

Quantization Inflates Reasoning: Token Inflation as a Hidden Cost of Low-Bit Reasoning Models

Xinyu Lian¹, Walid Krichene², Beichen Huang¹, Masahiro Tanaka³,
Olatunji Ruwase⁴, Li Zhang², Minjia Zhang¹

¹University of Illinois Urbana-Champaign, ²Microsoft, ³Anyscale, ⁴Snowflake

Abstract

Quantization is widely used to reduce the inference cost of large language models, but its effect on reasoning models is not fully captured by final-answer accuracy or per-token latency. We show that low-bit post-training quantization can introduce a hidden test-time compute cost: quantized reasoning models often generate longer chains of thought even when they still answer correctly. Across mathematical reasoning, code generation, scientific question answering, and agentic tool-use benchmarks, we find that INT4/INT3 quantization can preserve accuracy but increase reasoning-token usage, offsetting the expected per-token speedup. To measure this effect, we introduce the CoT Token Inflation Ratio, which compares reasoning length between quantized and full-precision models averaged across all evaluation benchmarks. We further show that token inflation is accompanied by behavioral changes in the reasoning trace, including more intermediate steps and greater semantic repetition. These changes translate into measurable end-to-end real-world serving penalties. Finally, we evaluate mitigation strategies and find that prompting and decoding-time sampling offer inconsistent accuracy-length trade-offs, while quantization-aware training shows more promise in reducing both accuracy degradation and token inflation. Our results suggest that reasoning-token usage should be reported alongside accuracy when evaluating quantized reasoning models.

1 Introduction

Recent large language models (LLMs) trained for step-by-step reasoning have achieved strong performance on mathematical reasoning, code generation, and scientific question answering (DeepSeek-AI, 2025; OpenAI, 2024). A key driver of this progress is the ability to spend more test-time computation: instead of producing a short answer directly, reasoning models generate long chains of

thought (CoT) that decompose the problem, explore intermediate steps, verify partial results, and sometimes revise earlier conclusions (Lightman et al., 2023; DeepSeek-AI, 2025; OpenAI, 2024; Kimi Team, 2025). This behavior improves accuracy on difficult tasks, but it also makes inference substantially more expensive. For reasoning models, deployment cost is no longer determined only by model size and per-token latency. It also depends on how many reasoning tokens the model chooses to generate.

Quantization is one of the most widely used techniques for reducing the inference cost of LLMs. Post-training quantization (PTQ) methods such as GPTQ (Frantar et al., 2022), AWQ (Lin et al., 2023), SmoothQuant (Xiao et al., 2022), ParoQuant (Liang et al., 2025) reduce model memory footprint and accelerate decoding with negligible training, and they have shown strong accuracy-efficiency trade-offs on LLM benchmarks. These results have made low-bit quantization a practical default for serving large models. However, most quantization evaluations measure final-answer accuracy and per-token efficiency. This evaluation protocol is incomplete for reasoning models, where the number of generated reasoning tokens is a major component of test-time compute.

In this paper, we show that low-bit post-training quantization can introduce a hidden cost for reasoning models: it can make models reason longer. Across mathematical reasoning, code generation, scientific question answering, and agentic tool-use benchmarks, we find that quantized models often preserve final-answer accuracy while generating longer chains of thought than their full-precision counterparts. This effect can offset, and in some cases outweigh, the per-token speedup expected from quantization. In other words, a quantized reasoning model can be faster per decoding step but slower end-to-end because it takes more tokens to reach the answer.

We call this phenomenon **reasoning-token inflation** and introduce the CoT Token Inflation Ratio (CTIR) to measure it. CTIR measures the relative change in reasoning-token count between a quantized model and its full-precision reference, averaged across all evaluation benchmarks.

Our analysis further shows that token inflation is not confined to newly incorrect responses. Quantized models often produce more intermediate reasoning steps and more semantically repetitive traces, revisiting or re-verifying prior reasoning even when final answers remain unchanged. These shifts impose measurable penalties on time to first visible token and serving throughput. We also examine mitigation strategies, finding inconsistent accuracy–length trade-offs for prompting and decoding-time controls, while quantization-aware training (QAT) is more promising for jointly reducing accuracy loss and token inflation in the evaluated settings. In summary, this paper makes three contributions:

- We study PTQ on SoTA reasoning models ranging from 4B to 30B parameters (dense and MoE) and quantitatively identify reasoning-token inflation as a hidden cost of low-bit post-training quantization. Across reasoning, code, and agentic tool-use benchmarks, quantized models can preserve final-answer accuracy while generating longer chains of thought, offsetting the expected per-token efficiency gains (§4).
- We introduce the CoT Token Inflation Ratio (CTIR), a metric for measuring quantization-induced increases in reasoning length relative to a full-precision reference. CTIR exposes an evaluation gap in standard quantization studies, which typically report accuracy and per-token latency but omit the amount of reasoning generated at inference time (§5).
- We analyze the behavioral consequences of token inflation. Quantized models often produce more intermediate reasoning steps and more semantically repetitive traces, and this increased reasoning length translates into higher end-to-end serving latency and cost (§6). We further evaluate mitigation strategies, finding that prompting and decoding-time sampling offer inconsistent accuracy-length trade-offs, while quantization-aware training is a promising direction (§7).

2 Background and Related Work

2.1 Quantization Methods for LLM Deployment

Quantization for LLM deployment. Post-training quantization (PTQ) and quantization-aware training (QAT) are two main paradigms for reducing LLM inference cost. PTQ converts a pre-trained model to low precision with little or no additional training (Badri and Shaji, 2023; Frantar et al., 2022; Lin et al., 2023; Cheng et al., 2024; Liang et al., 2025), while QAT simulates low-precision arithmetic during training or fine-tuning with quantize–dequantize operators (Esser et al., 2020; Liu et al., 2024a; Chee et al., 2025). Because training or fine-tuning large LLMs is expensive, most prior work has focused on PTQ methods that avoid additional training cost (Gadhikar et al., 2025; Arai and Ichikawa, 2025; Sharify et al., 2025; Liu et al., 2024b; Tseng et al., 2024; Shao et al., 2023). Representative methods include ZeroQuant (Yao et al., 2022), GPTQ (Frantar et al., 2022), AWQ (Lin et al., 2023), SmoothQuant (Xiao et al., 2022), QuIP# (Tseng et al., 2024), and SpinQuant (Liu et al., 2024b). These methods reduce memory footprint and memory traffic through reconstruction-based quantization, activation-aware scaling, outlier smoothing, rotations, or low-bit codebooks, and have become attractive for LLM deployment due to their accuracy–efficiency trade-offs. QAT can better adapt weights to low-bit constraints, but its training cost remains substantial at LLM scale, making PTQ the more common practical choice.

Quantization for reasoning models. Recent work has begun to study how quantization affects reasoning-heavy tasks. Li et al. (2025) evaluate common PTQ methods, including GPTQ, AWQ, and SmoothQuant, on mathematical reasoning and show that extreme low-bit settings can substantially degrade accuracy. They further propose lightweight fine-tuning for recovery. Liu et al. (2025) broaden the study across model families and quantization targets, including weights, activations, and KV cache, and observe that harder reasoning tasks suffer larger quantization-induced degradation. These studies show that reasoning models are more sensitive to quantization than standard LLM benchmarks reveal. Our work studies a different, deployment-critical effect. Prior work primarily evaluates final-answer accuracy, with limited analysis of how quantization changes reason-

ing length. Liu et al. (2025) consider CoT length in a limited setting of distilled models. In contrast, we analyze quantized reasoning models across a range of scales and architectures, including both dense and MoE models, and jointly study accuracy, reasoning-token usage, trace-level behavior, and serving efficiency. We show that PTQ can preserve final-answer accuracy while inflating the reasoning CoT span, creating a hidden test-time compute cost that accuracy-only evaluation misses. We further evaluate mitigation strategies spanning inference-time controls, reasoning-aligned calibration data, and quantization-aware training, showing that reducing token inflation requires preserving the model’s reasoning trajectory rather than merely shortening its output.

