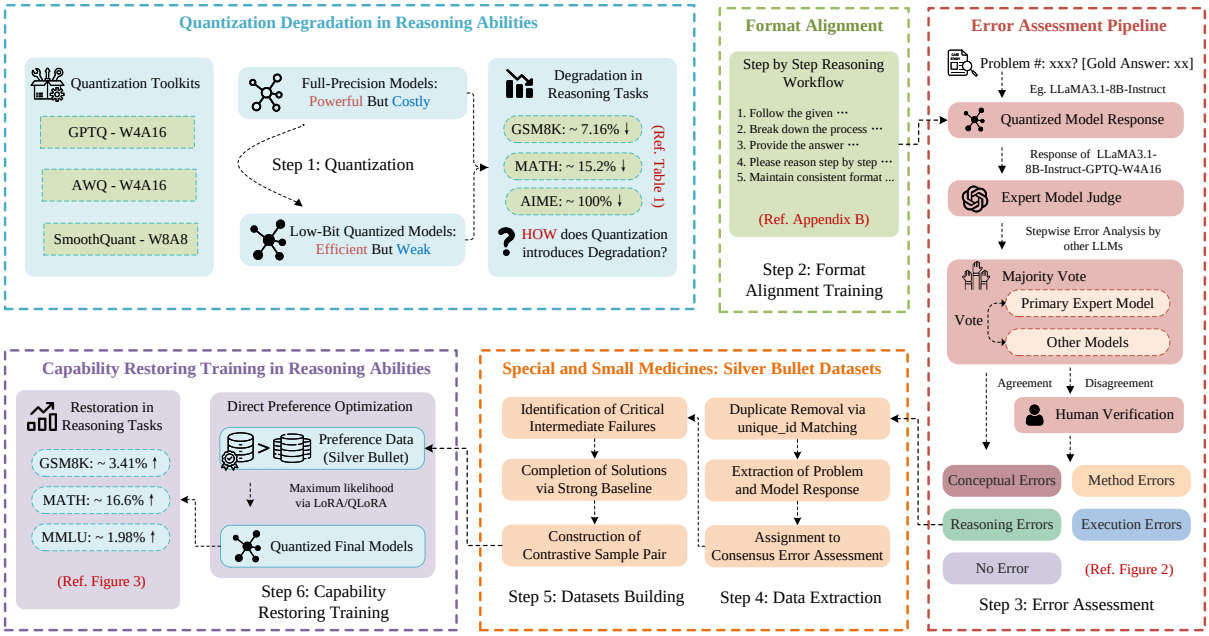

Figure 1: Pipeline of our study for investigating and restoring mathematical reasoning capabilities in quantized language models. We begin by identifying performance degradation caused by quantization, then apply format alignment training and a structured error assessment pipeline involving expert model judgments. Through this process, we analyze reasoning failures in step-by-step outputs. Targeted "Silver Bullet" datasets are constructed based on consensus error types, and recovering reasoning performance while maintaining the efficiency of low-bit models.

ten differ in surface formatting—step delimiters, answer boxing, variable naming—we then perform a lightweight format alignment stage that standardizes the chain-of-thought layout without injecting new mathematical knowledge (§3.1.2). With outputs thus comparable, we carry out a joint quantitative-qualitative error analysis on each model's step-by-step solution trajectory, grounding our error taxonomy in established frameworks from mathematics education (§3.1.3). This protocol lets us pinpoint which cognitive operation breaks first and at which step, providing the diagnostic signal that drives the targeted restoration in §3.2. All analyses center on mathematical reasoning tasks, which demand multi-step procedural accuracy and therefore amplify the effects of low-bit precision loss.

3.1.1 Quantization

We study quantization-induced degradation under representative post-training quantization (PTQ) settings commonly used in practice. Specifically, we consider two weight-only 4-bit methods, AWQ and GPTQ (W4A16), and one weight-activation 8-bit method, SmoothQuant (W8A8). These settings are also practically meaningful from a deployment perspective. As summarized in Appendix A.2.1, they substantially **reduce memory 41.8%-62.4% footprint** relative to FP16. For throughput, it can further improve inference efficiency 1.42×-3.9× when paired with optimized serving kernels, detailed algorithmic descriptions and implementation details are provided in Appendix A.2.2.

3.1.2 Format Alignment Training

To address the challenge of inconsistent instruction following and irregular output formatting in model-generated solutions, we therefore introduce a **format alignment stage**. This phase aims to instill in the model a structured, step-by-step reasoning workflow without altering its underlying mathematical knowledge. Crucially, this stage is designed to standardize presentation, *not* to inject new mathematical knowledge; it only teaches the model *how* to format an answer, not *what* to answer.

We use LoRA (Hu et al., 2021) for full-precision models and QLoRA (Dettmers et al., 2024) for quantized models to perform efficiently format alignment into model's latent space with minimal retraining. Training uses PRM800K (Lightman et al., 2023a) which provides 800K step-level annotations across 75K solutions to 12K problems. A fixed system prompt enforces explicit step boundaries and boxed final answers (see Appendix B),

producing traces with a uniform layout that enables reliable head-to-head comparison before and after quantization.

3.1.3 Detailed Examination of Reasoning Process

Qualitative Analysis. To systematically investigate quantization-induced degradation, we ground our error taxonomy in the cognitive process of mathematical problem-solving (Brown et al., 2016; Delastri and Lolang, 2023), specifically adapting Newman's Error Analysis (NEA) (Newman, 1977) and Hiebert's Conceptual vs. Procedural Knowledge framework (Hiebert and Lefevre, 2013) yielding four hierarchical error types: ***Conceptual***, ***Method***, ***Reasoning*** and ***Execution Errors*** ordered from high-level understanding to low-level execution, details of the defination and explanation of error types shown on Appendix C.

Quantitative analysis and error assessment pipeline. We automate error attribution with a three-stage automatic pipeline that combines LLM-based judgment with human-in-the-loop annotation:

1. **Gnerative Model Judgement:** For each instance in which a quantized model produces an incorrect answer, we employ a dedicated "expert model" to analyze the error. The expert model is tasked with: (a) identifying the first occurrence of an error, (b) specifying the exact step where the error is introduced, (c) assigning an error category based on a nested classification scheme, and (d) providing an explanation along with a confidence score for its determination.

2. **Majority Voting:** To curb hallucinations and improve evaluation reliability, we apply a three-stage majority-vote protocol to the outputs of five language models—*DeepSeek-R1* (Guo et al., 2025) (primary), GPT-4o, GPT-4, Qwen-Max (Yang et al., 2025), and DeepSeek-V3 (Liu et al., 2024). Instances of disagreement are flagged for further review, ensuring consistency and minimizing spurious judgments. **Rule1-Unanimous agreement**: If all four auxiliary models concur with the reference judgment from DeepSeek-R1, the answer is accepted. **Rule2-Simple majority**: If exactly three auxiliary models concur with DeepSeek-R1, the answer is likewise accepted. **Rule3-Escalation**: Otherwise, the instance is forwarded to two independent human annotators for arbitration.

3. **Human-in-the-loop Annotation:** For cases with conflicting assessments from the majority vote, we introduce two human annotators to manually review is conducted. The annotator need to follow the annotation document and review the explanations of five expert models then give the final assessment. Additionally, we also randomly sample 2% of the passed evaluated cases to verify the accuracy and consistency of the automated judgments.

The pipeline is intentionally conservative: disagreements are always escalated rather than resolved by tie-breaking heuristics. We designate DeepSeek-R1 as the primary judge because its reflective, step-by-step chain-of-thought proved best suited to localizing the first erroneous step; in a pilot study it also achieved the highest inter-annotator agreement with two human experts. Under this protocol, automated labels match final human judgments on 97.2% of 9,908 failure cases, with remaining discrepancies concentrated in borderline situations such as format-equivalent answers (e.g., $\frac{1}{2}$ vs. 0.5). Further implementation details and annotation guidelines are provided in Appendix C.

3.2 Restoring Reasoning Abilities in Quantized Models

The diagnostic findings above reveal two actionable regularities: (i) PTQ disproportionately increases method and execution errors rather than high-level conceptual mistakes, and (ii) these errors tend to surface at early reasoning steps and cascade to invalidate the remainder of the solution. Together, these patterns suggest that quantization damage is local—concentrated at specific procedural steps—rather than diffuse. This motivates a targeted intervention: if we can identify the first faulty step in each failure trace and provide a corrective signal from that point onward, a small amount of supervised tuning should suffice to restore the quantized model's reasoning trajectory. We implement this idea as a three-step locate $\rightarrow$ construct $\rightarrow$ tune procedure described below.

3.2.1 Silver Bullet Datasets Building

Building on the analysis in Section 3.1.3, we construct our evaluation subset by filtering and categorizing problem instances according to model error types. First, to eliminate any risk of data leakage, we remove all overlapping examples between the MATH and MATH-500 test sets by matching

Table 1: Comparison of quantization methods applied to the Llama-3, Qwen2.5, and Qwen3 model families. AWQ and GPTQ employ 4-bit weight and 16-bit activation quantization, whereas SQ (SmoothQuant) uses 8-bit weight and 8-bit activation quantization. Percentages in parentheses denote relative accuracy change from the FP16 baseline; positive values indicate improvement.