3 Experimental Setup

We evaluate how low-bit post-training quantization (PTQ) affects both the accuracy and generation behavior of reasoning models.

Models. We evaluate open-source reasoning models spanning dense and mixture-of-experts (MoE) architectures. The dense models include Qwen3-4B-Thinking-2507 (Qwen3-4B), Phi-4-Reasoning-Plus (Phi-4-14B). The MoE models include Qwen3-30B-A3B-Thinking (Qwen3-30B). This model set covers a broad range of scales, from small reasoning models to large MoE reasoning models, allowing us to study whether quantization-induced changes in reasoning behavior manifest across model size and architecture. The full model checkpoint information and architectural details are provided in Appendix D.2.

Benchmarks. Our benchmark suite covers mathematical reasoning, code generation, scientific reasoning, broad multi-step reasoning, and general language understanding. For mathematical reasoning, we use AIME25 and GSM8K. These benchmarks span grade-school word problems and high-difficulty competition problems. For code generation, we use LiveCodeBench (LCB) and MBPP, evaluating generated programs by functional correctness against held-out tests. For scientific and broad-knowledge reasoning, we use GPQA-Diamond. We additionally include BBH as a mixed multi-step reasoning benchmark. To compare against less reasoning-intensive workloads, we report HellaSwag, MMLU-Pro and PIQA as general benchmarks. Each benchmark is evaluated on 100 questions selected by shuffling the evaluation

split with seed 42 and taking the first 100; AIME25 is evaluated on its full set of 30 questions. Thus, instances are fixed before comparing precision and quantizers rather than selected from model outputs. Benchmark details are provided in Appendix D.1. All outputs are generated using vLLM with temperature 0. Greedy decoding removes sampling variation and makes each checkpoint deterministic, but it does not remove uncertainty due to the finite evaluation set. Appendix E therefore reports a complementary full-set, five-run temperature-0.6 check for Qwen3-4B, together with distribution- and correctness-aware CTIR statistics in the main analysis.

Quantization Methods. We focus on weight-only PTQ, a common low-bit deployment setting for LLM inference. This scope lets us isolate a widely used quantization setting and study whether lower-precision weights alone can change the reasoning behavior of LLMs. We evaluate representative PTQ methods, including *RTN* (round-to-nearest, the zero-cost baseline with no calibration), *GPTQ* (second-order, reconstruction-based weight quantization) (Frantar et al., 2022), *AWQ* (activation-aware weight quantization) (Lin et al., 2023), *HQQ* (calibration-free weight quantization) (Badri and Shaji, 2023), and *ParoQuant* (pairwise Givens-rotation quantization with channel-wise scaling that flattens per-channel magnitudes) (Liang et al., 2025). For each method, we apply INT4 and INT3 quantization with group size 128. INT4 represents a practical low-bit deployment target, while INT3 provides a more aggressive setting for testing whether reasoning-token inflation becomes stronger as quantization noise increases. BF16 inference serves as the full-precision reference throughout the paper.

4 When Accuracy Preservation Fails to Predict Efficiency

Low-bit PTQ often preserves final-answer accuracy. We begin with a broad evaluation of PTQ on reasoning models. Table 1 summarizes accuracy across reasoning and general benchmarks. To make the trends easier to interpret, we categorize absolute accuracy drops relative to BF16 into four bands: *negligible* (drop < 1), *small* (1–2), *moderate* (2–4), and *large* (> 4). At INT4, the majority of (model, quantizer, benchmark) cells fall into the negligible or small degradation bands. This is especially true for general benchmarks (e.g., HellaSwag, PIQA).

Model	Prec.	Quant.	Reasoning Tasks								General Tasks				
			AIME25	GSM8K	GPQA-D	BBH	LCB	MBPP	Avg-R	Drop-R↓	HellaSwag	PIQA	MMLU-Pro	Avg-G	Drop-G↓
Qwen3-4B	INT4	BF16	—	—	—	—	—	—	—	—	50.0	81.0	60.0	63.7	—
		RTN	50.0	91.0	58.0	84.0	19.0	65.0	61.2	11.3	51.0	83.0	62.0	65.3	−1.7
		GPTQ	73.3	95.0	68.0	87.0	16.0	90.0	71.6	0.9	49.0	77.0	56.0	60.7	3.0
		AWQ	80.0	95.0	63.0	89.0	18.0	94.0	73.2	−0.7	50.0	79.0	61.0	63.3	0.3
		HQQ	70.0	95.0	64.0	89.0	17.0	89.0	70.7	1.8	50.0	79.0	58.0	62.3	1.3
		PareQuant	80.0	96.0	64.0	87.0	19.0	91.0	72.8	−0.4	50.0	81.0	61.0	64.0	−0.3
	INT3	GPTQ	63.3	92.0	52.0	85.0	22.0	88.0	67.1	5.4	49.0	76.0	58.0	61.0	2.7
		AWQ	20.0	89.0	44.0	78.0	2.0	41.0	45.7	26.8	48.0	76.0	57.0	60.3	3.3
		HQQ	26.7	93.0	54.0	84.0	9.0	66.0	55.4	17.0	49.0	78.0	61.0	62.7	1.0
		PareQuant	43.3	96.0	50.0	84.0	34.0	91.0	66.4	6.1	50.0	77.0	57.0	61.3	2.3
Phi4-plus-14B	INT4	BF16	—	—	—	—	—	—	—	—	64.0	82.0	77.0	74.3	—
		RTN	60.0	26.0	61.0	16.0	0.0	38.0	33.5	4.6	62.0	81.0	76.0	73.0	1.3
		GPTQ	53.3	41.0	64.0	18.0	0.0	41.0	36.2	1.9	62.0	81.0	78.0	73.7	0.6
		AWQ	53.3	30.0	69.0	21.0	0.0	33.0	34.4	3.7	64.0	82.0	77.0	74.3	0.0
		HQQ	60.0	48.0	66.0	15.0	0.0	43.0	38.7	−0.6	65.0	81.0	78.0	74.7	−0.3
		GPTQ	50.0	32.0	51.0	15.0	0.0	22.0	28.3	9.7	60.0	82.0	75.0	72.3	2.0
	INT3	AWQ	50.0	53.0	59.0	12.0	0.0	45.0	36.5	1.6	62.0	81.0	77.0	73.3	1.0
		HQQ	33.3	37.0	49.0	12.0	1.0	36.0	28.1	10.0	61.0	82.0	74.0	72.3	2.0
Qwen3-30B	INT4	BF16	—	—	—	—	—	—	—	—	59.0	80.0	80.0	73.0	—
		RTN	83.3	95.0	70.0	90.0	22.0	93.0	75.6	−0.8	60.0	82.0	75.0	72.3	0.7
		GPTQ	86.7	95.0	74.0	85.0	17.0	90.0	74.6	0.2	59.0	80.0	80.0	73.0	0.0
		AWQ	83.3	96.0	72.0	90.0	18.0	85.0	74.1	0.7	59.0	78.0	78.0	71.7	1.3
		HQQ	86.7	97.0	74.0	92.0	27.0	87.0	77.3	−2.5	60.0	79.0	81.0	73.3	−0.3
		GPTQ	83.3	94.0	74.0	90.0	23.0	84.0	74.7	0.1	56.0	80.0	78.0	71.3	1.7
	INT3	AWQ	73.3	96.0	64.0	88.0	15.0	74.0	68.4	6.4	56.0	78.0	72.0	68.7	4.3
		HQQ	63.3	95.0	66.0	85.0	12.0	65.0	64.4	10.4	56.0	78.0	78.0	70.7	2.3

Table 1: Overall evaluation of 3-bit and 4-bit quantization for Qwen3 and Phi-4 on reasoning and general benchmarks. We report separate averages and drops for reasoning (Avg-R/Drop-R) and general (Avg-G/Drop-G) tasks. Green, yellow, orange, and red cells denote negligible (< 1), small (1–2), moderate (2–4), and large (> 4) accuracy drops, respectively. LCB denotes LiveCodeBench. *Phi4 scores 0% on LCB due to harness/format mismatches.