		GSM8K				MATH				AIME			
		FP16	INT4 AWQ	INT4 GPTQ	INT8 SQ	FP16	INT4 AWQ	INT4 GPTQ	INT8 SQ	FP16	INT4 AWQ	INT4 GPTQ	INT8 SQ
Llama	3.1-8B-Inst.	79.98	79.53 (0.56%)	78.85 (1.41%)	80.14 (+0.20%)	50.72	46.44 (8.44%)	46.04 (9.23%)	50.26 (0.91%)	10	1.11 (88.90%)	5.56 (44.40%)	6.67 (33.30%)
Llama	3.2-3B-Inst.	77.26	74.45 (3.64%)	71.57 (7.36%)	77.71 (+0.58%)	45.82	40.76 (11.04%)	42.66 (6.90%)	45.74 (0.17%)	4.44	0 (100%)	2.22 (50%)	4.44 (0%)
Llama	3.2-1B-Inst.	45.56	39.12 (14.14%)	39.58 (13.13%)	45.19 (0.81%)	20.58	16.2 (21.28%)	15.78 (23.32%)	20.96 (+1.85%)	3.33	0 (100%)	0 (100%)	0 (100%)
Qwen2.5	7B-Inst.	87.04	86.5 (0.62%)	85.14 (2.18%)	86.73 (0.36%)	72.48	69.84 (3.64%)	69.6 (3.97%)	72.04 (0.61%)	11.11	10 (9.99%)	8.89 (19.98%)	8.89 (19.98%)
Qwen2.5	3B-Inst.	81.35	79.68 (2.05%)	79.76 (1.95%)	81.27 (0.10%)	63.3	56 (11.53%)	55.02 (13.08%)	63.52 (+0.35%)	4.44	3.33 (25%)	3.33 (25%)	2.22 (50%)
Qwen2.5	1.5B-Inst.	68.23	61.11 (10.44%)	59.89 (12.22%)	68.46 (+0.34%)	43.74	25.6 (41.47%)	31.2 (28.67%)	43.52 (0.50%)	2.22	1.11 (50%)	0 (100%)	2.22 (0%)
Qwen2.5	0.5B-Inst.	43.37	27.07 (37.58%)	26.61 (38.64%)	41.55 (4.20%)	23.98	8.02 (66.56%)	7.24 (69.81%)	24 (+0.08%)	0	0 (–)	0 (–)	0 (–)
Qwen3	8B	90.6	90.67 (+0.08%)	90.22 (0.42%)	91.21 (+0.67%)	55.88	53.96 (3.44%)	53.58 (4.11%)	54.04 (3.29%)	35.83	6.67 (81.38%)	6.67 (81.38%)	33.33 (6.98%)
Qwen3	4B	90.52	90.98 (+0.51%)	88.55 (2.18%)	90.67 (+0.17%)	59.04	57.83 (2.05%)	56.03 (5.10%)	58.67 (0.63%)	35	6.67 (80.94%)	3.33 (90.49%)	30.83 (11.91%)
Qwen3	1.7B	85.82	81.8 (4.68%)	81.58 (4.94%)	85.75 (0.08%)	58.16	53.9 (7.32%)	46.75 (19.62%)	57.44 (1.24%)	24.17	10.83 (55.19%)	7.5 (68.97%)	19.17 (20.69%)
Qwen3	0.6B	72.02	58.45 (18.84%)	48.9 (32.11%)	70.06 (2.72%)	46.96	32.27 (31.28%)	20.87 (55.56%)	46.36 (1.28%)	10.83	2.5 (76.92%)	0 (100%)	8.33 (23.08%)

on their `unique_id` fields. Next, for each quantized model, we identify those problems that the full-precision counterpart answers correctly but on which the quantized variant fails, based on the models' majority-vote outputs. We then collect the corresponding problem prompts and model-generated responses for these failure cases. Finally, leveraging the labels produced by our error-assessment pipeline, we assign each case to its consensus error category for downstream analysis.

During the execution of our error-assessment pipeline, we identify and record the exact reasoning step at which each quantized model initially commits an error. Our qualitative analysis indicates that many reasoning failures originate from incorrect intermediate computations or boundary adjustments, on which all subsequent solution steps heavily depend. Leveraging this observation, we construct a targeted counterexample dataset by truncating the incorrect reasoning traces precisely at the identified erroneous steps. Subsequently, we prompt powerful baseline models (Llama-3.2-70B and Qwen2.5-Max) to resume and complete these truncated solutions until the correct answers are derived. Consequently, we designate the original quantized models' erroneous partial solutions as negative samples, while adopting the accurately completed solutions generated by the larger models as positive samples. This approach yields our "Silver Bullet" datasets, specifically designed to facilitate downstream error correction and model fine-tuning.

3.2.2 Capability Restoring Training

To reclaim the reasoning capability lost after low-bit quantization, we fine-tune each *quantized* model using **Direct Preference Optimization** (DPO) (Rafailov et al., 2024). Given a prompt x and a pair of responses (y^+, y^-) where y^+ is the correct answer and y^- is the quantized model's incorrect answer, y^+ is prefered to y^-, DPO maximizes the log-likelihood gap between the two while softly constraining the new policy π_θ toward the frozen reference policy π_{ref}. The objective is

$$\mathcal{L}_{\text{DPO}}(\theta) = \mathbb{E}_{(x,y^+,y^-)\sim\mathcal{D}}\Big[\log\sigma\big(\beta(r_\theta - r_{\text{ref}})\big)\Big],$$
$$r_\pi = \log\pi(y^+\,|\,x) - \log\pi(y^-\,|\,x). \quad (1)$$

where σ is the sigmoid function and β is an inverse-temperature hyper-parameter (we set $\beta = 1$). Because the reference gap is constant with respect to θ, maximizing $\mathcal{L}_{\text{DPO}}$ is equivalent to minimizing $\text{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)$ subject to pairwise preference constraints, thus yielding a stable, RL-free preference-alignment procedure with solid theoretical footing.

We realize the adaptation using LoRA and 4-bit QLoRA. Across all experiments, we set the LoRA rank to 32 for every injected adapter matrix and optimize with a cosine learning-rate schedule (base learning rate 1×10^{-6}, warm-up ratio 0.1) under a global batch size of 8. Training minimizes the sigmoid preference loss implied by $\mathcal{L}_{\text{DPO}}$.

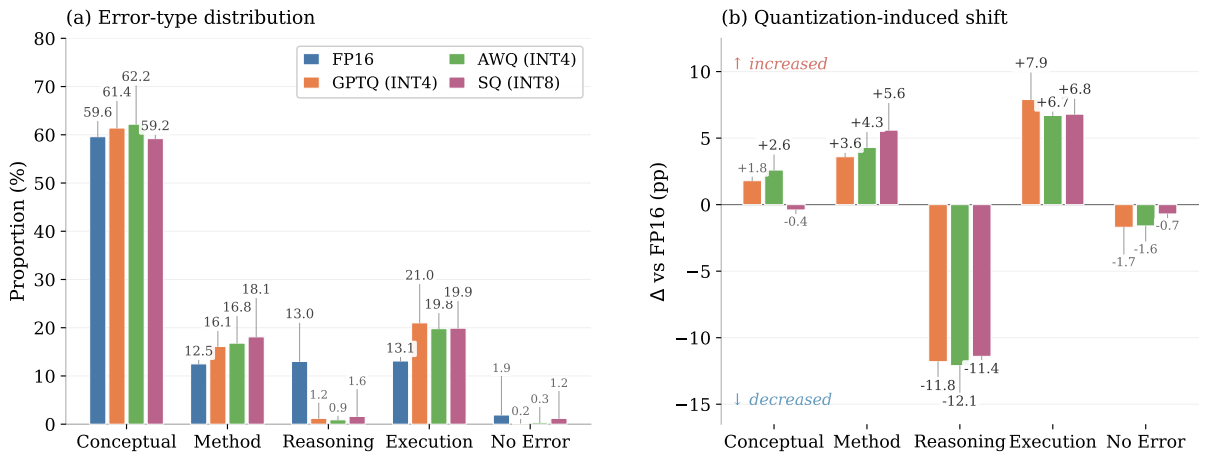

Figure 2: Error-type distribution and quantization-induced shift. (a) Proportion of each error type. For the full-precision model (FP16), we aggregate all problems it answered incorrectly; for each quantized model, we count only problems that FP16 solved correctly but the quantized variant failed, isolating errors attributable to quantization. (b) Shift in each error type relative to FP16, measured in percentage points. Across all three PTQ methods, Method and Execution errors increase substantially (+3.6 to +7.9 pp), while Conceptual errors remain largely unchanged. The steep drop in Reasoning errors (-11 to -12 pp) is a masking artifact: quantization induces earlier Method/Execution failures that are recorded as the first error, hiding downstream reasoning flaws from the statistics (see §4.2 for discussion).

4 Experiments

4.1 Evaluating Quantization Effects

We evaluate quantization-induced degradation on three math benchmarks of increasing difficulty: **GSM8K** (Cobbe et al., 2021) (grade-school word problems, 2–8 reasoning steps), **MATH** (Hendrycks et al., 2021) (competition-level problems across seven domains, $\geq$15 steps), and **AIME-22–25** (ChuGyouk, 2025) (olympiad-level, 120 problems). All runs share the same batch size and prompt used during format alignment. Results are reported in Table 1; two consistent trends emerge.

We maintaining consistency in hyper parameter and the prompt with those used during alignment and evaluation. This setup ensures a fair comparison across all models. According to the Table 1 we find these two trends:

Smaller-scale models suffer more severe losses in complex reasoning ability after quantization: Across every quantization methods, accuracy loss grows as model scale shrinks and smaller-scale models consistently demonstrate increased vulnerability to quantization. For instance, Qwen2.5-0.5B-Instruct suffers >60% relative drops on MATH under both AWQ and GPTQ, whereas Qwen2.5-7B-Instruct loses only 3–4%; the same pattern holds within the LLaMA-3 family. To rule out calibration-set bias we repeated the evaluation with varying calibration sizes and obtained consistent results, suggesting that smaller models have less capacity to absorb quantization-induced distributional shifts.

Performance degradation becomes more pronounced as the task complexity increases: On the simpler GSM8K, quantized models exhibited relatively minor accuracy losses on average 7.16% accuracy; on the harder MATH the average drop rises to 15.18%; and on the most challenging benchmark AIME, quantization frequently collapses performance to near zero. This monotonic trend indicates that multi-step reasoning chains amplify precision-loss errors, consistent with the step-level analysis in §4.2.

We also evaluate the quantization-induced degradation on both thinking-mode models and larger-scale models for mathematical reasoning tasks, the detailed results are reported in Appendix E.1. Together with the corresponding evaluations under the same quantization settings on general-purpose benchmarks, the detailed results are reported in Appendix E.2.

4.2 Error Taxonomy and Its Shift Under Quantization

Error profile of full-precision models. Using the assessment pipeline in §3.1.3, we examined every problem that the full-precision models answered incorrectly. *Conceptual Errors* were the most frequent (59.6%), while *Method*, *Execution*,

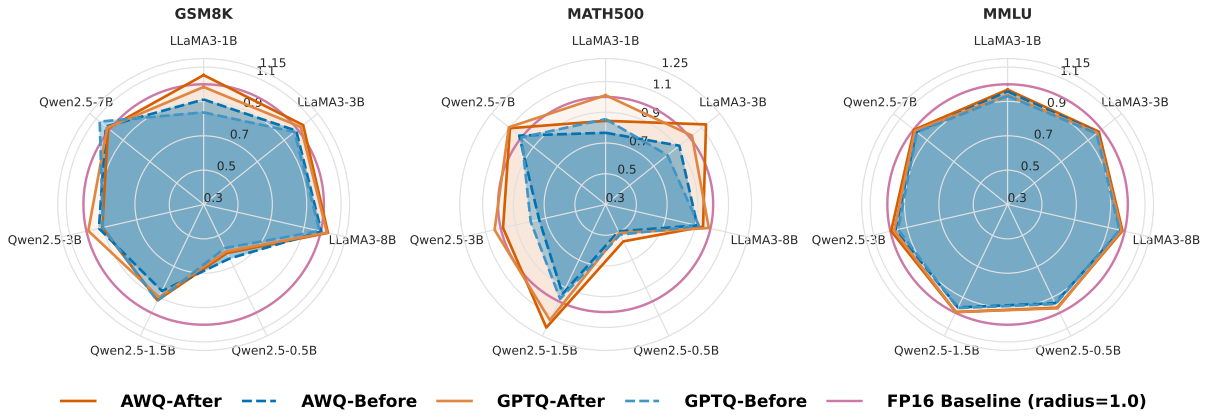

Figure 3: **Relative capability restoration with our method.** Radar values are normalized to each model's Vanilla-FP16 accuracy on the same benchmark (radius 1.0). Solid = After Restoration, dashed = Before Restoration (AWQ, GPTQ).

and *Reasoning* Errors appeared at comparable rates of 12.5%, 13.0%, and 13.1%, respectively.

Quantization-induced shift. We next isolate the problems that full-precision models solve correctly *but* their quantized counterparts fail. As shown in Figure 2, all three PTQ methods produce a consistent redistribution: Method Errors rise by 3.6–5.6 percentage points and Execution Errors by 6.7–7.9 pp, while Conceptual Errors remain largely unchanged. This confirms that quantization predominantly impairs procedural operations and arithmetic execution rather than high-level understanding—a pattern further supported by case-study evidence of increased difficulty with basic arithmetic and numerical computation (Appendix D).

Why Reasoning Errors seem to 'vanish'. Reasoning Errors decrease by 11–12 pp after quantization, but this is a *statistical masking effect* rather than a genuine improvement. Our protocol assigns each trajectory a single label based on the *first* erroneous step. Because quantization introduces earlier Method and Execution failures, these are recorded as the primary error, *'hiding'* downstream reasoning flaws that still exist later in the trace. Case inspection confirms that many trajectories labeled with other error types contain additional reasoning problems at subsequent steps. The shift therefore reflects how quantization moves the first-error frontier earlier, not an actual reduction in reasoning deficiencies. Absolute counts and representative examples are provided in Appendix D.

4.3 Capability Restoration

Our restoration experiments focus on the Qwen2.5-Instruct and LLaMA-3-Instruct families evaluated in Figure 2. Qwen3 models are included in the quantization degradation analysis but excluded from the restoration pipeline for a practical reason: their default thinking mode produces long internal reasoning traces with non-standardized step boundaries, which complicates both the format alignment stage and the step-level error attribution that our 'Silver Bullet' construction relies on.

To prevent data leakage during evaluation, we report results on **MATH-500** (Lightman et al., 2023b) which mirrors the original MATH benchmark in topic coverage and difficulty. Performance on MATH-500 thus reflects genuine reasoning recovery rather than memorization. We also measure accuracy on GSM8K and MMLU (Hendrycks et al., 2020) to assess how well the restored model generalises to other reasoning-intensive tasks. The results are visually presented in Figure 3, with additional details provided in Appendix F.

4.4 Ablation Study

To isolate the contributions of each component in our quantization recovery pipeline, we perform a series of ablation studies. Unless otherwise noted, all runs fix the training budget, optimizer, prompts, and decoding policy. We compare four variants of our training pairs (the *failure subset* refers to instances the quantized model answers incorrectly under baseline evaluation):

- **ALL-STEP:** all error cases from the failure subset; step-aligned supervision *resumes at the first-error step* (i.e., the first step where the model deviates from the gold solution) and continues step-wise along the gold trajectory to the final answer.

Table 2: Ablation results on GSM8K, MATH-500, and MMLU under INT4 quantization. Row labels denote training subsets: ALL-S = all error cases with step-aligned supervision from the first error; CE/ME/EE-S = only conceptual/method/execution errors with step alignment; ALL-NS = all error cases without step alignment. Numbers are accuracy (%). **Bold** = best per column.

		INT4 AWQ			INT4 GPTQ			
		GSM8K	MATH 500	MMLU	GSM8K	MATH 500	MMLU	Avg.
Llama-3B	ALL-S	**74.30**	**36.8**	**60.57**	73.01	**33.0**	59.90	**56.26**
	CE-S	73.19	35.4	60.42	**73.31**	31.6	**59.98**	55.65
	ME-S	73.62	32.0	60.45	72.40	30.4	59.86	54.79
	EE-S	73.39	31.2	60.47	72.63	31.2	59.95	54.81
	ALL-NS	70.74	27.8	60.25	69.45	15.0	59.94	50.53
Qwen2.5-3B	ALL-S	68.84	**38.4**	64.80	75.21	**40.6**	63.63	**58.58**
	CE-S	70.28	35.6	**65.00**	**75.28**	34.6	63.61	57.40
	ME-S	69.60	34.8	65.01	76.27	32.6	**63.81**	57.02
	EE-S	69.29	34.8	64.95	76.12	34.0	63.84	57.17
	ALL-NS	66.79	29.4	65.01	71.04	22.4	64.10	53.12

- **CE/ME/EE-STEP:** identical to ALL ERRORS-STEP but restricted to a single error type (*Conceptual-Error* / *Method-Error* / *Execution-Error*), with supervision resuming from the first-error step.
- **ALL-NONSTEP:** all error cases from the failure subset, but *without* resuming from the first-error step; positives are the full gold solutions.