Larger reasoning models are also relatively robust: for Qwen3-30B, most INT4 reasoning-task drops remain within the negligible-to-moderate range (< 4 points). These results are consistent with the common view that INT4 PTQ can be a practical deployment strategy when judged by final-answer accuracy alone.

Reasoning-heavy tasks are more sensitive to quantization. Despite the overall robustness of INT4, Table 1 also shows a clear gap between reasoning and general benchmarks. Under the same quantization setting, general-task accuracy usually changes little, while reasoning benchmarks exhibit larger drops and more frequent moderate or large degradations. The degradation is concentrated on harder, long-horizon tasks such as AIME25, where small perturbations can compound over many reasoning steps. This suggests that final-answer accuracy on general benchmarks is not sufficient to characterize quantization quality for reasoning models.

Accuracy degradation grows under more aggressive quantization. The accuracy gap becomes more pronounced as precision decreases from INT4 to INT3. Smaller reasoning models are particularly sensitive: for example, Qwen3-4B suffers a large absolute drop on AIME25 under INT4, and INT3 further increases degradation across difficult reasoning tasks. Larger models are more tolerant of INT4 noise, but they are not immune to aggres-

sive quantization. Across models, INT3 consistently widens the gap between reasoning and general benchmarks, indicating that reasoning-heavy workloads are a more sensitive stress test for low-bit quantization than general-purpose evaluations. **Accuracy preservation does not imply end-to-end efficiency.** The above results might suggest that INT4 PTQ is broadly effective: accuracy is often preserved, especially for larger models, and low-bit weights should reduce memory traffic and improve per-token decoding efficiency. However, this conclusion is incomplete for reasoning models. *End-to-end latency depends not only on the cost of generating each token, but also on the number of tokens generated before the final answer.* The commonly used W4A16 Marlin kernel achieves a 1.2–1.4 $\times$ tokens/s speedup over BF16 at batch sizes 1–16,¹ but the controlled serving experiment in §6 shows that longer reasoning can offset this local gain.

This motivates the central question of the next section: does quantization change how much reasoning the model generates?

5 Quantization Inflates Reasoning Tokens

The previous section showed that low-bit PTQ can preserve final-answer accuracy but fail to deliver

¹Tested in vLLM with input_length=512 and max_new_tokens=128.

Model	Prec.	Quant.	Reasoning Tasks										CTIR Avg.		
			AIME25		GSM8K		GPQA-D		BBH		LCB			MBPP	
Qwen3-4B	BF16	—	21307	—	1139	—	7796	—	2674	—	13459	—	4692	—	—
	INT4	RTN	24178	+13.5%	1548	+35.9%	10596	+35.9%	5470	+104.6%	17574	+30.6%	6310	+34.5%	+42.5%
		AWQ	22263	+4.5%	1108	−2.7%	7248	−7.0%	2866	+7.2%	14930	+10.9%	6400	+36.4%	+8.2%
		GPTQ	24388	+14.5%	1157	+1.6%	8565	+9.9%	3417	+27.8%	13930	+3.5%	5380	+14.7%	+12.0%
		HQQ	21795	+2.3%	1202	+5.5%	7626	−2.2%	2995	+12.0%	13303	−1.2%	5597	+19.3%	+6.0%
	ParoQuant	21652	+1.6%	1136	−0.3%	7744	−0.7%	2905	+8.6%	14416	+7.1%	5240	+11.7%	+4.7%	
	INT3	AWQ	30924	+45.1%	1529	+34.3%	13876	+78.0%	7599	+184.2%	61590	+357.6%	49609	+957.4%	+276.1%
		GPTQ	25402	+19.2%	1456	+27.8%	8912	+14.3%	3946	+47.6%	23773	+76.6%	9704	+106.8%	+48.7%
		HQQ	26038	+22.2%	2508	+120.2%	16599	+112.9%	7392	+176.4%	49690	+269.2%	54018	+1051.3%	+292.1%
		ParoQuant	23814	+11.8%	1241	+9.0%	10148	+30.2%	2883	+7.8%	16124	+19.8%	7918	+68.8%	+24.5%
Qwen3-30B	BF16	—	18587	—	865	—	6071	—	1692	—	15374	—	3330	—	—
	INT4	RTN	23613	+27.0%	1098	+27.0%	7188	+18.4%	2177	+28.7%	28088	+82.7%	5809	+74.4%	+43.0%
		AWQ	20103	+8.2%	996	+15.2%	6743	+11.1%	2096	+23.9%	17063	+11.0%	3902	+17.2%	+14.4%
		GPTQ	18277	−1.7%	912	+5.5%	6671	+9.9%	1853	+9.5%	15355	−0.1%	3302	−0.8%	+3.7%
		HQQ	19844	+6.8%	895	+3.6%	6323	+4.1%	1678	−0.8%	15751	+2.5%	3569	+7.2%	+3.9%
	INT3	AWQ	21827	+17.4%	982	+13.6%	7352	+21.1%	2194	+29.7%	22131	+43.9%	5700	+71.2%	+32.8%
		GPTQ	18857	+1.5%	964	+11.5%	7233	+19.1%	1932	+14.2%	17583	+14.4%	5114	+53.6%	+19.0%
		HQQ	21183	+14.0%	1036	+19.9%	8380	+38.0%	2388	+41.1%	30812	+100.4%	8235	+147.3%	+60.1%

Table 2: Average CoT tokens and CTIR across reasoning tasks. For each task, the two columns report CoT tokens and CTIR, respectively.

the expected end-to-end efficiency gains on reasoning workloads. We now show that the missing factor is output length. Quantized reasoning models often generate longer chains of thought than their full-precision counterparts.

CoT Token Inflation Ratio. To quantify this effect, we introduce the *CoT Token Inflation Ratio* (CTIR). Let M_{org} be the BF16 model and M_q be its quantized counterpart. For an instance x , we define the CoT length as

$$\ell_{\text{cot}}(M, x) := |\text{tok}(y_{\text{cot}})|. \quad (1)$$

For a dataset $\mathcal{D}$, we define CTIR as the relative increase in average CoT length:

$$\bar{\ell}(M; \mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ell_{\text{cot}}(M, x). \quad (2)$$

$$\text{CTIR}(M_q; \mathcal{D}) := \frac{\bar{\ell}(M_q; \mathcal{D}) - \bar{\ell}(M_{\text{org}}; \mathcal{D})}{\bar{\ell}(M_{\text{org}}; \mathcal{D})} \times 100\%. \quad (3)$$

For the benchmark collection $\mathcal{B}$, the value reported as *CTIR Avg.* is the macro-average

$$\text{CTIR}_{\text{macro}}(M_q; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{\mathcal{D} \in \mathcal{B}} \text{CTIR}(M_q; \mathcal{D}). \quad (4)$$

Each task therefore receives equal weight after normalization by its own BF16 mean; longer-CoT tasks do not receive more weight. To test sensitivity to runaway generations, we also report the median paired instance-level relative length change and a 10%-trimmed statistic. To separate deployment cost from correctness, *both-correct CTIR* restricts

each task to instances answered correctly by both BF16 and the quantized model before applying the same macro-averaging rule.

Token inflation persists beyond failures and outliers. Table 2 retains the original per-task CoT-token and CTIR breakdown. For Qwen3-4B, the full macro, full median, and 10%-trimmed CTIR triplets are respectively 42.5/38.6/27.4% for INT4 RTN, 8.2/2.8/3.1% for INT4 AWQ, 12.0/8.2/8.2% for INT4 GPTQ, 276.1/106.5/220.2% for INT3 AWQ, and 48.7/40.1/38.1% for INT3 GPTQ. All median and trimmed values remain positive, showing that the aggregate increase is not solely produced by a few runaway generations. On the strict both-correct subset, INT3 GPTQ retains positive macro CTIRs of 53.9% for Qwen3-4B and 22.6% for Qwen3-30B; the other evaluated configurations are also positive. Thus, inflation is not confined to newly incorrect responses. Full-dataset CTIR remains the appropriate systems metric because correctness is unavailable at serving time and every generation consumes resources, whereas both-correct CTIR supports the narrower behavioral claim that an otherwise successful trajectory can become longer.

Across the full results, INT4 usually produces mild inflation whereas INT3 is more severe, particularly for the smaller model. The largest Qwen3-4B INT3 AWQ/HQQ values coincide with large accuracy drops, so we do not use those cells alone to claim accuracy-preserved inflation. GPTQ generally produces smaller CTIR than AWQ and HQQ, although it does not eliminate the effect. Ap-

Model	Quant.	Acc.	CoT Tok.	Tool Calls
Qwen3-4B	BF16	52.12%	6556	7.56
	GPTQ-INT4	50.88%	7425	7.85
	GPTQ-INT3	31.38%	7455	7.36
Qwen3-30B	BF16	55.88%	5351	7.44
	GPTQ-INT4	54.12%	5488	7.57
	GPTQ-INT3	47.25%	6611	7.46

Table 3: BFCLv4-Multiturn evaluation on Qwen3-4B and Qwen3-30B models.

pendix G further shows output-length inflation for three newer model families; because those experiments use third-party quantized checkpoints and do not include a matched correctness analysis, we treat them only as a scope check.

Reasoning-token inflation can be even more costly in agentic workloads, where a model may perform multiple rounds of reasoning interleaved with tool calls. In such settings, latency is affected by both the number of tool invocations and the amount of reasoning generated around each invocation. We evaluate Qwen3-4B and Qwen3-30B on BFCLv4-Multiturn using GPTQ with a 4096-token reasoning budget per step. Table 3 presents the results and highlights that quantization causes clear token inflation in agentic workloads, while leaving tool-use behavior nearly unchanged. For example, GPTQ-INT4 increases Qwen3-4B’s CoT tokens by 13.3% with only a small change in tool calls, from 7.56 to 7.85. For Qwen3-30B, GPTQ-INT3 increases CoT tokens by 23.5%, while tool calls remain almost unchanged, from 7.44 to 7.46. This indicates that the E2E latency increase under quantization mainly comes from additional generated reasoning tokens rather than more frequent tool invocations.

6 Anatomy of Reasoning-Token Inflation

Inflation appears as more reasoning steps. To unpack where the extra tokens come from, Fig. 1 compares per-instance CoT step counts under BF16 and INT3 (GPTQ) for Qwen3-4B and Qwen3-30B on MBPP as an illustrative example. The majority of instances fall above the parity line, indicating that INT3 (GPTQ) generates more intermediate reasoning steps than BF16. Together with the both-correct analysis, this suggests a systematic trajectory change rather than an effect confined to failed instances.

Inflation increases step-level repetition. Longer traces are not necessarily worse if the extra tokens correspond to useful additional reasoning. Manual

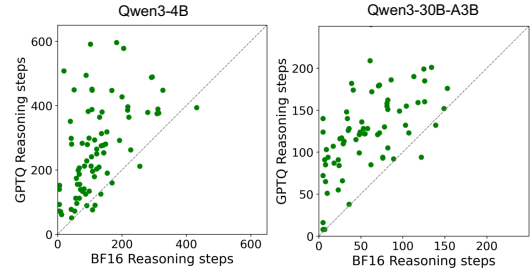


Figure 1: Per-instance CoT step counts under BF16 vs. INT3 GPTQ for Qwen3-4B (left) and Qwen3-30B (right) on MBPP. Each point plots BF16 steps (x-axis) against GPTQ steps (y-axis); the dashed line indicates parity. Points above the diagonal indicate more reasoning steps under quantization.

inspection of reasoning traces reveals what appears to be an increase in *repetitiveness* of the model. This can manifest in a number of ways: increased self-doubt and verification (e.g., “wait”, “hold on”, “let me verify”), back-tracking and retries, or general repetition of prior CoT steps (e.g., “again”, “once more”). These behaviors resemble the reflective style of reasoning models, but under quantization they can appear more frequently without improving final-answer correctness.

To quantify this effect, we extract the text within `<think>...</think>` (falling back to the full response when the tags are absent) and segment it into non-empty paragraphs separated by blank lines, matching our step-counting rule. Given steps $c_1, \dots, c_k$, we embed each step with Qwen3-Embedding-4B and compute pairwise cosine similarity $s(c_i, c_j)$. A step is repeated if another step has similarity at least $\tau = 0.8$. Finally, the repetitiveness of c is the fraction of repeated steps. Formally,

$$\text{Rep}(c) = \frac{1}{k} |\{i : \exists j \neq i, s(c_i, c_j) \geq \tau\}|.$$

Fig. 2 shows that many instances lie above parity, indicating more near-duplicate steps under quantization. We additionally count a fixed set of verification and self-doubt markers (Appendix F). Most configurations increase the raw marker count, although this descriptive statistic partly reflects longer responses and should not be interpreted as an increase in marker density. Extra reasoning is therefore not uniformly harmful, it can occasionally accompany a corrected answer, but it is not cost-free at serving time.

Identical-prefix diagnostics suggest increased local uncertainty. We evaluate BF16 and quan-

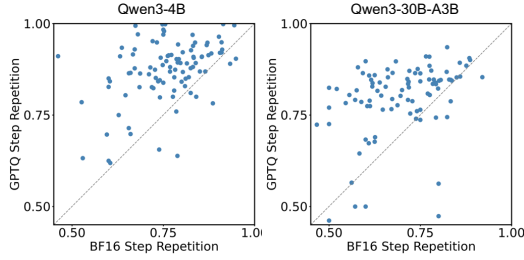


Figure 2: Per-instance CoT step repetitiveness (higher means more repetition) under BF16 vs. INT3 GPTQ for Qwen3-4B and Qwen3-30B on MBPP. Each point plots BF16 (x-axis) against INT3 GPTQ (y-axis); the dashed line indicates parity ($y = x$). Points above the diagonal indicate more repeated steps under quantization.

Model (quant.)	Task	D_{KL}	ΔH	Δ margin
Qwen3-4B (INT4 RTN)	BBH	0.162	0.030	-0.691
	GSM8K	0.131	0.042	-0.146
	MBPP	0.125	0.033	-0.850
Qwen3-30B (INT3 RTN)	BBH	0.399	0.204	-3.306
	GSM8K	0.261	0.109	-4.972
	MBPP	0.434	0.132	-2.961

Table 4: Next-token diagnostics under identical prefixes from BF16 traces. We report $D_{KL}(p_{BF16}||p_q)$, the entropy change $H(p_q) - H(p_{BF16})$, and the change in the top-two logit margin (quantized minus BF16). Positive entropy changes and negative margin changes indicate a less concentrated next-token distribution.

tized next-token distributions under identical prefixes sampled from BF16 reasoning traces. Holding the prompt and reasoning history fixed isolates the immediate distributional perturbation caused by changing the weights. Table 4 shows positive KL divergence, higher entropy, and smaller top-two margins in every tested setting. This supports a plausible mechanism in which local perturbations make a greedy trajectory more likely to enter verification or reconsideration branches over many decoding steps. It is supporting evidence rather than a causal proof, and it does not fully explain differences among quantizers.

Modest token inflation can erase quantization speedups. The trace-level changes above have direct serving implications. In reasoning-model deployments, the chain of thought may be generated internally before the final answer is exposed to the user. In this setting, longer reasoning traces increase the time to first visible token, even if the per-token decoding cost is lower under quantization. Longer traces also occupy decoding slots for more steps and increase KV-cache residency time, reducing effective serving capacity under continuous batching.

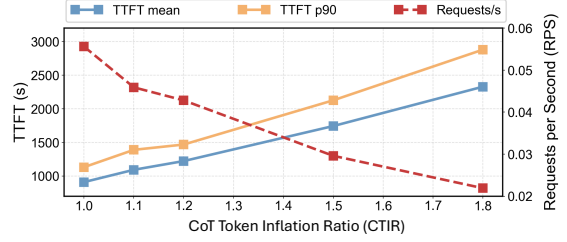


Figure 3: Impact of CoT length on serving efficiency. As reasoning length increases, time-to-first-token (TTFT; mean and p90) rises while vLLM throughput (requests/second) decreases.