4.5 Discussion

Synthesizing the results from § 4.3 and 4.4 together with the trends in Figure 2 and Table 2, we draw three main conclusions:

(i) Targeted recovery with our "Silver Bullet" datasets. Fine-tuning on the compact failure-targeted split substantially restores performance on MATH500 while also boosting GSM8K, and does so without hurting broad-domain reasoning as measured by MMLU. This intervention uses only a few hundred preference pairs and a few minutes of training on a single GPU, yet closes much of the gap between quantized and full-precision models. This confirms that the "Silver Bullet" provides *sample-efficient capability recovery* rather than *simple memorization* (Wu et al., 2024).

(ii) Quantization disproportionately erodes procedural and executional skills. Our error taxonomy shows that weight–activation quantization mainly increases *method* and *execution* errors—such as carrying out multi-step arithmetic or handling boundary conditions—rather than high-level conceptual reasoning. Because these mistakes often occur early, they propagate to invalidate otherwise correct derivations, explaining the steep drop on math-centric tasks.

(iii) Step-wise positives outperform naive alternatives. Under the strictly controlled budget of § 4.4, training on our error-targeted step-wise ALL split consistently outperforms the non-stepwise variant (NON-STEP) that supplies full gold solutions without restarting from the first erroneous step. On average, ALL improves accuracy by about 5.6 points over NON-STEP, with the largest gap of 6.2 points on MATH500. These results confirm that resuming the positive trajectory from the diagnosed first-error step, yields a sharper contrastive gradient that directly targets the quantization-induced failure frontier.

5 Conclusion

In this study, we presented a systematic study of quantization-induced degradation in mathematical reasoning. Across representative PTQ methods, model families, and math benchmarks, we find that quantization mainly increases method and execution errors, with failures often emerging early and then cascading through the remaining steps. Based on this diagnosis, we proposed a lightweight measure→locate→restore loop that targets the first faulty step with compact Silver Bullet preference pairs. In the evaluated Weight-4bit settings, this intervention substantially improves quantized models' mathematical reasoning while preserving PTQ efficiency. To summarize, our findings suggest that, within the studied regimes, quantization-induced degradation is better viewed as a localized process failure than only a drop in final-answer accuracy.

6 Limitations and Future Works

This study focuses primarily on mathematical reasoning benchmarks, including GSM8K and MATH-style problems, where solution trajectories are explicitly step-structured and intermediate errors are relatively easy to verify. We view this as a useful testbed rather than a complete characterization of quantization-induced reasoning degradation: mathematics provides a particularly controlled setting for studying error cascades because the dependence between steps is unusually strict and observable. Whether the same first-error dynamics transfer to other multi-step domains, such as code generation or broader commonsense reasoning, remains an important direction for future work.

A second limitation concerns model scale. Our experiments mainly cover small- to medium-scale open-source models that are practically relevant for constrained deployment. Although some larger-model results suggest that quantization degradation remains observable as scale increases, it is still unclear whether the same first-failure-step regularity holds in the same form for ultra-large models beyond the scales studied here. One plausible complication is that larger models may exhibit stronger parameter redundancy and error-masking effects, making local failures harder to identify even when final-answer degradation is still present. Extending the present analysis to 70B+ models is therefore an important next step.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ron Banner, Yury Nahshan, and Daniel Soudry. 2019. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32.

Janice Brown, Kim Skow, and Center IRIS. 2016. Mathematics: Identifying and addressing student errors. *The Iris Center*, 31.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

ChuGyouk. 2025. AIME_22_25 — hugging face datasets. https://huggingface.co/datasets/ChuGyouk/AIME-22-25. Accessed 2025-04-06.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Lusiana Delastri and Enos Lolang. 2023. Students' conceptual error and procedural error in solving algebraic problems. *Multicultural Education*, 9(1):18–24.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. QLoRA: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Guhao Feng, Kai Yang, Yuntian Gu, Xinyue Ai, Shengjie Luo, Jiacheng Sun, Di He, Zhenguo Li, and Liwei Wang. 2024. How numerical precision affects mathematical reasoning capabilities of llms. *arXiv preprint arXiv:2410.13857*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Elias Frantar, Roberto L Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. 2025. Marlin: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 239–251.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

James Hiebert and Patricia Lefevre. 2013. Conceptual and procedural knowledge in mathematics: An introductory analysis. In *Conceptual and procedural knowledge*, pages 1–27. Routledge.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023a. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023b. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-V3 technical report. *arXiv preprint arXiv:2412.19437*.

Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng Yu, Chun Yuan, and Lu Hou. 2025. Quantization hurts reasoning? an empirical study on quantized reasoning models. *arXiv preprint arXiv:2504.04823*.

Sandra P Marshall. 1995. *Schemas in problem solving*. Cambridge University Press.

Mary A Newman. 1977. An analysis of sixth-grade pupil's error on written mathematical tasks. *Victorian Institute for Educational Research Bulletin*, 39:31–43.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Bethany Rittle-Johnson and Robert S Siegler. 1999. Learning to spell: Variability, choice, and change in children's strategy use. *Child development*, 70(2):332–348.

Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. 2023. A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Transactions on Intelligent Systems and Technology*, 14(6):1–50.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonSenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Jaydev Tonde. 2026. The complete guide to llm quantization with vllm: Benchmarks & best practices.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2024. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1819–1862.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Congkai Xie, Shuo Cai, Wenjun Wang, Pengxiang Li, Zhijie Sang, Kejing Yang, Yiming Zhang, Zhen Li, Guanghao Zhu, Zeyu Liu, and 1 others. 2025. InfiR:

Crafting effective small language models and multimodal small language models in reasoning. *arXiv preprint arXiv:2502.11573*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. 2019. Quantization networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7308–7316.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Quantization Methods and Efficiency Gains

A.1 Quantization

A.1.1 AWQ

AWQ (Activation–Aware Weight Quantization) compensates for the long-tailed distribution of activations before the weight tensor is discretised. Let $\mathbf{A} \in \mathbb{R}^{B\times d}$ be the mini-batch activations and $\mathbf{W} \in \mathbb{R}^{d\times m}$ the corresponding weights. A positive scale vector $\boldsymbol{\gamma} \in \mathbb{R}^{d}_{+}$ is chosen such that

$$\begin{aligned}\widetilde{\mathbf{Y}} &= \big(\mathbf{A} \odot \boldsymbol{\gamma}^{-1}\big)\big(\boldsymbol{\gamma} \odot Q(\mathbf{W})\big)^{\top},\\ \gamma_k &= \big(\operatorname{mean}|\mathbf{A}_{,k}|^{\alpha}\big)\big(\operatorname{mean}|\mathbf{W}_{k,:}|^{-\beta}\big),\end{aligned}$$

where $(\alpha, \beta) \in [0, 1]$ control the balance between activation and weight magnitudes and $Q(\cdot)$ denotes an asymmetric 4-bit quantiser. Because the rescaling is folded back into $\mathbf{W}$, the forward pass is identical to the unscaled INT4 kernel and incurs no extra latency.

A.1.2 GPTQ

GPTQ formulates post-training quantisation as a blockwise least-squares problem over a small calibration set $\mathcal{C} = \{\mathbf{A}^{(i)}\}_{i=1}^{|\mathcal{C}|}$:

$$\widetilde{\mathbf{W}} = \arg\min_{\mathbf{W}' \in \mathcal{Q}} \sum_{i=1}^{|\mathcal{C}|} \big\|\mathbf{W}'\mathbf{A}^{(i)} - \mathbf{W}\mathbf{A}^{(i)}\big\|_F^2,$$

where $\mathcal{Q}$ is the set of weight tensors representable by the target bit-width. The optimisation proceeds greedily over 128-channel blocks. After quantising one block, GPTQ updates the remaining full-precision weights with a rank-r approximation of the corresponding Hessian inverse, cheaply computed from second-order activation statistics. This strategy yields near-optimal INT4 weights with negligible calibration cost.

A.1.3 SmoothQuant

SmoothQuant jointly scales activations and weights so that both can be represented with the same uniform INT8 format. For each output channel, a learned scale $\sigma_k > 0$ migrates range from activations to weights:

$$\widetilde{\mathbf{Y}} = \big(\mathbf{A} \odot \sigma^{-1}\big)\big(Q(\mathbf{W} \odot \sigma)\big)^{\top}.$$

The scales $\{\sigma_k\}$ are obtained by minimising the worst-case per-channel quantisation error across the calibration set, typically using a few thousand

tokens.[1] Once trained, the scales are fused into **W** and the model runs on standard INT8 kernels without auxiliary tensors or runtime re-scaling.

Implementation Notes. All three methods use per-channel affine quantisation. AWQ and GPTQ quantise weights to 4 bits while retaining 16-bit activations, whereas SmoothQuant produces a fully INT8 model. For calibration, we use Wikipedia to maintain distributional independence from downstream evaluation tasks and avoid contaminating the benchmark outcomes. We keep the original hyper-parameters recommended by the respective authors to ensure reproducibility across implementations.