We measure this effect using vLLM. We run Qwen3-30B-A3B on an H100 GPU with a 1k-token prompt and batch size 64. To isolate the effect of reasoning length, we control the CoT budget and measure time to first visible token (TTFT) and throughput. Here, the first visible token refers to the first user-facing output after the hidden reasoning span, so TTFT captures the user-perceived delay caused by generating the reasoning trace. Fig. 3 shows that even modest CoT inflation can substantially affect serving performance. A controlled 10% increase in CoT length raises p90 time to first visible token from 1050 s to 1400 s, an extra 350 s (33%) of latency for tail users in this batch-64 stress test. Longer generations also hold decoding slots and expand KV-cache footprints, squeezing concurrency under continuous batching: in the same setting, throughput drops from 0.055 to 0.047 requests/second. These results show why reasoning-token inflation matters in practice: the local efficiency gains of quantization can be offset when the quantized model generates more reasoning tokens.

7 Can We Reduce Reasoning-Token Inflation?

We now study whether token inflation can be reduced. We consider three classes of interventions with different costs: applying inference-time controls through decoding or prompting, choosing better calibration data for PTQ, and using quantization-aware training.

7.1 Inference-Time Controls Trade Accuracy for Shorter Reasoning

A natural way to reduce reasoning token inflation is to intervene at inference time, either by discouraging repetition during decoding or by prompting the model to reason more concisely. These methods are attractive because they do not require retraining or re-quantizing the model.

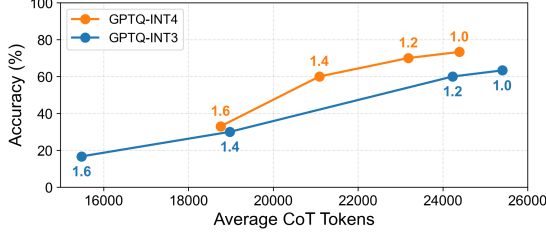


Figure 4: GPTQ quantized Qwen3-4B evaluation on AIME25, showing a trade-off between CoT token and accuracy under different repetition penalty settings.

(i) **Repetition penalty.** We use the `repetition_penalty` provided in vLLM, which discourages the model from generating tokens that have already appeared by rescaling their logits before sampling:

$$z'_i = \begin{cases} z_i/r, & i \in \mathcal{S}, z_i > 0, \\ z_i \cdot r, & i \in \mathcal{S}, z_i \leq 0, \\ z_i, & i \notin \mathcal{S}, \end{cases}$$

where z_i is the original logit, r is the repetition penalty, and $\mathcal{S}$ denotes the set of tokens that have appeared in the prompt or previously generated output. The default `repetition_penalty` is 1, and we test values $r \in \{1.1, 1.2, 1.3, 1.4\}$ on Qwen3-4B and evaluate the performance on AIME25 as reported in Fig. 4. Increasing the repetition penalty reduces the average CoT tokens, indicating that stronger repetition suppression can mitigate token inflation during reasoning. However, this token reduction often comes with a clear accuracy cost. These results suggest that repetition penalty is an effective but delicate control knob: it can reduce redundant reasoning tokens, yet overly strong penalties may also suppress useful intermediate reasoning steps and harm final-answer accuracy.

(ii) **Conciseness prompting.** Beyond decoding-time penalties, we further examine whether token inflation can be mitigated through prompting alone. Specifically, we test a set of lightweight prompt-based interventions that explicitly encourage shorter reasoning traces. The prompts are summarized as "*do not think*", "*less tokens*", and "*no restate*", and details are provided in Appendix C. We conduct this study on the same Qwen3-4B + GPTQ setup used in the sampling-penalty experiments and evaluate on MBPP. The results are reported in Table 5, which shows that prompt-based interventions are a simple heuristic for reducing CoT length, but they often trade off accuracy and behave incon-

	Quant.	Prompt	Acc.	CoT Tok.
MBPP	INT4	—	90.0%	5380
		<i>do not think</i>	90.0%	4496
		<i>few tokens</i>	87.0%	5211
		<i>no restate</i>	81.0%	3767
	INT3	—	88.0%	9704
		<i>do not think</i>	83.0%	7494
		<i>few tokens</i>	77.0%	6710
		<i>no restate</i>	80.0%	7367

Table 5: Prompt-based interventions on Qwen3-4B under GPTQ quantization for MBPP, reporting accuracy and CoT tokens. “—” denotes the baseline without additional prompt intervention.

sistently. For example, *no restate* cuts INT4 CoT tokens by 30.0% but drops accuracy from 90.0% to 81.0%; for INT3, all prompts reduce tokens while causing 5–11 point accuracy drops. This suggests prompting is a heuristic, unstable mitigation for token inflation rather than a reliable solution.

7.2 Calibration Data as a Lightweight PTQ Mitigation

Most PTQ methods rely on a small *calibration* corpus to estimate activation statistics or solve layer-wise reconstruction objectives (Frantar et al., 2022; Lin et al., 2023; Cheng et al., 2024). While calibration data is known to affect accuracy, we show it simultaneously shapes reasoning-token inflation.

We compare five sources: (i) *ultrachat_200k*, instruction dialogue (HuggingFaceH4, 2023); (ii) *wikitext* (Merity et al., 2016); (iii) *c4* (Raffel et al., 2019); (iv) *Math-CoT*, target-model reasoning traces generated from OpenR1-Math-220k problems (open-r1 team, 2025); and (v) *Code-CoT*, target-model reasoning traces generated from Codeforces problems.

Fig. 5 plots macro-average accuracy against average CoT length for Qwen3-4B under INT3 GPTQ. Calibration data has a large joint effect on both axes: switching from Wikitext to Math-CoT raises accuracy from 28% to 65% while cutting CoT length from 30k to 12k tokens. Reasoning-domain corpora (Math-CoT, Code-CoT) outperform generic text (Ultrachat, C4, Wikitext) on both dimensions. Since this improvement requires no retraining beyond the standard PTQ procedure, we adopt Math-CoT as the default calibration corpus for all main-paper PTQ experiments.

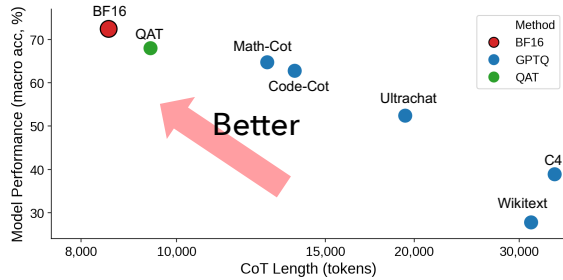


Figure 5: Accuracy-efficiency trade-off across calibration corpora for Qwen3-4B under INT3 GPTQ. Each point is a (GPTQ, corpus) pair; upper-left is better.

7.3 Quantization-Aware Training Achieves a Pareto Improvement

We evaluate *DiscQuant* (Chee et al., 2025), a recent QAT method that jointly optimizes weight rounding under end-to-end task supervision. We use 128 warmup steps, 1,024 total training steps, and group size 128. In Fig. 5, QAT reaches 68% accuracy at approximately 9.5k average CoT tokens, compared with 73% and 8.5k for BF16. In the tested Qwen3-4B INT3 configuration, it gives the strongest observed PTQ/QAT accuracy-length trade-off.

Why QAT can help. PTQ applies quantization post hoc to a model trained in full precision. Even when layer-wise reconstruction methods such as GPTQ reduce local quantization error, they do not directly optimize the model’s behavior over long reasoning trajectories. Small perturbations can accumulate across many decoding steps, changing not only the final answer but also how long the model continues to reason. QAT provides a different signal. By back-propagating task loss through quantize-dequantize operators, QAT adapts the weights to low-bit execution rather than only correcting post-hoc rounding error. More importantly for reasoning models, when QAT is performed on reasoning traces, the model observes the target reasoning behavior under simulated quantization. This gives training signal not only for preserving final-answer correctness, but also for maintaining the structure and length scale of the full-precision reasoning trajectory. In this sense, QAT can re-align the quantized model with the reasoning process it is expected to preserve, whereas PTQ mainly attempts to minimize local reconstruction error.