A.2 Deployment Efficiency

A.2.1 Memory Footprint Reduction

Across model families and scales, W4A16 methods (AWQ, GPTQ) consistently reduce memory by around 62–66%, while W8A8 (SmoothQuant) achieves more moderate savings (42%). These results confirm that PTQ delivers substantial and consistent memory efficiency gains across practical deployment settings.

A.2.2 Inference Throughput Increasing

While our study primarily focuses on reasoning accuracy and capability restoration, the post-training quantization (PTQ) methods considered in this work are also known to provide substantial deployment benefits in memory footprint and inference efficiency when paired with optimized serving frameworks and kernels.

For weight-only quantization, AWQ (W4A16) has been shown to deliver strong end-to-end acceleration in practical systems. In particular, Lin et al. (2024) report that AWQ with the TinyChat framework achieves up to $3.9\times$ speedup over the HuggingFace FP16 baseline on RTX 4090 and Jetson Orin, runs $2.7\times$ faster than FP16 on RTX 4090. In addition, when deployed with the Marlin kernel, recent benchmark results running on an H200 GPU report 741 tok/s for AWQ, compared with 461 tok/s for the FP16 baseline, corresponding to a $1.6\times$ throughput improvement. GPTQ with the same Marlin backend achieves 712 tok/s on the same setup, or about $1.5\times$ over FP16. At the kernel level, Marlin itself was designed to accelerate FP16$\times$INT4 matrix multiplication and reports speedups of approximately $3.9\times$ relative to FP16 on NVIDIA A10 for batch sizes up to 16–32 (Frantar et al., 2025).

For weight–activation quantization, SmoothQuant (W8A8) also offers clear system-level gains. Xiao et al. (2023) show that SmoothQuant integrated with FasterTransformer achieves up to $1.56\times$ speedup together with roughly $2\times$ memory reduction, and reaches up to $1.51\times$ speedup on a single NVIDIA A100 GPU. The W8A8 models served with vLLM achieve about $1.6\times$ speedup at 5 QPS relative to the FP16 baseline using $4\times$A100 GPUs.

These results are complementary to the findings of our paper. They indicate that the PTQ models studied here are not only relevant because of their reasoning degradation patterns, but also because they retain tangible deployment advantages in throughput, latency, and memory efficiency. Consequently, restoring reasoning performance remains practically meaningful, since the recovered models continue to benefit from the efficiency gains that motivate low-bit deployment in the first place.

[1] We use 2 048 tokens, following Xiao et al. (2023).

Table 3: GPU memory footprint of quantized models on a single NVIDIA H800 (PyTorch 2.9, CUDA 12.8). We report weight memory (GB) and relative reduction compared to the FP16 baseline.

			AWQ (W4A16)		GPTQ (W4A16)		SQ (W8A8)	
Family	**Scale**	**FP16 (GB)**	GB	↓%	GB	↓%	GB	↓%
Qwen2.5	0.5B	0.92	0.43	53.6	0.43	53.6	0.59	36.0
	1.5B	2.88	1.07	62.8	1.07	62.8	1.69	41.1
	3B	5.75	1.92	66.5	1.91	66.7	3.21	44.2
	7B	14.19	5.22	63.2	5.21	63.3	8.11	42.8
	14B	27.51	9.34	66.0	9.31	66.2	15.30	44.4
Qwen3	0.6B	1.11	0.50	54.7	0.50	54.8	0.70	36.6
	1.7B	3.20	1.26	60.6	1.26	60.8	1.89	40.9
	4B	7.49	2.49	66.7	2.48	66.9	4.12	45.0
	8B	15.26	5.68	62.8	5.66	62.9	8.79	42.4
	14B	27.51	9.30	66.2	9.26	66.3	15.31	44.3
Avg. reduction			**62.3%**		**62.4%**		**41.8%**	

Table 4: Representative deployment efficiency results reported for PTQ methods evaluated in this work. Speedups are reported relative to corresponding FP16 baselines.

Method	**Efficiency Gain**	**Hardware**	**Source**
AWQ (TinyChat)	2.7×–3.9× vs. FP16	RTX 4090 / Jetson Orin	(Lin et al., 2024)
AWQ + Marlin	741 vs. 461 tok/s (1.6×)	H200	(Tonde, 2026)
GPTQ + Marlin	712 vs. 461 tok/s (1.5×)	H200	(Tonde, 2026)
Marlin kernel	∼3.9× vs. FP16	NVIDIA A10	(Frantar et al., 2025)
SmoothQuant	1.42×–1.56× vs. FP16	A100	(Xiao et al., 2023)

B Prompt

System Prompt

Your task is to solving mathematical problems by following these rules:

1. Follow the given instructions strictly and ensure the output aligns with the expected format.
2. Break down the reasoning process into clear, logical steps using numbered lists (e.g., 1., 2., 3.) to sequentially present each step. Each reasoning step should be isolated and clearly described to enhance readability and logical flow.
3. After completing all reasoning steps, provide the final answer.
4. Please reason step by step, and put your final answer within `\boxed{}`. The final answer must be formatted as (`\boxed{your\_final\_answer}`). For example: Answer: `\boxed{42}`.
5. Maintain consistent formatting across training and evaluation phases to ensure the model outputs meet evaluation requirements seamlessly.

Problem Statement (from PRM-800K Dataset Training Subsets)

The product of the proper positive integer factors of n can be written as $n^{(ax+b)/c}$, where x is the number of positive divisors n has, c is a positive integer, and the greatest common factor of the three integers a, b, and c is 1. What is $a + b + c$? **[Gold Answer: 1]**

Response Sample(from PRM-800K Dataset Training Subsets)

1. I notice that the problem involves the product of the proper positive integer factors of n, which means all the factors of n except n itself.
2. I also notice that the problem involves the number of positive divisors of n, which I'll call x.
3. I wonder if there is a way to relate the product of the proper factors of n to n and x.
4. I recall that the product of all the factors of n is $n^{x/2}$, since each factor can be paired with its

complement, and their product is n.
5. Therefore, the product of the proper factors of n is $n^{x/2}/n$, which is $n^{(x-2)/2}$.
6. This means that the product of the proper factors of n can be written as $n^{(ax+b)/c}$, where $a = 1$, $b = -2$, and $c = 2$.
7. The greatest common factor of a, b, and c is 1, as required.
8. The answer is $a + b + c = 1 + (-2) + 2 = 1$.
Answer: `\boxed{1}` **[Answer: 1]**

Evaluation Prompt

Role = `'HUMAN'`,
Prompt = "Your task is to solving mathematical problems by following these rules:

1. Follow the given instructions strictly and ensure the output aligns with the expected format.
2. Break down the reasoning process into clear, logical steps using numbered lists (e.g., 1., 2., 3.) to sequentially present each step. Each reasoning step should be isolated and clearly described to enhance readability and logical flow.
3. After completing all reasoning steps, provide the final answer.
4. Please reason step by step, and put your final answer within `\\boxed{}`. The final answer must be formatted as `\\boxed{your_final_answer}`. For example: Answer: `\\boxed{42}`.
5. Maintain consistent formatting across training and evaluation phases to ensure the model outputs meet evaluation requirements seamlessly.

Problem: `problem`"

Assessment System Prompt

You are a helpful assistant that identifies and classifies errors in mathematical reasoning steps.

You will be given:

- **Problem Statement:** A math problem statement.
- **Answers:** The right answer and answer from full-precision model and quantized model. Which model's answer is correct.
- **Full-Precision Reasoning:** The reasoning steps and final answer from a full-precision model.
- **Quantized-Model Reasoning:** The reasoning steps and final answer from a quantized model.
- **Error Type Definition:** The definition and explanation of error types.

Your task:

1. **Ground Truth Verification:** Compare both models' answers against the provided correct answer.
2. **Error Detection Protocol (Quantized Model):** If the quantized model is incorrect:
   1. Trace error origin using this hierarchy:

Conceptual	`conceptual_misunderstanding` `contextual_oversight`
Reasoning	`logical_reasoning_error`
Method	`procedural_error` `formula_rule_error`
Execution	`computational_error` `symbolic_manipulation_error`

   2. Identify first point of divergence from correct reasoning.
   3. Classify using the most specific applicable type.
   4. Provide step-specific evidence.
3. **Conflict Resolution:**
   1. If multiple types apply, choose the earliest in the hierarchy.
   2. If ambiguity persists, use `procedural_error` as default.

Return your analysis in the following JSON format strictly:

```
{
  "quantized_error_analysis": {
    "primary_error_type": ["..."],
    "error_step"         : 1,
    "explanation"        : "Short evidence from
      reasoning steps",
    "confidence_score"   : 0.7  // between 0.7 and 1.0
  }
}
```

C Human Annotation Guidebook

Purpose

This guideline specifies the manual verification protocol applied to *disagreement cases* that survive the automated evaluation pipeline—namely the expert-LLM judges and the five-model majority vote. Annotators produce the *final ground-truth verdict* (error type, error step, explanation, confidence) for every instance in which

(a) the majority vote conflicts with the baseline judge **DeepSeek-R1**, or

(b) a "passed" case is randomly drawn for audit ($\approx 2\%$ of all cases).

Materials Provided

- `problem.txt`: problem statement.
- `answers.json`: correct answer, full-precision answer, quantized answer.
- `fp_trace.txt`, `qt_trace.txt`: step-by-step reasoning traces.

- judge_outputs/: five JSON files—DeepSeek-R1 (*baseline*), DeepSeek-V3, GPT-4o, GPT-4, Qwen-Max—each containing primary_error_type, error_step, explanation, confidence_score.
- vote_summary.json: ensemble result, per-model confidences, disagreement flag.

Error-Type Taxonomy

Conceptual Errors	conceptual_misunderstanding
	contextual_oversight
Reasoning Errors	logical_reasoning_error
Method Errors	procedural_error
	formula_rule_error
Execution Errors	computational_error
	symbolic_manipulation_error

Earliest-precedence rule: when multiple labels apply, choose the first that appears in the above list.

Annotation Procedure

1. **Answer verification**. Confirm which model(s) yield the correct final answer. If both are wrong, mark the case dual_failure.

2. **Locate first divergence**. Read fp_trace and qt_trace in parallel and find the earliest step where the quantized trace deviates from valid reasoning.

3. **Review automated evidence**. Inspect the five judge outputs and majority-vote result.

4. **Decision**.

   4.1. Adopt the ensemble consensus if at least three judges agree *unless* compelling counter-evidence exists.

   4.2. Otherwise, perform an independent assessment using the taxonomy.

5. **Label assignment**. Record primary_error_type, error_step (1-indexed), explanation ($\leq$ 40 words, quote the critical step), confidence_score.

6. **Quality flag**. Set needs_second_opinion = true if residual uncertainty remains.

Confidence-Score Heuristic

- **0.90 – 1.00**: clear evidence; ≥ 4 judges concur.
- **0.80 – 0.89**: moderate certainty; majority concurs; minor ambiguity.
- **0.70 – 0.79**: plausible but alternate interpretations exist; split vote (3–2 or worse).

Output Schema

Annotators create human_verdict.json using

```
{
  "quantized_error_analysis": {
    "primary_error_type": "procedural_error",
    "error_step": 4,
    "explanation": "Applied quadratic formula with sign error
      at step 4.",
    "confidence_score": 0.83
  }
}
```

Dimension Definition

- **Conceptual Errors** occur when the model exhibits a fundamental misunderstanding of the underlying principles or relevant context of the problem (Marshall, 1995). This can manifest either as a conceptual misunderstanding, where the core ideas or foundational theories are not correctly grasped, resulting in an erroneous approach or framing of the problem; or as contextual oversight, in which crucial situational constraints or domain-specific factors (such as physical boundaries or geometric limitations) are overlooked, significantly distorting the solution process and its outcome.
- **Method Errors** refer to inaccuracies stemming from the improper selection or application of mathematical methods or established procedural approaches. Specifically, procedural errors happen when prescribed sequences or standard algorithms are incorrectly executed or entirely skipped, causing incomplete or invalid solutions (Rittle-Johnson and Siegler, 1999). Formula rule errors are another subtype, characterized by the misuse or misapplication of relevant mathematical theorems, formulae, or rules—such as applying a formula in an inappropriate context—which fundamentally undermines the validity of the resulting calculations or conclusions.
- **Execution Errors** arise during the process of mathematical computation and symbolic manipulation. They encompass computational errors involving incorrect arithmetic or algebraic operations, such as flawed summations, erroneous expansions, or factorization mistakes, thus jeopardizing the accuracy of final answers. Additionally, symbolic manipulation errors include improper handling or representation of symbolic expressions, variables, or transformations. This could involve mislabeling variables or misinterpreting symbolic forms, leading to an incorrect representation

and subsequent solution of the problem (Lightman et al., 2023a).

- **Reasoning Errors** involve flaws in the logical flow of problem-solving. Specifically, logical reasoning errors occur when there is a breakdown in the reasoning process itself, such that inference steps either do not logically follow one another or omit essential connections. This causes a logical gap or disconnect between the initial premises and the eventual conclusion, rendering the derived solution fundamentally flawed or unsupported.

Decision Aids

- *Conceptual misunderstanding*: misstates theorem before algebra begins.
- *Contextual oversight*: ignores domain restrictions or boundary conditions.
- *Logical reasoning error*: unsupported logical jump.
- *Procedural error*: applies an inappropriate solution method.
- *Formula rule error*: violates algebraic/derivative rule.
- *Computational error*: arithmetic slip.
- *Symbolic manipulation error*: incorrect simplification of an expression.

Quality Control & Ethics

- Two Annotators work independently; no discussion of live cases.
- Evaluate reasoning quality, not model identity; avoid bias.
- Flag any toxic or sensitive content present in traces.
- Project leads re-annotate 2 % of "agree" cases and all `needs_second_opinion` cases; overall agreement $< 95\%$ triggers targeted review.

Note.—DeepSeek-R1 is designated the baseline judge owing to its highest pilot agreement with human experts.

D Case Studies

D.1 Example of Reasoning Errors

In Problem #128, the GPTQModel made a logical reasoning error when calculating the minimum number of miles Suzanne could walk in February. While the model correctly identified that February has 28 or 29 days, it missed the 27th day when calculating her walking schedule every third day, leading to an incorrect conclusion of 8 walking days and 32 miles. In reality, the correct number of walking days is 9 (3, 6, 9, 12, 15, 18, 21, 24, 27), resulting in a total of $9 \times 4 = 36$ miles. This error highlights the model's logical reasoning gap in iterating through sequential intervals accurately. The case #128 shows the **Reasoning Errors**.

Problem #128

Suzanne walks four miles every third day. What is the fewest number of miles she can walk in February? [Gold Answer: 36]

Response of `LLaMA-3.1-8B-Instruct`

1. To find the fewest number of miles Suzanne can walk in February, I need to figure out how many days she walks four miles in that month.
2. February has 28 or 29 days, depending on whether it is a leap year or not.
3. I do not know if this is a leap year, so I will assume it has 28 days.
4. If Suzanne walks four miles every third day, she walks on the 3rd, 6th, 9th, 12th, 15th, 18th, 21st, 24th, and 27th of February.
5. That is nine days, so she walks a total of $9 \times 4 = 36$ miles in February.
6. This is the fewest number of miles she can walk; if she skipped any of those days she would have to walk on a different day, increasing her total.
7. Therefore, the answer is 36 miles. **[Answer: 36]**

Response of `LLaMA-3.1-8B-Instruct-W4A16` (GPTQ Model)

1. To find the fewest number of miles Suzanne can walk in February, I must determine how many days she walks during that month.
2. February has 28 days, but some years

have 29 days.

3. I will assume February has 28 days, since that is the most common case.
4. If Suzanne walks every third day, she will walk on days 3, 6, 9, 12, 15, 18, 21, and 24.
5. That is a total of 8 days, and she will walk 4 miles on each of those days.
6. Thus, the fewest number of miles she can walk in February is $8 \times 4 = 32$ miles. **[Answer: 32]**

D.2 Example of Execution Errors

In Case #93, the task was to find the value of c for which the circle defined by $x^2 - 10x + y^2 + 6y + c = 0$ has a radius of 1. The correct approach involves completing the square, resulting in $(x-5)^2 + (y+3)^2 = -c + 34$, and solving $-c + 34 = 1$ to find $c = 33$. However, the GPTQ-Model made a computational error by incorrectly rearranging the equation as $34 - c = 1$ leading to the wrong answer $c = 35$. This error stemmed from mishandling the sign of c during algebraic manipulation, highlighting the importance of careful sign management in equation solving. The case #93 shows the **Execution Errors**.