Table 6 summarizes the observed one-time preparation cost and quality-length trade-off. QAT reduces macro CTIR from 48.7% (GPTQ) to 13.6%

Model	Method	Prep.	Avg. acc. (%)	CTIR (%)
Qwen3-4B	BF16	0	72.5	0.0
	INT3 GPTQ	7 min	67.1	48.7
	INT3 AWQ	5 min	45.7	276.1
	INT3 QAT	20 min	68.0	13.6
Qwen3-30B	BF16	0	74.8	0.0
	INT3 GPTQ	20 min	74.7	19.0
	INT3 AWQ	15 min	68.4	32.8
	INT3 QAT	1.5 h	—	3.2

Table 6: Observed preparation-cost and quality-length trade-offs. Accuracy and CTIR are macro-averages over six reasoning tasks. Qwen3-30B QAT accuracy was unavailable in the recorded aggregate and is not inferred. Times are one-time model conversion/training costs on the authors’ H100 setup and exclude evaluation. All low-bit rows use group size 128.

on Qwen3-4B and from 19.0% to 3.2% on Qwen3-30B, at a larger one-time cost. Prompting and repetition penalties require no model preparation but can reduce accuracy; reasoning-trace calibration adds only the usual PTQ calibration cost but is domain dependent. Because these interventions are not all evaluated on an identical task-by-task grid, we do not claim universal Pareto dominance. QAT instead provides the strongest observed trade-off in our tested configuration, and its preparation cost can be amortized only when the model is served sufficiently often.

8 Conclusion

Post-training quantization is widely used to reduce LLM inference cost, but our results show that accuracy alone is insufficient for evaluating quantized reasoning models. Low-bit PTQ can preserve final-answer accuracy while inflating chain-of-thought length, creating a hidden test-time compute cost that offsets expected per-token efficiency gains. Median, trimmed, and both-correct analyses show that this effect is not solely explained by a few runaway or newly incorrect responses. Inflation is associated with longer and more repetitive traces, while identical-prefix diagnostics suggest that quantization broadens local next-token distributions. This evidence is descriptive rather than a complete causal account. Longer reasoning is not uniformly harmful to correctness, but it increases time to first visible token and reduces serving throughput. Evaluations of quantized reasoning models should therefore report reasoning-token usage and its distribution alongside accuracy.

Limitations

Our study focuses on weight-only INT3 and INT4 quantization. Other settings, including activation and KV-cache quantization, mixed precision, different group sizes, and hardware-specific kernels, may affect accuracy and generation length differently. Our main evaluation uses a fixed, randomly shuffled subset of each benchmark with greedy decoding. Although this eliminates sampling variation for a given checkpoint, it does not capture uncertainty arising from the finite evaluation set. We supplement the main results with full-set stochastic evaluations and distribution-aware statistics, but do not exhaustively repeat every model-quantizer-benchmark combination because evaluating long reasoning traces is computationally expensive. Broader sweeps may reveal configuration-specific effects not captured here.

We evaluate only open-weight reasoning models, primarily from the Qwen3 and Phi-4 families. Our additional output-length study broadens the evaluation to three newer model families, but it relies on publicly released third-party quantized checkpoints. Consequently, these experiments provide less experimental control than comparisons produced by quantizing identical base checkpoints with a unified pipeline. Our findings may therefore not generalize to closed-source models or to models that do not expose a separable reasoning trace. Results may also depend on model-specific termination behavior.

Acknowledgments

We sincerely appreciate the anonymous reviewers. Their insightful feedback helped significantly improve the quality of the paper. This research was supported by the National Science Foundation (NSF) under Grant No. 2441601. The work utilized the Delta and DeltaAI system at the National Center for Supercomputing Applications (NCSA) and Jetstream2 at Indiana University through allocation CIS240055 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296. The Delta advanced computing resource is a collaborative effort between the University of Illinois Urbana-Champaign and NCSA, supported by the NSF (award OAC 2005572) and the State of Illinois. UIUC SSAIL Lab is supported by research

funding from Google, Amazon, Microsoft, IBM, and AMD.

References

- Yamato Arai and Yuma Ichikawa. 2025. Quantization Error Propagation: Revisiting Layer-Wise Post-Training Quantization. In *Advances in Neural Information Processing Systems (NeurIPS 25)*.
- Hicham Badri and Appu Shaji. 2023. Half-Quadratic Quantization of Large Machine Learning Models. https://dropbox.github.io/hqq_blog/. Blog post.
- Jerry Chee, Arturs Backurs, Rainie Heck, Li Zhang, Jannardhan Kulkarni, Thomas Rothvoss, and Sivakanth Gopi. 2025. DiscQuant: A Quantization Method for Neural Networks Inspired by Discrepancy Theory. In *Proceedings of Thirty Eighth Conference on Learning Theory (COLT 2025)*.
- Wenhua Cheng, Weiwei Zhang, Haihao Shen, Yiyang Cai, Xin He, Lv Kaokao, and Yi Liu. 2024. Optimize weight rounding via signed gradient descent for the quantization of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11332–11350, Miami, Florida, USA. Association for Computational Linguistics. Code: <https://github.com/intel/auto-round>.
- DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*.
- Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. 2020. Learned step size quantization. In *ICLR*.
- Elias Frantar, Saleh Ashkboos, Torsten Hoeftler, and Dan Alistarh. 2022. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. *arXiv preprint arXiv:2210.17323*.
- Advait Gadhihar, Riccardo Grazi, and James Hensman. 2025. Optrot: Mitigating weight outliers via data-free rotations for post-training quantization. In *Advances in Neural Information Processing Systems*.
- HuggingFaceH4. 2023. UltraChat 200k. https://huggingface.co/datasets/HuggingFaceH4/ultrachat_200k. HuggingFace dataset. Accessed: 2026-01-23.
- Kimi Team. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. *arXiv preprint arXiv:2501.12599*.
- Zhen Li, Yupeng Su, Songmiao Wang, Runming Yang, Congkai Xie, Aofan Liu, Ming Li, Jiannong Cao, Yuan Xie, Ngai Wong, and Hongxia Yang. 2025. Quantization Meets Reasoning: Exploring and Mitigating Degradation of Low-Bit LLMs in Mathematical Reasoning. *arXiv preprint arXiv:2505.11574*.

- Yesheng Liang, Haisheng Chen, Zihan Zhang, Song Han, and Zhijian Liu. 2025. Paroquant: Pairwise rotation quantization for efficient reasoning llm inference. *arXiv preprint arXiv:2511.10645*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. *arXiv 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. 2023. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *arXiv preprint arXiv:2306.00978*.
- 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*.
- Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2024a. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. In *Findings of the Association for Computational Linguistics (ACL 2024)*.
- Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024b. SpinQuant: LLM Quantization with Learned Rotations. *arXiv preprint arXiv:2405.16406*.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. *CoRR*, arXiv preprint abs/1609.07843.
- open-r1 team. 2025. OpenR1-Math-220k. <https://huggingface.co/datasets/open-r1/OpenR1-Math-220k>. HuggingFace dataset. Accessed: 2026-01-23.
- OpenAI. 2024. OpenAI o1 System Card. *arXiv preprint arXiv:2412.16720*.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.
- Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2023. OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models. *arXiv preprint arXiv:2308.13137*.
- Sayeh Sharify, Utkarsh Saxena, Zifei Xu, Wanzin Yazar, Ilya Soloveychik, and Xin Wang. 2025. Post training quantization of large language models with microscaling formats. In *Advances in Neural Information Processing Systems*.
- Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. QuIP#: Even Better LLM Quantization with Hadamard Incoherence and Lattice Codebooks. *arXiv preprint arXiv:2402.04396*.
- Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. 2022. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. *arXiv preprint arXiv:2211.10438*.
- Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers. *arXiv preprint arXiv:2206.01861*.