Problem #93

For what value of c will the circle with equation $x^2 - 10x + y^2 + 6y + c = 0$ have a radius of length 1? [Gold Answer: 33]

Response of `LLaMA-3.1-8B-Instruct`

1. To find the radius of a circle, I need to complete the square for both x and y terms in the equation.
2. I can rewrite the equation as $(x^2 - 10x) + (y^2 + 6y) = -c$.
3. To complete the square for $x^2 - 10x$, I add $(10/2)^2 = 25$ to both sides.
4. To complete the square for $y^2 + 6y$, I add $(6/2)^2 = 9$ to both sides.
5. This gives $(x^2 - 10x + 25) + (y^2 + 6y + 9) = -c + 25 + 9$.
6. I factor the left side as $(x-5)^2 + (y+3)^2$.
7. The right side simplifies to $-c + 34$.
8. Therefore the equation becomes $(x - 5)^2 + (y + 3)^2 = -c + 34$.
9. The radius of the circle is the square root of the right-hand side.
10. Setting the radius to 1 gives $-c + 34 = 1^2 = 1$.
11. Solving for c yields $c = 34 - 1 = 33$. **[Answer: 33]**

Response of `LLaMA-3.1-8B-Instruct-W4A16` (GPTQ Model)

1. To find the radius, I first complete the square for the x and y terms and rewrite the equation in the form $(x-h)^2 + (y-k)^2 = r^2$, where (h, k) is the centre and r is the radius.
2. I begin with the x terms, adding and subtracting the same value to form a perfect square.
3. Because the coefficient of x is -10, I add and subtract $\left(-10/2\right)^2 = 25$ to obtain $x^2 - 10x + 25 - 25$.
4. Factoring gives $(x-5)^2$ and the equation becomes $(x-5)^2 - 25 + y^2 + 6y + c = 0$.
5. I repeat the process for the y terms, adding and subtracting $\left(-6/2\right)^2 = 9$ to get $y^2 + 6y + 9 - 9$.
6. Factoring yields $(y+3)^2$ and the equation is now $(x-5)^2 - 25 + (y+3)^2 - 9 + c = 0$.
7. Combining constants, I have $(x-5)^2 + (y+3)^2 - 34 + c = 0$.
8. Setting $r^2 = 1$ requires $-34 + c = 1$; solving gives $c = 35$.
9. Hence $c = 35$. **[Answer: 35]**

D.3 Example of No Error

Despite the care taken in designing our extraction scripts, a small number of predictions remain hard to classify, leading to a residual "**No Errors**" category. A strong judge model can usually flag these edge cases; for instance, in case #3812, Deepseek-

R1 correctly returns **No Errors** after a meticulous comparison. We subsequently review such instances and update the final labels accordingly.

Most ambiguities stem from multiple notations for the same numeric value—particularly decimals versus fractions and natural versus programming syntax—e.g., $1/2$, 0.5, `\frac{1}{2}`, $5\mathrm{E}{-01}$, and 5×10^{-1}.

Problem #3812

What is the probability that when a fair 6-sided die is rolled, a prime number faces up? **Gold Answer:** $\frac{1}{2}$

Response of `LLaMA-3.1-8B-Instruct-W4A16` (GPTQ Model)

1. To find the probability of rolling a prime number on a fair 6-sided die, I need to know how many prime numbers appear on such a die and how many faces there are in total.
2. A prime number is a positive integer divisible only by 1 and itself. The prime numbers on a 6-sided die are 2, 3, and 5.
3. There are 3 prime numbers on a 6-sided die, so the probability of rolling a prime number is 3 out of 6, or $1/2$.
4. Converting this fraction to a decimal, I divide 1 by 2 to obtain 0.5.
5. Alternatively, multiplying 1 by 100 and dividing by 2 gives 50%.
6. Therefore, the probability of rolling a prime number on a fair 6-sided die is 0.5 or 50%. **[Answer: 0.5]**

Judgement of `Deepseek-R1` (Primary Expert Model)

The quantized model's answer (0.5) is numerically equivalent to the correct answer ($1/2$). All reasoning steps are logically valid and mathematically correct, with no conceptual, procedural, or computational errors detected. **[No_Error: []]**

D.4 Example of Conflicting Judgments

Case #342 illustrates our conflict-resolution protocol when the judge models disagree on an error label. We perform a five-way cross-model validation using Deepseek-R1, Deepseek-V3, ChatGPT, GPT-4o, and Qwen-Max. Each model independently assigns an error category to the quantized trace, and the final label is set by majority vote.

If the vote is inconclusive (e.g., a 2-2-1 split), a human annotator re-examines the example. The annotator consults (i) each model's confidence score, (ii) the accompanying explanations, and (iii) the step-by-step reasoning provided by Deepseek-R1, together with the raw model outputs. This double-check ensures that every ambiguous case receives a consistent, well-justified error type.

Problem #342

What integer x satisfies $\frac{1}{4} < \frac{x}{7} < \frac{1}{3}$?Gold Answer: 2

Response of `LLaMA-3.1-8B-Instruct-W4A16` (GPTQ Model)

1. To solve this inequality, I need to isolate x on one side.
2. I can start by multiplying both sides by 4, since 4 is a common denominator for the fractions.
3. This gives me $1 < \frac{4x}{7} < \frac{4}{3}$.
4. Now I can multiply both sides by 7, since 7 is a common denominator for the fractions.
5. This gives me $7 < 4x < \frac{28}{3}$.
6. To make the inequality easier to read, I can divide both sides by 4.
7. This gives me $\frac{7}{4} < x < \frac{7}{1}$.
8. I can simplify the fractions by writing them as mixed numbers or decimals.
9. This gives me $1.75 < x < 7$.
10. Since x is an integer, I need to find the integers that satisfy this inequality.
11. The smallest integer greater than 1.75 is 2.
12. The largest integer less than 7 is 6.

13. Therefore, the integers that satisfy the inequality are $2, 3, 4, 5, 6$. **[Answer: 2,3,4,5,6]**

Judgement of `Deepseek-R1` (Primary Expert Model)

When multiplying $4/3$ by 7, the correct result is $28/3 \approx 9.33$. In step 5 the quantized model treats $7 \times (4/3)$ as $28/3$ but, in step 7, fails to divide this by 4 correctly, giving an upper bound of 7 instead of $7/3 \approx 2.33$. **[Execution_Errors: computational_error]**

Judgement of `Deepseek-V3`

The quantized model incorrectly multiplies both sides by 4 instead of 7, the proper common denominator. This leads to a procedural error that propagates through the subsequent steps. **[Method_Errors: procedural_error]**

Judgement of ChatGPT

The model simplifies the inequality to $\frac{7}{4} < x < \frac{7}{1}$, overlooking that the upper limit should be $\left(\frac{28}{3}\right) \div 4 = \frac{7}{3}$, not $\frac{7}{1}$. **[Conceptual_Errors: contextual_oversight]**

Judgement of `GPT-4O`

The quantized model reports the range $\frac{7}{4} < x < \frac{7}{1}$ instead of the correct $\frac{7}{4} < x < \frac{7}{3}$, yielding an incorrect set of integer solutions. **[Method_Errors: procedural_error]**

Judgement of `Qwen-Max`

The model simplifies $\frac{7}{4} < x < \frac{28}{3}$ to $1.75 < x < 7$. The upper bound $\frac{28}{3}$ is approximately 9.33, not 7. **[Conceptual_Errors: conceptual_misunderstanding]**

E Experiment Results

E.1 Quantization on Larger Models

To strengthen the generality of our conclusions and bringing potential insights, we have conducted additional experiments on the Qwen3 series, which natively support a "thinking mode" as the default configuration. These models already integrate internal CoT-like mechanisms, making them representative of both standard and "thinking" (CoT) LLMs. We evaluated Qwen3-8B, Qwen3-14B, and Qwen3-32B under identical quantization settings (bit-width, calibration set, and hyperparameters) as other qutization methods.

These results yield several key insights:

- **Larger models show greater quantization robustness.** Accuracy degradation from full-precision to quantized versions diminishes significantly as model size increases, consistent with observations from other studies. We believe because larger models possess a richer parameter space, which grants stronger robustness to quantization-induced numerical errors when mapping from high-precision to low-precision data formats.
- **Mild regularization effects appear in quantized models.** You can find an interesting result that Qwen3-14B under GPTQ slightly outperforms its vanilla counterpart, suggesting that moderate compression may enhance generalization, a phenomenon also noted in recent quantization studies. Notably, this effect has also been frequently cited as one of the underlying reasons why Quantization-Aware Training (QAT) can sometimes improve the generalization ability of post-quantized models.
- **Scalability of the "Silver Bullet" principle.** The consistently small degradation across larger models supports our hypothesis that targeted recovery using compact, well-curated data is even more effective for high-capacity models with stronger learning abilities.

E.2 Additional Evaluation on General-Reasoning Benchmarks

To test whether the degradation patterns observed on mathematical reasoning generalize to other task types, we evaluate Qwen2.5-0.5B/1.5B/3B/7B-Instruct on five additional benchmarks: ARC Easy and ARC Challenge (grade-school science QA) (Clark et al., 2018), HellaSwag (commonsense inference) (Zellers et al., 2019), IFEval (instruction following) (Zhou et al., 2023), and CommonSenseQA (commonsense knowledge) (Talmor et al., 2019). For each model we compare the full-precision checkpoint with its AWQ and GPTQ vari-

Table 5: Performance of Qwen3 models on MATH under different quantization methods.

	Qwen3-8B			Qwen3-14B			Qwen3-32B		
	FP16	AWQ	GPTQ	FP16	AWQ	GPTQ	FP16	AWQ	GPTQ
MATH	55.88	53.96	53.58	63.52	62.28	63.12	66.92	65.72	65.94

ants; Table 6 reports per-model accuracy together with the cross-model average degradation.

As shown on Table 6 across all five benchmarks, the average absolute accuracy drop is 2.3–3.6 points (3.9–5.1% relative), substantially smaller than the 6.8-point (13.2% relative) drop on GSM8K and the 10.6-point (29.8% relative) drop on MATH observed under the same quantization settings (Table 1). This contrast supports our central claim that PTQ disproportionately impairs mathematical reasoning while leaving general language understanding and commonsense capabilities largely intact.