A Additional background on LLM quantization

A.1 LLM quantization

Quantization preliminaries. Most practical LLM quantization methods use affine (uniform) quantization, mapping a floating-point tensor x to an integer tensor q with scale $s > 0$ and zero-point z :

$$q = \text{clip}(\text{round}(x/s) + z, q_{\min}, q_{\max}), \hat{x} = s \cdot (q - z), \quad (5)$$

where $\hat{x}$ denotes the dequantized approximation of x . For Transformer weights, deployment-oriented pipelines typically adopt *weight-only* quantization (e.g., INT4 weights with FP16/BF16 activations) because activation and KV-cache quantization can be more sensitive and often require kernel/hardware support. In addition, most LLM methods quantize weights *per-channel* (or per-group) rather than per-tensor to reduce outlier-induced errors; in 4-bit regimes, group-wise quantization (e.g., group size 32/64/128) is common to balance accuracy and metadata overhead (Frantar et al., 2022; Lin et al., 2023; Shao et al., 2023).

Post-training quantization (PTQ). PTQ converts a pretrained model to low precision without full backpropagation-based training. It is attractive because it is efficient, requires limited calibration data (sometimes only a few hundred sequences), and can be applied to a frozen checkpoint. Modern PTQ methods for LLMs largely fall into two families:

(i) *Second-order / error-compensating weight quantization.* GPTQ introduced a practical post-training procedure that quantizes weights layer-by-layer while approximately accounting for output error using curvature information (typically via a Hessian approximation from calibration data)

(Frantar et al., 2022). The key idea is to choose quantized weights that minimize an output reconstruction objective rather than performing naive round-to-nearest (RTN). GPTQ and its descendants (including highly optimized kernels in popular tooling) are widely used for INT4/INT3 weight-only deployment.

(ii) *Saliency- and activation-aware weight quantization.* AWQ observes that not all weights contribute equally to the output distribution; it protects “important” channels/weights by rescaling activations and applying clipping/quantization rules that reduce error on salient subspaces (Lin et al., 2023). In practice, AWQ is often robust at INT4 for instruction-following models, and it is commonly paired with efficient inference kernels.

Beyond GPTQ/AWQ, the PTQ landscape includes methods that target different bottlenecks: SmoothQuant addresses activation outliers by migrating quantization difficulty from activations to weights via a mathematically equivalent rescaling transformation, enabling efficient W8A8 inference while preserving accuracy (Xiao et al., 2022). OmniQuant optimizes quantization parameters (e.g., clipping thresholds and equivalent transformations) in a data-efficient manner to approach QAT-like performance while retaining PTQ efficiency (Shao et al., 2023). Many additional variants exist (e.g., alternative rounding, clipping, mixed precision, and KV-cache/activation quantization), but a recurring theme is that PTQ aims to *approximate* the full-precision model with minimal adaptation cost.

B Metrics

We evaluate quantized reasoning models along two axes: task performance and reasoning efficiency.

- **Task performance.** We follow each benchmark’s official protocol. By default we report exact-match accuracy; for math benchmarks we normalize the output formatting before comparing the extracted final answer to the reference, and for code benchmarks we report Pass@1.
- **Reasoning and answer length.** Reasoning model outputs typically consist of a chain-of-thought (CoT) trace followed by an (often shorter) final answer. We report the token counts of the two portions separately, which lets us distinguish changes in reasoning length from changes in answer length.

Benchmark	N	Metric	Domain and version
AIME25	30	Exact match	2025 math competition
GSM8K	1,319	Exact match	Grade-school mathematics
LiveCode-Bench	182	Pass@ k	Code generation; V5–V6
MBPP	500	Pass@1	Code generation; official test split (974 examples across all splits)
GPQA-Diamond	198	Accuracy	Graduate-level science; Diamond split
BBH	6,511	EM/acc.	General reasoning; 23 tasks
HellaSwag	10,042	Accuracy	Commonsense reasoning
PIQA	1,838	Accuracy	Physical commonsense
MMLU-Pro	12,032	Accuracy	Multitask knowledge

Table 7: Benchmark pools and evaluation protocols. The main experiments use the fixed subsets described in §3.

C Prompt based methods

We evaluate three prompt-based interventions that aim to suppress redundant reasoning, encourage token-efficient reasoning, and directly discourage explicit CoT generation. Here is the detail of the prompts.

- *do not think:* During reasoning, do not think.
- *less tokens:* During reasoning, use as few tokens as possible.
- *no restate:* During reasoning, do not restate previous steps unless necessary for correctness. Each step should introduce a new inference, calculation, or decision. If the current thought does not add new information, stop that line of reasoning and move on.

D Additional evaluation details

This appendix provides benchmark descriptions, prompt templates, decoding settings, and additional evaluation details.

D.1 Benchmarks and evaluation protocol

Table 7 summarizes the key characteristics of the benchmarks used in our evaluation.

We use the default prompt for reasoning models. More specifically:

“Please reason step by step, and put your final answer within `\boxed{}`.”

Checkpoint	Size / arch.	Release	Context
Qwen3-4B-Thinking-2507	4B / dense	2025-07	256K
Qwen3-30B-A3B-Thinking-2507	30B (3B active) / MoE	2025-07	256K
Phi-4-Reasoning-Plus	14B / dense	2025-04	32K

Table 8: Model metadata. All evaluations use temperature 0 and top- $p = 0.95$; top- k is 20 for Qwen3 and 50 for Phi-4, but neither sampling parameter affects greedy decoding.

Dataset	Metric	Subset	Full set [95% CI]
GSM8K	Accuracy	94.0	94.1 ± 0.7 [93.2, 95.0]
	Tokens	1,139	1,180 [1, 124, 1, 236]
MBPP	Accuracy	92.0	91.1 ± 0.6 [90.4, 91.8]
	Tokens	4,692	4,800 [4, 552, 5, 048]
BBH	Accuracy	87.0	86.9 ± 0.4 [86.4, 87.4]
	Tokens	2,674	2,750 [2, 626, 2, 874]

Table 9: Qwen3-4B BF16 robustness check. Subset results use temperature 0; full-set results use temperature 0.6 over five runs. Accuracy is mean $\pm$ standard deviation, followed by its two-sided 95% t -interval; token intervals use the same five-run construction.

D.2 Model details

Table 8 lists the evaluated checkpoints, model metadata, and decoding settings used throughout the paper.

Hardware. All experiments (PTQ and evaluation) are run on NVIDIA H100 GPUs.

E Statistical Robustness

The main evaluation is deterministic for a fixed checkpoint and prompt because it uses greedy decoding. To check whether the fixed 100-example subsets track broader benchmark behavior under a stochastic decoding policy, we additionally evaluate Qwen3-4B BF16 on the complete GSM8K, MBPP test, and BBH evaluation sets with temperature 0.6 over five runs. Table 9 shows that the subset point estimates are close to the full-set estimates, although the MBPP subset accuracy is slightly above the interval computed from the five stochastic runs. This check does not supply uncertainty estimates for every quantized configuration; we therefore complement the mean with the paired medians, trimming, and correctness-conditioned statistics reported in the main text, and treat finite-sample uncertainty as a limitation.

Model	Setting	BF16	Quant.	Δ
Qwen3-4B	INT4 RTN	41.20	113.10	71.90
	INT4 AWQ	41.20	42.66	1.46
	INT4 GPTQ	41.20	43.19	1.98
	INT4 HQQ	41.20	52.73	11.52
	INT3 AWQ	41.20	50.76	9.56
	INT3 GPTQ	41.20	63.70	22.49
	INT3 HQQ	41.20	77.08	35.88
Qwen3-30B	INT4 RTN	24.08	30.12	6.03
	INT4 AWQ	24.08	34.87	10.78
	INT4 GPTQ	24.08	26.51	2.42
	INT4 HQQ	24.08	27.80	3.72
	INT3 AWQ	24.08	35.29	11.21
	INT3 GPTQ	24.08	34.76	10.67
	INT3 HQQ	24.08	44.98	20.89

Table 10: Average number of fixed self-doubt markers per response, macro-averaged across the six reasoning tasks. Δ is quantized minus BF16. Because raw counts grow mechanically with response length, this is a descriptive trace diagnostic rather than evidence that marker density increases.

F Behavioral Diagnostic Details

For the self-doubt diagnostic, we search generated reasoning spans for the following fixed, case-insensitive phrase list: “wait”, “hold on”, “let me check”, “let me verify”, “let me double-check”, “let me reconsider”, “let me recalculate”, “double-check”, “recheck”, “verify”, “not sure”, “make sure”, “let’s check”, and “let’s verify”. Table 10 reports raw matches per response. The list contains related and partially overlapping expressions and the raw count is correlated with response length, so we use it only to quantify the qualitative verification style, not as a length-normalized behavioral metric.

G Additional Model Families

We perform an additional scope check using recently released models and public INT4 checkpoints. The exact Hugging Face repositories are GLM-5.2 (BF16), GLM-5.2 (AWQ), MiniMax-M3 (BF16), MiniMax-M3 (GPTQ), MiniMax-M3 (AWQ), Qwen3.6-27B (BF16), Qwen3.6-27B (GPTQ), and Qwen3.6-27B (AWQ). We verified that these repository identifiers resolve at camera-ready preparation time. Because the quantized checkpoints are maintained by third parties, this experiment cannot rule out conversion or configuration differences and is reported separately from the controlled main evaluation.

Task	BF16	INT4 GPTQ	INT4 AWQ
GLM-5.2 (753B)			
AIME25	20,269.8	—	22,257.9
GSM8K	1,373.2	—	4,050.9
GPQA-D	32,432.4	—	34,628.4
BBH	5,673.8	—	7,699.9
MBPP	8,023.9	—	10,259.6
Δ_{macro}	—	—	55.0%
MiniMax-M3 (427B)			
AIME25	22,798.2	50,023.1	54,251.4
GSM8K	270.3	274.2	912.4
GPQA-D	10,510.0	25,776.6	25,039.9
BBH	629.1	3,069.5	5,641.3
MBPP	4,475.0	15,409.1	4,380.9
Δ_{macro}	—	179.7%	261.7%
Qwen3.6-27B			
AIME25	26,303	26,885	26,109
GSM8K	7,770	9,615	8,976
GPQA-D	9,804	10,211	11,308
BBH	4,889	4,239	5,248
MBPP	7,869	16,329	9,177
Δ_{macro}	—	24.9%	10.8%

Table 11: Average generated output tokens for additional recently released models. LiveCodeBench is omitted because of its evaluation cost. Macro Δ is the unweighted average of the five task-level relative output-length changes. These third-party checkpoints provide a scope check for length inflation, not an accuracy-preservation result or a controlled comparison of quantizers.

H Mitigation Details

Table 12 gives the per-task calibration-data comparison, and Table 13 gives the token counts underlying the compact cost comparison in the main paper. The preparation times in Table 6 measure model conversion or QAT on the authors’ H100 setup and exclude downstream evaluation. They should be interpreted as setup-specific wall-clock costs rather than hardware-independent complexity estimates.

I Correctness-Conditioned Token Inflation

We provide two complementary correctness slices. The strict both-correct results reported in the main text directly test whether a trajectory becomes longer while preserving the answer. Here we partition Qwen3-30B instances only by whether BF16 is correct. This second view includes both preserved and flipped quantized outcomes and is therefore diagnostic rather than a clean accuracy-preservation test. Figures 6 and 7 show that inflation is generally stronger and more consistent on BF16-correct instances, whereas the BF16-wrong subset is more variable and sometimes shows deflation. We do

Task	Math-CoT	Code-CoT	C4
AIME25	25,402	22,057	44,095
GSM8K	1,456	2,329	7,678
GPQA-D	8,912	10,142	31,938
BBH	3,946	4,942	9,612
LCB	23,773	19,871	62,948
MBPP	9,704	10,803	54,172

Table 12: Average CoT tokens for Qwen3-4B INT3 GPTQ under different calibration corpora. Reasoning-trace calibration is consistently shorter than generic C4, but matching the calibration domain to a particular benchmark does not always produce the shortest trace.

Task	BF16	GPTQ	AWQ	QAT
Qwen3-4B				
AIME25	21,307	25,402	30,924	22,017
GSM8K	1,139	1,456	1,529	1,155
GPQA-D	7,796	8,912	13,876	8,979
BBH	2,674	3,946	7,599	2,857
LCB	13,459	23,773	61,590	17,099
MBPP	4,692	9,704	49,609	6,007
Qwen3-30B				
AIME25	18,587	18,857	21,827	18,457
GSM8K	865	964	982	874
GPQA-D	6,071	7,233	7,352	6,252
BBH	1,692	1,932	2,194	1,783
LCB	15,374	17,583	22,131	16,080
MBPP	3,330	5,114	5,700	3,521

Table 13: Per-task average CoT tokens for the INT3 mitigation-cost comparison in Table 6. All low-bit methods use group size 128.

not interpret this as evidence that extra reasoning is always harmful: a small number of responses can flip from wrong to right, and a larger study would be needed to estimate that transition reliably.

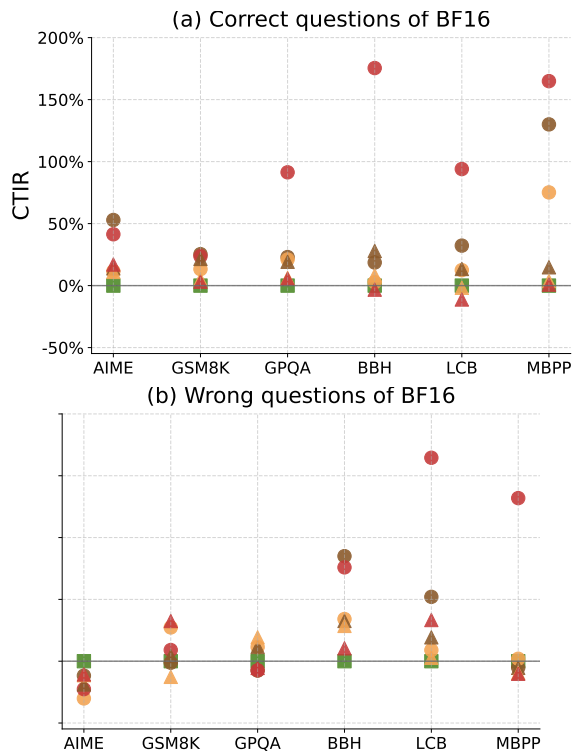


Figure 6: Qwen3-30B correctness-conditioned CTIR by benchmark. The upper and lower panels contain BF16-correct and BF16-wrong instances, respectively; each subset can include preserved or flipped quantized outcomes. Green squares denote BF16; brown, orange, and red encode AWQ, GPTQ, and HQQ, with circles for INT3 and triangles for INT4.

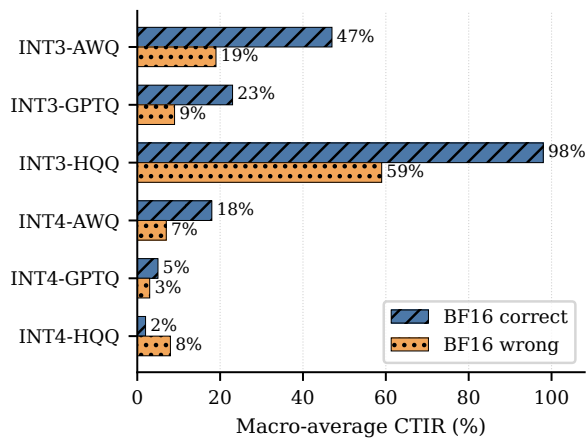


Figure 7: Qwen3-30B CTIR macro-averaged over benchmarks after partitioning instances by whether BF16 answers correctly. Each subset can include either preserved or flipped quantized outcomes.