E.3 Subject-wise and Difficulty-wise Degradation on MATH

To better understand how quantization-induced degradation relates to problem structure, we leverage the rich annotations in the MATH dataset, which covers multiple subject domains (such as algebra, geometry, number theory and combinatorics) and five difficulty levels (Level 1 to Level 5). For each model scale and quantization method, we compute the distribution of errors across mathematical subfields and difficulty levels. This analysis connects quantization sensitivity with different types of reasoning and provides practical guidance for deploying quantized models under varying reasoning complexities.

From the subject-wise analysis in Table 7, we observe a clear and consistent trend across all model scales: quantization-induced degradation is not uniformly distributed over mathematical domains. Subfields that involve multi-step symbolic manipulation, such as Intermediate Algebra, Precalculus and more advanced algebraic transformations, show noticeably larger performance drops for both AWQ and GPTQ. In contrast, domains that rely more on direct recall or simpler numerical reasoning remain comparatively stable. This pattern suggests that low-bit perturbations disproportionately affect tasks that require long-range dependency tracking and precise arithmetic transformations, which is consistent with the step-level error analysis in the main paper.

From the difficulty-level analysis in Table 8, we see a monotonic increase in degradation as problem difficulty grows. Level 1 and Level 2 questions exhibit only marginal changes after quantization, whereas degradation becomes much more pronounced for Levels 3 to 5. For Level 5 in particular, the gap between full-precision and quantized models can exceed 6–10 percentage points even for larger models.

These findings provide empirical evidence for the following points:

- Conceptually demanding and algebraically heavy subfields of MATH are especially vulnerable to precision reduction.
- Higher-difficulty problems, which require deeper chains of reasoning, tend to accumulate quantization noise and errors more severely.

Overall, this analysis further supports our claim that post-training quantization disproportionately affects mathematical reasoning compared with general language understanding. We include these tables and observations in the appendix for completeness.

Table 6: Accuracy (%) of Qwen2.5-Instruct models on general benchmarks before and after quantization. The bottom rows report the average absolute and relative accuracy change across all model scales; for comparison, the corresponding averages on GSM8K and MATH from the main experiments are included.

Scale	Method	ARC-c	ARC-e	CSQA	HellaSwag	IFEval
0.5B	FP16	46.44	65.43	59.38	39.43	36.69
	AWQ	51.86	64.90	53.81	37.81	36.81
	GPTQ	43.39	50.97	50.37	36.47	32.01
1.5B	FP16	77.97	89.95	76.00	62.19	50.96
	AWQ	72.54	84.13	73.38	58.62	47.60
	GPTQ	71.86	86.42	71.42	60.19	46.04
3B	FP16	84.75	91.53	78.62	76.62	68.23
	AWQ	75.59	88.36	76.41	73.28	65.47
	GPTQ	78.64	89.77	77.56	73.52	64.87
7B	FP16	86.10	92.59	84.19	85.18	77.82
	AWQ	86.44	93.47	82.47	84.38	76.86
	GPTQ	84.75	92.24	83.95	83.75	76.86
Avg. abs. drop (pts)		−3.56	−3.59	−3.38	−2.35	−2.61
Avg. rel. drop (%)		−4.10	−4.76	−5.06	−3.95	−5.01
Comparison:						
GSM8K avg. drop				**−6.78 pts / −13.21%**		
MATH avg. drop				**−10.56 pts / −29.84%**		

F Detailed Capability Restoration Results

As shown in Table 9, after capability restoration using our *Silver Bullet* dataset, the quantized 4-bit models not only recover but even surpass the performance of their full-precision counterparts on the MATH benchmark. Meanwhile, performance on GSM8K remains stable, and accuracy on MMLU—a diverse benchmark covering various complex reasoning tasks—is also preserved. These results demonstrate that our *Silver Bullet* dataset effectively restores mathematical reasoning capabilities without compromising general-purpose abilities, highlighting both the effectiveness and generalizability of our approach.

Table 7: Distribution of errors across mathematical domains on MATH under different quantization settings. All values are percentages. All experiments are conducted on Qwen2.5-Instruct models.

Scale	Method	Error Distribution by Domain (%)						
		Number	Counting	Interm.	Algebra	Geometry	Prealg.	Precalc.
0.5B	FP16	11.59	10.33	20.48	18.65	10.30	16.02	12.63
	AWQ	11.26	9.82	18.77	22.24	9.84	16.68	11.39
	GPTQ	11.08	9.80	18.74	22.57	9.65	16.71	11.45
1.5B	FP16	11.31	10.67	22.98	15.79	10.67	13.73	14.86
	AWQ	11.39	10.26	20.16	19.16	10.62	15.72	12.68
	GPTQ	11.57	10.28	21.74	18.99	10.28	13.94	13.56
3B	FP16	10.40	10.35	24.85	12.91	11.94	12.07	17.49
	AWQ	10.74	10.78	25.05	13.92	11.13	12.29	16.09
	GPTQ	11.23	9.98	24.49	14.58	11.20	12.30	16.22
7B	FP16	7.96	10.31	26.28	11.75	12.87	11.91	18.91
	AWQ	9.58	9.88	25.89	11.87	12.22	11.67	18.90
	GPTQ	8.56	10.50	27.13	11.20	12.39	11.60	18.62

Notes: Interm. = Intermediate Algebra; Counting = Counting & Probability; Prealg. = Prealgebra; Precalc. = Precalculus. FP16 = full-precision baseline.

Table 8: Error distribution across MATH difficulty levels under different quantization settings for Qwen2.5-Instruct models. All values are relative percentages; Level 1 is the easiest and Level 5 is the hardest.

Scale	Method	Error Distribution by Level (%)				
		Level 1 (Easiest)	Level 2	Level 3	Level 4	Level 5 (Hardest)
0.5B	FP16	5.72	15.41	21.39	26.04	31.44
	AWQ	7.53	17.29	22.43	24.99	27.75
	GPTQ	7.17	17.35	22.44	24.99	28.05
1.5B	FP16	4.38	10.42	20.30	25.59	35.71
	AWQ	4.89	16.57	21.37	25.31	31.48
	GPTQ	4.41	14.22	21.03	23.82	36.53
3B	FP16	3.39	12.20	19.03	25.73	39.65
	AWQ	3.30	12.21	20.12	25.86	38.50
	GPTQ	3.16	15.27	19.52	26.50	35.55
7B	FP16	3.42	12.45	19.07	26.01	39.05
	AWQ	3.34	12.02	18.75	25.79	40.10
	GPTQ	3.38	12.00	19.31	25.19	40.12

Table 9: Capability restoration results on GSM8K, MATH500, and MMLU benchmarks across different model scales using our curated *Silver Bullet* datasets. **Full Precision** refers to the full-precision model after format alignment. **BF** indicates performance before restoration, while **AF** shows performance after applying our restoration pipeline.

		Llama-3-Inst.			Qwen2.5-Inst.			
Quantization	**Task**	**1B**	**3B**	**8B**	**0.5B**	**1.5B**	**3B**	**7B**
Full Precision	**GSM8K**	38.44	71.34	76.88	42.99	61.87	76.04	75.51
	MATH500	18	32.4	36.4	16.6	22.2	39	41.6
	MMLU	45.14	61.81	68.62	45.49	59.71	65.1	73.32
	AVG	**33.86**	**55.18**	**60.63**	**35.03**	**47.93**	**60.05**	**63.48**
AWQ-BF	**GSM8K**	35.03	70.58	77.1	27.9	53.15	70.36	77.63
	MATH500	13.8	29.6	33.2	8.2	21	29	42.4
	MMLU	43.26	60.08	67	42.65	57.65	63.16	71.77
	AVG	**30.70**	**53.42**	**59.10**	**26.25**	**43.93**	**54.17**	**63.93**
GPTQ-BF	**GSM8K**	32.15	69.67	76.27	25.02	57.09	68.54	81.12
	MATH500	15.4	26.4	33.6	8.6	21.8	31.2	41.6
	MMLU	42.07	59.49	66.44	42.91	57.86	62.09	71.49
	AVG	**29.87**	**51.85**	**54.94**	**25.51**	**45.58**	**53.94**	**64.74**
AWQ-AF	**GSM8K**	40.49	74.3	80.14	26.38	56.86	68.84	76.42
	MATH500	15.2	36.8	34.6	9.4	26.4	38.4	45.6
	MMLU	43.72	60.57	67.67	43.99	59.43	64.8	73.1
	AVG	**33.14**	**57.22**	**60.80**	**26.59**	**47.56**	**57.35**	**65.04**
GPTQ-AF	**GSM8K**	37.83	73.01	79.68	25.93	55.65	75.21	76.88
	MATH500	18.2	33	36	8.4	25.2	40.6	46
	MMLU	42.29	59.9	67.23	44.15	59.43	63.63	72.56
	AVG	**32.77**	**55.30**	**60.97**	**26.16**	**46.76**	**59.81**	**65.15**